## SESSION

## CLOUD COMPUTING, ANALYSIS, AND PERFORMANCE EVALUATION + BIG DATA

### Chair(s)

**TBA**

# Cyber Analysis Grids for Asynchronous Distributed Cloud Services

**R. William Maule, Ph.D.**
Information Sciences Department, Naval Postgraduate School, Monterey, CA, USA

**Abstract -** *Comprehensive application performance assessment across geographically diverse private and public clouds will require an integrated grid of analytics services able to characterize services and processes in both synchronous and asynchronous communication environments. The analysis grid assumes continuous in-line data collection of packets at cloud nodes and aggregates services across the grid. Sampled data from service performance measurement tools are integrated to assess global enterprise performance and cyber operations across distributed public and private clouds. This paper presents analysis variables and data collection attributes important for a comprehensive assessment of cyber operations within a diverse and dynamic global cloud enterprise. Analysis scenarios address asynchronous services, data synchronization, and cyber security assessment. Scenarios are abstracted from current test, measurement and analysis issues ongoing in field experimentation. Variables, attributes and tools to measure application and service performance for implementations of distributed clouds are advanced.*

**Keywords:** Cloud, Analytics, Cyber, Grid, Enterprise

## 1 Introduction

Application performance monitors with robust metrics and fine-grained measurement capabilities are critical for comprehensive assessment of global hybrid cloud networks. Real-time and continuous assessment from an out-of-band analytics grid can help secure enterprise systems and ensure adequate performance of mission critical applications. Cyber analysis tools have evolved from a focus on Open Systems Interconnection (OSI) Layers 2-4 for Media Access Control (MAC), Internet Protocol (IP), and Transport Layer analysis (respectively) to the current focus on OSI Layer 7 for application, service and process monitor and assessment. Layer 7 analytics capabilities for diverse and geographically distributed hybrid clouds are not yet sufficient for comprehensive analytics, especially when the cloud network is global, faces communication-challenged operating environments, and must support cloud nodes and machine-to-machine (M2M) synchronization across dynamic nodes. This paper advances some consideration for measurement and analytics—from traditional network packet capture to application layer service and process assessment. The concept of an out-of-band analytics grid is advanced as a means to mitigate content synchronization problems in asynchronous services, such as might be found in "Internet-of-everything" and dynamic device scenarios.

Historically, sensor-based data sampling and capture services have enabled network and application monitoring sufficient to characterize sampled data for a comprehensive enterprise view of networks, protocols, and overall communications. Tools to additionally address applications and services in a dynamic and diverse cloud-based enterprise, and able to reach within cloud virtual machines to assess processes, process security, and process dependencies, are loosely available but not sufficiently integrated to provide comprehensive real-time analytics. An out-of-band analytics grid, similar to telecommunications out-of-band signaling systems, might benefit those required to perform cyber operational assessment of application content in globally distributed networks, especially those composed of clouds and virtual machines at one extreme, and Internet-of-everything devices at the other extreme, and the collective married with asynchronous communications and content synchronization requirements.

In the above scenario the cloud network becomes the backbone on a global information grid. The smaller cloud nodes the intermediary devices, and the end-user devices the service initiators in service-based publication and subscription scenarios. In this context, cloud services can be measured through traditional layer 2-4 network tools, layer 7 application tools, virtual machine (VM) analysis suites, and specialized tools from cloud service providers. A first step toward a systematic understanding of cloud network analytics requirements is to address the variables required for analysis of multi-layered architectures. This can begin with categorization of key variables required for analysis and the metrics necessary to effectively evaluate services and processes. Once established, Quality of Service (QoS) metrics can be established to help measure primary components in real-time. Metrics therein provide a foundation for assessment of future services including QoS contracts and assessment of web services between heterogeneous, distributed clouds. The context and referenced scenarios assume a need for real-time or near-real-time analytics vice post-capture analysis.

Toward this objective, this paper examines some of the measurement variables available to those responsible for network, application, process and security analysis—herein collectively termed "cyber analysis". The collective and integrated use of the tools for distributed clouds is considered a "cyber analysis grid" and herein is considered out-of-band, separate network able to mitigate asynchronous communications.

## 2 Cloud Analytics

Researchers have addressed cloud analytics from multiple perspectives. Some have separated the physical attributes of communication, computation, memory and storage from capacity measures such as transaction speed, availability, latency, reliability, and throughput [1]. Cloud servers and databases have been given stress tests in various configurations using both Web Services Description Language (WSDL) and Representational State Transfer (REST) queries to determine application availability and responsiveness [2].

While the history of analysis for distributed systems is well established, the defining characteristic of clouds as distributed systems is virtualization – and researchers have accordingly assessed attributes of cloud virtualization for dependability and associated measures [3]. In a similar vein, studies have examined more narrowly the performance and scalability metrics in cloud Software as a Service (SaaS) offerings to establish baselines [4] which can be applied to cloud offerings from different vendors.

Tools have been developed to look specifically at Layer 7 applications, services and processes and some reach into cloud virtual machines, although few reach to the other extreme and into end-user devices. Application Performance Monitoring (APM) tool suites can be applied within the cloud and are available in SaaS offerings by cloud vendors. When applied against cloud resources, APM tools can provide insight into not only application and service performance but to underlying network infrastructure.

Gartner defines APM as tracking, in real time, the execution of the software algorithms that constitute an application; measuring and reporting on finite hardware and software resources that are allocated to be consumed as the algorithms execute; determining whether the application executes successfully according to the application owner's requirements; recording latencies associated with execution step sequences; and determining why an application fails to execute successfully, or why resource consumption and latency levels depart from expectations [5].

To achieve the capabilities above, networks and applications need to be mapped, transactions profiled, and analytics applied to event processes to determine operational patterns. Once the metrics for evaluation have been established, and some operational baselines have been set, researchers can evaluate the metrics against those baselines to assess the viability of new applications or services.

An advantage of programming services for clouds is that assessments against baselines can be incremental—with small easily integrated web services cumulatively evolving to provide intended services. Metrics and measurement in this scenario can be more straightforward. However, this environment of easy reuse and integration means that the cloud software infrastructure is extremely flexible and therein complex for analysis; hence, the need to evaluate applications and services within the context of the particular cloud instance under evaluation. The concept of an integrated and comprehensive approach, and real-time end-to-end analytics, is lost.

Additionally, de-composition of services can introduce cloud-specific bias and therein prevent comprehensive assessment. For example, a composite application may subscribe to a data set from a remote web service, process and otherwise manipulate that data, add new data, and then publish the composite as a web service. A network outage that impacts any dependent data stream may cause or errors or incorrect data in the composite application, or for other applications that subscribe to the composite, and so on. Fault traces in highly integrated composite applications that use widely dispersed or federated web services can be extremely difficult to monitor and decompose. Interpreting composite data delivered during a failure can lead to incorrect decisions. To prevent this, some minimal set of performance and QoS standards must be developed, agreed to within the federation, adhered to, and modified when necessary [6].

## 3 Cloud Security

The debate continues on whether clouds increase or decrease security; the underlying cyber issues are too often ignored—cloud or not. Most of those that wear the "cyber" or "security" cap in their organizations have a background in the network area and the focus of security is on perimeter defense. Yet, the perimeter was breached years ago and the varmints are already in the enterprise—necessitating a very different cyber strategy. In many aspects we are fighting yesterday's cyber wars.

Added to this is that the multi-tenancy aspect of clouds may facilitate cyber-attacks on a massive scale. In addition to traditional security measures for servers and computers each virtual machine will need to be carefully monitored to ensure the hypervisor is not compromised [7]. Complicating matters is that today's anti-virus, malware and firewall detection and protection methods are proving to be ineffective in multi-tenant cloud environments [8].

Since consumption of an input causes a change to system memory and resultant processes—no matter how seemingly benign—every input is essentially a program [9]. Exploitation can be a simple form-based data inject that triggers existing bugs in software at one extreme—or a persistent threat loaded years ago from a popular web site and waiting all these years for activation. Various methods have been advanced to attempt to mitigate such threats.

A layer of middleware can derive context for role-based access controls to assist with content authorization control while simultaneously tracing user access to system resources [10]. Advancing the "analytic grid" concept, in a related context researchers have modeled frameworks that feature a user data collector, cloud service component, and cloud intrusion detection with encrypted communications between each component [11]. Similarly, a distributed architecture which collects data to provide intrusion detection in hierarchical and multi-layer architectures has been advanced which uses distributed security components to perform complex event correlation analysis [12].

While all of the above are helpful they are steeped in perimeter defense vice security at the process level. Agreed, as a first step in security the perimeter defense is required. The ability to aggregate data from security sensors and send this data via an agent across an analytics grid for collective processing helps build a more robust infrastructure—which can be accomplished with the traditional layer 2-4 tools. Next is to evaluate deep into the Layer 7 applications and their services and processes.

## 4 Service And Process Evaluation

Taken for granted within the programming community, but perhaps not apparent to users is that clouds provide services and are therein a facet of a Service Oriented Architecture. Platform as a Service (PaaS), Infrastructure as a Service (IaaS), Software as a Service (SaaS) and so on are all services of clouds. Clouds are therein both a component of a Service Oriented Architecture (SOA) and can themselves host or provide a SOA. As such, SOA engineering tools, methodologies and algorithms offer an additional means for cloud network analysis [13]. SOA approaches tend toward fine-grain process analysis and may provide useful perspectives to assess risk from component interaction. In this context the focus would be on frameworks for component assessment and methods for component performance and security evaluation.

Monitoring, analyzing, and understanding component interaction is difficult—yet essential to solving and preventing performance and QoS failures [14]. Cumulative or composite services multiply the number of component interactions that must be monitored, controlled, and debugged due to this increased number of components and processes [15]. This is accentuated in widely distributed hybrid clouds, with asynchronous communications and the need for content synchronization, and with the ever-growing plethora of end-user devices needing to subscribe to the services. Measures of service and resource availability in composite, cumulative services becomes a primary concern as the paradigm shifts from single process to integrated services [16, 17].

In addition to the complexity of composite services is the cumulative impact of operations in each of the OSI layers, e.g., process, service, routing, transformation, etc. [18]. The loosely coupled and heterogeneous nature of cloud services necessitates well-defined metrics to diagnose performance [19]. One technique is to define desirable quality attributes and then trace the metrics required to measure them, and at different levels of abstraction [20]. Another is to assess service granularity and service coupling between services and clients [21]. Measures may include process speed, system reliability, throughput, and availability [22].

Services are often re-used across multiple projects [23]. While there are solid heuristics for evaluation of specific service projects [24], heuristics to address the impact of component variations are generally absent. Yet, practical limitations of federation performance risk in distributed, asynchronous clouds requires such heuristics. A basis for analysis may be adopted from research on architectural frameworks [25]. Performance assessment might apply frameworks for service interaction against user requirements [26, 27] to formalize analysis of component relationships, object interactions, and associated rules.

Finally, it must be borne in mind that service interactions with clouds are rarely absolute and often constantly changing. Analysis tools which assess patterns may be difficult to scale to global cloud services. Analysis frameworks must therein be at a rather high level of abstraction, which further complicates analysis.

## 5 Cyber Analytics

As a preliminary assessment to help categorize network and application analysis tools and variables several laboratory and field tests were developed. Component interaction for distributed cloud nodes was assessed that included six (6) physical hosts supporting thirty (30) virtual servers with representative enterprise SOA builds on VMware, Hyper-V, and Xen virtual machines. The focus was assessment of the tools and analysis approaches vice specific component or application evaluation. Table 1 outlines the test components and defines the broad categories for preliminary analysis of tools and analysis methods to more fully understand component and process interaction in a distributed cloud network.

TABLE I. TECHNICAL ANALYTICS FRAMEWORK.

| # | Service | Variables / Attributes |
|---|---|---|
| 1 | Storage | ***Data retrieval:***<br>1. *Authoritative data sources*<br>2. *Content prioritization*<br>3. *Data synchronization*<br>4. *Conflict resolution*<br>5. *Archival operations* |
| 2 | Repository | ***Analytics metadata:***<br>1. *Hardware/software clusters*<br>2. *Database performance*<br>3. *VM performance and caching*<br>4. *Security authentication*<br>5. *Content authorization*<br>6. *Search and pattern recognition* |
| 3 | Grid | ***Information collection:***<br>1. *Database queries and latencies*<br>2. *Process and performance*<br>3. *Shared or dedicated resources*<br>4. *Packet and/or flow data*<br>5. *Physical or virtual collection*<br>6. *Agent processes* |
| 4 | Virtualization | ***Cloud management:***<br>1. *VM systems management*<br>2. *VM performance characteristics*<br>3. *IaaS, PaaS, SaaS attributes*<br>4. *Shared resource metrics*<br>5. *Security and systems messaging* |
| 5 | Services | ***Application capabilities:***<br>1. *User and machine interfaces*<br>2. *Individual/composite processes*<br>3. *Content discovery and delivery*<br>4. *Utilization statistics* |
| 6 | Containers | ***Processing services:***<br>1. *Component interoperability*<br>2. *Deployment compatibility*<br>3. *Processes reliability*<br>4. *Performance QoS* |
| 7 | Registry | ***Initiation / Acknowledgement:*** |

| | | |
|---|---|---|
| | | 1. *Object permissions*<br>2. *Replication and synchronization*<br>3. *Lookup throughput, latency* |
| 8 | Service Bus | ***Federation Messaging:***<br>1. *Cache and queue*<br>2. *Interface / exchange*<br>3. *Message interoperability*<br>4. *M2M compatibility*<br>5. *Throughput, latency*<br>6. *Transmission errors* |

The intent is to not only develop a methodology for understanding interaction but also a means to structure data collection nodes in key locations. Again, the desired end state is an analysis grid capable of fine-grained analysis and sufficient for cyber operational assessment of distributed cloud networks to include various communication scenarios and end-user devices. Analysis methodology and supporting tools will need to be sufficient to assess both synchronous and asynchronous communications, and in the latter the synchronization processes required for content management in dynamic, complex, multi-layered composite transactions.

Performance variables in the ***storage*** tier will be assessed based on the intended use of the services—in-lieu of tests on hardware functions such as memory, processor speed or caching services. The concern herein is with the location of data in widely distributed clouds with 100+ nodes and the analysis metrics required to address authoritative data sources, and content prioritization, synchronization, and conflict resolution. Example builds will range from small and optimized for specific local functions at one extreme, to large and capable of federation across a global cloud backbone at the other extreme. Archival operations will be a concern at all nodes. There are performance variables associated with media selection and associated metrics for throughput, latency and capacity.

The metadata ***repository*** will be considered a function of an in-memory or persistent database, XML schema, or similar. Performance characteristics, metrics and variables will include those of a traditional database as used for caching, security authentication or authorization, as well as content-specific metadata for search and associated content pattern recognition. Clustering and virtualization on overall metadata repository operations can be addressed. Traditional metrics for database analysis can be applied to the metadata repository, plus, performance metrics specific to user access—such as performance issues in virtualization and distributed server clusters.

As an example in this area, Table II provides the results of a test of basic repository functionality in the test environment and exposes some basic metrics. Eighteen (18) tests were run with only one (1) failure and no errors for an overall success rate of 94.44%. Time required to run the assessment tests was 0.437 seconds. Tests captured typical user sessions in secured operations and then replayed those interactions with increasing load and number of users. For QoS capability validation, future tests might record Table II metrics at periodic time intervals while increasing load on the Repository, and compare recorded metrics with baseline measures of Table II.

TABLE II. METADATA REPOSITORY TEST METRICS.

| Metric | Status | Time |
|---|---|---|
| Test for Directory Structure | Success | 0.016 |
| Test Properties File For False Positive | Success | 0.000 |
| Test Properties File For CMEE | Success | 0.000 |
| Test Properties File For Database | Success | 0.000 |
| Test Event Setup | Success | 0.000 |
| Test Encryption Password | Success | 0.000 |
| Test SSL URL | Success | 0.281 |
| Test Single Substitution | Success | 0.063 |
| Test Single Match | Success | 0.000 |
| Test Number Sub String Match | Success | 0.000 |
| Test XQL Tool Loop Children | Success | 0.000 |
| Test UTF8 Encoded XML | Success | 0.015 |
| Test Reading Build Tag From Jar | Success | 0.000 |
| Test Invalid XML String | Success | 0.016 |
| Test Parse Roundtrip | Success | 0.000 |
| Test XPath Query | Success | 0.000 |

Returning to Table I, the "***grid***" is herein considered specific to analysis of independent cloud nodes and the aggregate of that analysis. Specifically the analytics available from layer 1-7 data capture and the processing tools that render statistics on applications, services or processes. Current commercial tool offerings in grid computing tend to focus on either the storage or application tier—we are concerned with latter. Grid performance measurement in this context can be collected through raw packet capture, header information from packets, flow data from routers or other devices, or via agents added to hosts on cloud nodes. Agents remain resident on the host and gather information from specified applications, services and processes and send that data to the central console, portal or dashboard. To note in this example is that the use of an agent can add to host latency, reduce the host's working memory, or otherwise detrimentally impact the processes on which it is reporting. Multiple agents on a host may conflict and further degrade performance. So, introduction of an agent on a host can introduce performance risk—the solution can become the problem. However, in some contexts, the performance tradeoff is worth the risk. For example, a grid agent may not only monitor a process but intervene to control that process should a malfunction or security breach necessitate intervention. A final variable is whether the data collection grid is composed of physical, dedicated devices, or embedded within virtual machines. If the latter then the amount of physical assets assigned to the analysis grid, such as memory, caching and processing, will impact the analysis process as well as the hosts being monitored. While the overall concept is an out-of-band analytics grid, the precise monitoring of collective processes will require some level of machine intervention—agent or otherwise.

***Virtualization*** is assumed throughout each cloud node. Assessment addresses the various levels of resource sharing between VMs as well as the degree and type of separation between virtual machines and shared resources—such as storage, network interfaces, and security mechanisms including firewalls and network access controls. Messaging buses and apparatus between VMs both internally and between cloud nodes will address queues, caching, and messaging buses. In addition is the environmental context of the different virtualization approaches, including VMware, Xen and Hyper-V. Virtualization operational metrics can address performance measures in virtual machine creation, cloning, failover, and deployment.

***Services*** exist with the application server or middle tier. Services process messages across servers and clouds on a service bus, and coordinate this via a registry. So, assessment of applications in a SOA cloud necessitates an understanding of the physical and logical "plumbing" of the architecture—which includes the services and their operations. Without a complete understanding of the "plumbing" the architecture will never really be understood or secure. In this aspect the servers and their virtual machine instances can be measured for physical properties of the cloud node and the software processes of the services, bus, and registry. Performance variables might include process kills and restarts that impact services, latency within or across services in composite applications, and interfaces between services and to hardware or user interfaces.

***Containers*** are the building blocks of service deployment. Container metrics are generally for the processes they support. The author's laboratory and field tests have evaluated containers supporting a range of configurations, from a simple container in one hardware box such as a JVM and application server, to a container that spanned geographically distributed hosts at the other extreme. There are related options for service deployment, and there are different QoS metrics specific to each configuration.

***Registry*** metrics can help assess service invocation, metadata management processes, transport, and QoS. Enforcement mechanisms report whether component registrations meet contractual obligations. Metrics can be applied for each step of the registration process, including the WSDL forms that have been published to a Universal Description Discovery and Integration (UDDI) service, reference information for service providers, the endpoint interface specification to enable programs to connect M2M to services with associated policies and transformations.

The ***service bus*** provides communications between applications within a cloud node, and serves as the communications conduit between nodes in a distributed cloud architecture. As such, the service bus is the lifeline of a distributed cloud architecture and of the analytics grid atop that architecture. As such the service bus is a potential choke point. Failure of service interoperability or message throughput can cause a federation-wide system failure. At a high level, metrics consider interoperability and logical correctness, latency, transformations, and service interfaces. Current analysis techniques in this area tend to focus on operational variables, or on content performance variables in content-based routing. Table III provides more detailed metrics for service bus tests, to be gathered at each node. In an analytics grid, baseline metrics would be established over a range of loads at each node, then the metrics would be synchronized at the central node.

TABLE III. SERVICE METRICS.

| Metric | Description |
|---|---|
| ***Execution Time*** | Time between message reception at the transport and exceptions or responses; if the transaction aborts, and messages placed back in the queue, each retry de-queue counts as a message. |
| ***Success / Failure Ratio*** | The number of messages that result in an exit with the system error handler or in an exit with a reply failure action (Total Messages - Number of Messages with Errors) / Messages with Errors |
| ***Messages with Errors*** | Messages with WS-Security errors; validation errors and the count of validation actions that have failed—to include proxy services. |
| ***Status Messages*** | Service mediation metrics on requests and responses; routing status between service endpoints; conditional status messages with metrics on processing and transformation between service endpoints. |
| ***Message Routing*** | Metrics on message content, multicast or multi-path messages; dynamic service provisioning, versioning; data element values; transformations applied to messages to multiple destinations, SLA-based changes; conditional checks and metrics on branch statements, values of data elements that determine routing logic. |
| ***Service Level Agreements*** | Contractual thresholds for availability, performance, and queuing. |

At a distinctly different level in our analytics grid are the management components (Table IV). In some aspects these components—often overlooked in cloud analytics or cyber security because they are programming vice network tools—become our core for cyber security in a distributed cloud architecture. They are perhaps our best available means for comprehensive enterprise analysis. However, in order to fully realize the benefits of this approach, one must be narrowly focused to Layer 7 issues—and within the OSI application tier specifically on processes and services within applications. For perspective, the issue stems from coding issues vice overall technical operations. Do you view the cloud as a physical data center to be programmed and secured via physical processes? If so then you are concerned with traditional programming and perimeter defense. Or, do you see the cloud as a web of virtual services with content discovered and processed as needed. In this instance your concern is with the overall flow or orchestration of the services and the governance processes to oversee the service and process flows and resultant security. At this level, cyber security exponentially deepens. Here we are far deeper than traditional security measures, down to the code. Herein lies modern cyber operations. While the management layer in our analytics grid does not solve anything, it is a means to understand what is really happening in our distributed cloud network and therein a means to ensure performance of the enterprise, to secure the services that transit our cloud nodes, and to secure the processes that run our services.

TABLE IV. CYBER MANAGEMENT.

| # | Service | Variables / Attributes |
|---|---|---|
| 9 | Orchestrate | ***Software logic:*** 1. *Process / procedure monitors* 2. *Route logic and class paths* 3. *Service workflows* |
| 10 | Governance | ***Service logic:*** 1. *Agreements* 2. *Policy enforcement* 3. *Component management* |
| 11 | Cyber | ***Service security:*** 1. *Authentication* 2. *Authorization* 3. *Encryption* |

***Orchestration*** supports event coordination and therein helps manage services, typically providing interfaces such as Business Process Management (BPM) and tools to build and control processes and web service interactions, often through Business Process Execution Language (BPEL). Metrics are needed for BPEL interface functions, web service policy enforcement, and for BPEL throughput and latency to include remote procedure calls.

***Governance*** systems support SOA management in an enterprise architecture. Applications and services collect data and metrics over time intervals—although systems can assist with real-time management. Agents can be deployed to help provide service metrics and to support governance actions such as policy enforcement, service agreements, and runtime procedures. Metrics could include virtual machine or container operations and messaging linkages. Connections to Business Activity Monitoring (BAM) systems can occur at the governance layer. Potentially, output from Governance software can appear on BAM dashboards, in which case dashboard variables would assess compatibility with analytic and decision support software.

***Cyber Security*** can be addressed at the service or process level as a function of standards enforcement, such as Web Services Policy (WS-Policy) to define conditions under which a service is to be provided. Metrics based on the performance of the service can be monitored for deviations from expected patterns. WS-Policy metrics can address data processes within composite services to conceptually provide security assessment for distribution publication/subscription services. SLAs can define what service providers have agreed to publish and consumers have agreed to accept. When integrated into governance systems, we can generate profiles of who the users are and what they are doing with the data. Agents can trigger alerts when SLA QoS specifications are not achieved or are altered. Metrics may include: success rate (success ratio/failure ratio), message count, error count, failover/retry count, validation error count, WSS error count, minimum response time and maximum response time.

## 6 Application Analytics

Now that we have established a conceptual basis and some physical possibilities for a cyber-analytics grid based on practical experience, the next step is to look briefly at tooling which might support the concept of an analytics grid for distributed cloud service and security assessment. APM-based solutions seem to hold the most promise; however, there is variation in the capabilities and implementation.

Generally, network vendors are moving their products "up the stack" to add application insight to current network management suites [28]. APM big data analytics tools are being advanced that can correlate thousands of metrics to identify patterns from real-time monitoring to provide topology impact assessment, application performance testing, end-user experience monitoring, transaction and SLA assessment, application dependency mapping, automated network modeling, service modeling, root cause problem analysis, cyber security alerts, and fine-grain event monitoring with real-time predictive analytics [29].

Software-defined data centers [clouds] and networks have changed requirements for end-to-end application analysis, necessitating that monitors be non-invasive, able to persist and characterize data as it traverse through real and virtual servers and networks and into different types of end-user device, and provide context correlation from real-time packet analysis [30].

Purpose-built devices to support an analytics grid and capable of evaluating not only traditional network communications but also providing visibility into layer 7 traffic and cloud virtual machine services are steadily evolving. The "fabric" contains tools able to provide pervasive visibility across physical, virtual and software-defined networks (SDN) with appliances at cloud nodes to filter, replicate and aggregate flow data to a centralized monitoring station [31].

A consideration in deep analytics is to address critical variables within the code test and quality assurance process. Approached from this angle and we have a basis for not only understanding that something has occurred but a potential means for a "deep dive" to examine the code [32]. Such an approach would integrate quality assurance within the software development and deployment life cycle to provide continuous analytics within an agile software process. While this concept offers potential, technical and bandwidth challenges for such a solution in a distributed, operational cloud network are not fully available today.

The author believes the previously mentioned tools, together with the out-of-band analytics grid concept, and analysis of variables discussed throughout this paper, offers a next-generation possibility for comprehensive, distributed cloud service analytics.

## 7 Conclusion

An analytics grid for real-time assessment of a heterogeneous, distributed cloud network will require a number of tools and capabilities. Additionally the network will need to support extremely fine-grain analysis, providing not only visibility but code access. While not available today in an integrated package the "building blocks" are available and each addresses a required analytic task. This paper has presented the some of the required capabilities for a cloud analytics grid, some of the variables to be addressed, and some of the tools that might evolve into a comprehensive, integrated suite required for a cloud analytics grid. Subsequent research can address additional variables and evolve the concept as new tools increasingly address these analytic requirements.

## 8 Acknowledgment

The author wishes to acknowledge OPNAV for support of this research.

## 9 References

[1] Zheng L., O'Brien, L., Zhang, H., and Cai, R. "On a Catalogue of Metrics for Evaluating Commercial Cloud Services", Proceedings of the 2012 ACM/IEEE International Conference on Grid Computing (GRID), 2012, pp. 164-173.

[2] Zhao, L., Liu, A., and Keung, J. "Evaluating Cloud Platform Architecture with the CARE Framework", 2010 17th Asia Pacific Software Engineering Conference (APSEC), 2010, pp. 60-69.

[3] Guan, Q., Chiu, C., and Fu, S. "CDA: A Cloud Dependability Analysis Framework for Characterizing System Dependability in Cloud Computing Infrastructures", 2012 IEEE 18th Pacific Rim International Symposium on Dependable Computing (PRDC), 2012, pp. 11-20.

[4] Gao, J., Pattabhiraman, P., Bai, X., and Tsai, W. "SaaS Performance and Scalability Evaluation in Clouds", 2011 IEEE 6th International Symposium on Service Oriented System Engineering (SOSE), 2011, pp. 61-71.

[5] Kowall, J., Cappelli, W. "Magic Quadrant for Application Performance Monitoring". Stamford, CT: Gartner, 2013. Available: http://www.gartner.com/technology/reprints.do?id=1-1ODYDEA&ct=131219&st=sb

[6] Maule, R., and Lewis, W. "Performance and QoS in Service-Based Systems", Proceedings of the IEEE 2011 World Congress on Services Computing (SERVICES 2011), 4-9 July, Washington, DC, 2011.

[7] Kalagiakos, P., and Bora, M. "Cloud Security Tactics: Virtualization and the VMM", 2012 6th International Conference on Application of Information and Communication Technologies (AICT), 2012, pp. 1-6.

[8] Flood, J., and Keane, A. "A Proposed Framework For The Active Detection Of Security Vulnerabilities In Multi-Tenancy Cloud Systems", 2012 Third International Conference on Emerging Intelligent Data and Web Technologies, 2012, pp. 231-235.

[9] Bratus, S., Darley, T., Locasto, M., Patterson, M., Shapiro, R., and Shubina, A. "Beyond Planted Bugs in "Trusting Trust": The Input-Processing Frontier". IEEE Security & Privacy, January/February 2014, pp. 83-87.

[10] Hiray, S., and Ingle, R. "Context-Aware Middleware in Cyber Physical Cloud", 2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies, 2013, pp. 42 – 47.

[11] Yassin, W., Udzir, N., Muda, Z., Abdullah, A., and Abdullah, M. "A Cloud-Based Intrusion Detection Service Framework", 2012 International Conference on Cyber Security, Cyber Warfare and Digittal Forensic (CyberSec), 2012, pp. 213-218.

[12] Ficco, M., Tasquier, L., and Aversa, R. "Intrusion Detection in Cloud Computing", 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 2013, pp. 276-283.

[13] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall, New York, 2005.

[14] Parsons, T., Mos, A., Trofin, M., Gschwind, T., and Murphy, J. "Extracting Interactions in Component-Based Systems", IEEE Transactions on Software Engineering, Vol. 34, No. 6, 2008, pp. 783-799.

[15] Liu, G., Zhu, Z., Li, Y., Li, D., and Cui, J. "A New Web Service Model Based on QoS", International Symposium on Intelligent Ubiquitous Computing and Education, 2009, pp. 395-399.

[16] Luo, J., Li, Y., Pershing, J.; Xie, L., and Chen, Y. "A Methodology for Analyzing Availability Weak Points in SOA Deployment Frameworks", IEEE Transactions on Network and Service Management, Vol. 6, No. 1, 2009, pp. 31-44.

[17] Rajan, H., and Hosamani, M. "Tisa: Toward Trustworthy Services in a Service-Oriented Architecture", IEEE Transactions on Services Computing, Vol. 1, No. 4, 2008, pp. 201-213.

[18] Lee, Y. "Event-driven SOA Test Framework Based on BPA-Simulation," First International Conference on Networked Digital Technologies, 2009, pp. 189-194.

[19] Her, J., Choi, S., Oh, S., and Kim, S. "A Framework for Measuring Performance in Service-Oriented Architecture", Third International Conference on Next Generation Web Services Practices (NWeSP 2007), 2007, pp. 55-60.

[20] Shim, B., Choue, S., Kim, S., and Park, S. "A Design Quality Model for Service-Oriented Architecture", 15th Asia-Pacific Software Engineering Conference (APSEC 2008), 2008, pp. 403-410.

[21] Xiao-jun, W. "Metrics for Evaluating Coupling and Service Granularity in Service Oriented Architecture", International Conference on Information Engineering and Computer Science (ICIECS 2009), 2009, pp. 1-4.

[22] Gao, J., Wu, Y., Chang, L., and Meldal, S. "Measuring Component-Based Systems using a Systematic Approach and Environment", Second IEEE International Workshop on Service-Oriented System Engineering (SOSE 2006), 2006, pp. 121-129.

[23] Kumari, G., Kandan, B., and Mishra, A. "Experience Sharing on SOA Based Heterogeneous Systems Integration", 2008 IEEE Congress on Services, Honolulu, HI, 7-11 July 2008.

[24] Hau, T., Ebert, N., Hochstein, A., and Brenner, W. "Where to Start with SOA: Criteria for Selecting SOA Projects", Proceedings of the 41st Hawaii International Conference on System Sciences, Waikoloa, HI, 7-10 January 2008.

[25] Roach, T., Low, G., D'Ambra, J. "CAPSICUM–A Conceptual Model for Service Oriented Architecture", 2008 IEEE Congress on Services, Honolulu, HI, 7-11 July 2008.

[26] Choi, S., Her, J., and Kim, S. "Modeling QoS Attributes and Metrics for Evaluating Services in SOA Considering Consumers' Perspective as the First Class Requirement", IEEE 2nd Asia-Pacific Services Computing Conference, Tsubuka Science City, Japan, 11-14 December 2007.

[27] R. Maule, "Quality of Service Assessment in SOA Synchronous Networked Communications", Proceedings of the 2007 International Conference on Computing, Communications and Control Technologies (CCCT 2007), Orlando, FL, 12-15 July 2007.

[28] Craig, J. Application Performance Management (APM) in the Age of Hybrid Cloud: Ten Key Findings. Boulder, CO: Enterprise Management Associates, 2013.

[29] Azoff, M. Solution Guide: Application Performance Management (IT017-003964), 25 May 2012. Available: http://ovum.com

[30] Supasatit, T. ExtraHop IT Operational Intelligence Platform. Seattle, WA: ExtraHop Networks.Available: http://www.extrahop.com

[31] Gigamon. Pervasive Visibility for the Enterprise: Solutions Brief. Milpitas, CA: Gigamon, 2013. Available: http://www.gigamon.com

[32] TechTarget. A Guide to Agile Testing for QA and Test Managers. Available: http://www.techtarget.com

# Analysis of ICmetrics features requirements in Cloud environment

Bin Ye
School of Engineering and Digital Arts
University of Kent
Canterbury, UK
by30@kent.ac.uk

M.Haciosman, Gareth Howells
School of Engineering and Digital Arts
University of Kent
Canterbury, UK
mh521@kent.ac.uk , W.G.J.Howells@kent.ac.uk

***Abstract—As web-server spoofing is increasing,* we investigate a novel technology *termed* ICmetrics, *used* to identify fraud for given web servers based on measurable quantities/features. The novel concept ICmetrics is used to detect spoof websites with the advantages of a higher level of security with increased speed and template free encryption. ICmetrics technology is based on extracting features from digital systems' operation that may be integrated together to generate unique identifiers for each of the systems or create unique profiles that describe the systems' actual behavior. Ideally, the nature of the features should be identical for all of the systems considered, while the values of these features should allow for unique identification of each of the system servers. This paper looks at the properties of the several behaviors as a potential ICmetrics features, and explores properties which affects the stability of the system's performance. We conclude three requirements for ICmetrics system.**

***Keywords—security, ICmetrics, encryption, Cloud computing, biometrics***

## I. Introduction

In recent years, There are two problems arises due to internet security issues which has caused as increasing number of spoof websites, E-mail scams, fake application servers or some other format of fraudulent information [1]. Firstly, they all exhibit one thing in common that they guide you to a new link which pretends as one of your familiar web servers. For instance, you click a link on a page or in an email you have received. The email is sent from the bank; it has banks' logo and consists of their usual style. The clickable link redirects you to a page with the usual account login fields for you to enter your username and password. You type in your username and password but for some reason it doesn't log in. Everything is as it should be. The problem is that are you certain that the site you are looking at is what it appears to be? Unfortunately, it is very possible that you have just become a victim of a crime involving a "spoofed" website and the contents of all your bank accounts are now at risk. Although current bank systems already provided encryption systems, encryption cannot necessarily protect against fraudulent data manipulation where the security of encryption keys cannot be absolutely guaranteed. This encounters a second problem that current encryption techniques all expose a weakness that they all have to store an encryption template [2]. If the template is stolen, then, the entire system is under risk. Conventional encryption systems such as biometrics pose a similar problem; they have to store their template for key generation very carefully. Would feel safe with this kind of system?

The system in this paper present a novel technology called ICmetrics which would provide template free encryption, ease of provision, free from any form of malware, authentic and to allow use of service efficiently from diverse locations [3]. It is necessary in some sensitive area's such as the military and banks. The ICmetrics technology is developed at the University of Kent for deriving unique encryption keys based on the characteristics of hardware or software systems (or a combination of software and hardware configuration) [4]. Technically, it provides two advantages: (1). It removes the need to store any data directly containing the value of the encryption keys. (2). It requires all characteristics of features to generate encryption keys. It is based on extracting features from digital devices operations and software behaviors that may be integrated together to generate unique identifiers for each of the devices or software based services to create unique profiles that describe the systems actual behavior [5]. Any changes in these identifiers (profiles) during devices or systems operation would signal about a possible safety or security breaking within the system. ICmetrics is defined as a two-step process[6]:

- **Calibration phase:**
  1. For each sample device or system: measure the desired feature values.
  2. Generate feature distributions describing the frequency of occurrence of discrete value for each sample system.
  3. Normalize the feature distributions and generate normalization maps for each feature.
- **Operation phase:**
  1. Measure desired systems' features.
  2. Apply the normalization maps to generate values suitable for key generation.
  3. Apply the key generation algorithm.

However, in our previous work [1, 3], the target space was linear in nature. We used enhanced Peak-Trough detection[4, 5], and kernel estimation algorithms [10] to determine the various modal clusters taking one feature at a time. Our current research, however, is focused on investigation of multi-dimensional spaces combining various features where each system mode is equal-distant from every other. This would allow the system to be applied to Cloud servers which have not formed part of the calibration sample within any enrolment of known samples from the target servers. Such a generalization provides an improved mapping onto the key generation space and allows the multi-modal nature of the feature distributions to be effectively integrated within the overall system [11]. Considering multiple features that are different in nature has also another advantage of designing hybrid ICmetrics systems that can include features derived not only from one system, but also from different systems from a same Cloud server provider. Such an approach is particularly useful for autonomous and intelligent Cloud computing environments where Cloud server customers and Cloud server companies frequently use and interact with Cloud infrastructure. For example, data transmitting to and from a web server which it hosts on Cloud infrastructure can be encrypted using features extracted not only from servers' characteristics but also from the signal generated by the users' behaviors. We investigated several software behaviors as potential ICmetrics features and evaluate if it could be used to determine a device uniquely in a multi-dimensional feature space.

## II. Feature acquisition

Building an experimental platform for extracting ICmetrics features involves several stages: (1) designing the hardware-software test-bench; (2) programming simulations of systems' operation; (3) developing tracing methods for data acquisition; (4) recording feature values for their further analysis as required by ICmetrics research. The following subsections describe implementation of these stages in turn.

### A. Software test-bench

The server consist 3 desktops and it is managed by Eucalyptus. Three Xen virtual machines are running on top of the cluster. For this research we use LTTng 2.x [12] as a profiling tool to extract desired feature values. The list of employed software is listed below:

- Dell optiplex 745 core2 E6600 2400(memory: 2.0G; CPU 2.4G $\times$2) $\times$3.
- Apache 2.2, Eucalyptus 3.4.0, Xen 3.3.
- Linux OS-Ubuntu 10.04LTS. Server applications $\times$3.
- LTTng 2.x (tracing tools for Linux, use to profile server.)

### B. Server operations

Service operations in the real world can be very complex, large and consume many resources, including other services. It would not be practical to develop these types of services in the provided time scale. Therefore, it was decided that some simple, dummy services should be developed with functionalities that could occur frequently in practice. Functionalities such as interfacing with a database and data processing are quite common for web services; so basic examples have been implemented. Server 1 contains an ant colony algorithm. It solves a shortest path problem. Server 2 employs three sorting algorithms (bubble, insertion, shell and quick sort) to each sort a list of random numbers of a given length. Whereas the other methods did not return a result, this method returns the sorted lists. Each sorting algorithm runs in its own thread. Three of the algorithms are highly iterative with the quick sort being highly recursive, which will significantly affect the feature vectors obtained. Server 3 runs a map-reduce program based on Hadoop which include data exchange between two databases.

### C. Feature extracting

The most important aspect of the system is the feature selection and their subsequent extraction. It is important to choose features that have minimal variation between executions of the same service operation regardless of the input arguments and, equally as important, the platform on which the services are hosted (hence timings and memory locations are not considered). This is termed intra-sample variation. Furthermore, the features of one operation should exhibit a significant degree of difference from features extracted from that of another, which is called inter-sample variation. These two characteristics will provide the best separation between classes and result in optimal performance[13]. Cloud computing is a kind of computing architecture where it's hardware is hidden under the operating system and we cannot guarantee which machine the program is executing. So, hardware features such as performance counters and program counter [14] are not useful in this research. The features such virtual heap space and method invocations are under consideration. In this paper, we investigated Linux kernel function invocation as a potential ICmetrics features. The features were collected based on accumulation of every 5 seconds. Each feature was collected 1000 times. Totally, there are 17 features were collected.

## III. Feature reguirements analysis

The generation of encryption keys requires developing suitable methods for combining selected features so as to produce a unique basis number [15] – an initial number unique to the Cloud server from which actual encryption keys may be derived. The main requirement for such a method is that they should allow for generating basis numbers with low intra-sample variance (the values produced for the same device) but high inter-sample variance (the values produced for different devices) with the ideal case being no inter-sample overlap of potential basis numbers [16]. In our earlier work [4, 16], we have investigated two alternative techniques for combining features, namely, feature addition and concatenation [18]. Due the nature of Cloud computing, our previous works are no longer suitable. Because one Cloud computing cluster may contain a number of similar servers. They are independent but all explore similar servers' behaviors, so we have to extract more features. In this situation, we decide to use multi-dimensional feature space contains all kind features together. Following equation is used:

$$f(x) = \frac{1}{\sqrt{(2\pi)^k|\varepsilon|}} exp^{(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}$$

Each dimension contains one or more features and they are all independent. Through the equation all we want is the vector of the expectations and covariance matrix. The procedures of the servers' identification are: 1) A detector randomly measures a number of features. 2) It then maps vlues to the normalization map to generate a unique basis number (use to generate key). To make such a system to be available, feature data should satisfy following requirements:

## A. *Correlation of features*

To generate the normalization map, we need to map feature values to a key generation vector, but actual feature values are very complex and overlapped. Figure 1 is a sample feature distribution. We use 'S' to represent server and 'F' to represent feature. So, S1F1 means the feature 1 of Server1. Through the Figure we can see that feature distribution is very complex and overlap. So, it is very difficult to generate normalization maps based on the raw data.

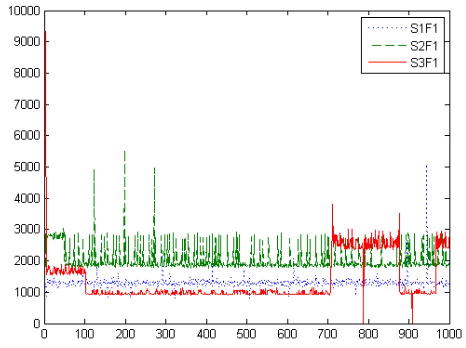


**Figure 1-Feature 1's distribution of three servers**

Correlated features reduce the entropy of the system because knowing the value of some lets you guess the values of others. We therefore need to treat them as an integrated unit to maximize the entropy of the system. Then, Pearson correlation coefficients are used as a new feature. For instance, table 1 shows coefficients of the same features combinations from different servers. The coefficient of F1-F12 from server1 is 0.000789 and the coefficient of S2 is 0.01169. This shows a great difference between S1 and S2. Although the coefficient of S3 is 0.0219, which it shows a small difference compared to S2, but it still distinguishable. For F10-F14, S1 and S2 show similarity. S3 shows enormous disparity between S1 and S2. In this case, S3 is distinguishable, but S1 and S2 are not separable. In this situation, we can still distinguish them according to the Pearson correlation distribution.

**Table 1-correlation of feature combination of three servers**

| | F1-F12 | F6-F14 | F10-F14 | F16-F17 |
|---|---|---|---|---|
| S1 | 0.000789 | -0.0611 | 0.974 | 0.735 |
| S2 | 0.01169 | 0.01137 | 0.969 | 0.0809 |
| S3 | 0.0219 | 0.6533 | 0.0759 | 0.1763 |

For example, Figure 2 is a correlation distribution diagram of feature 3 and feature 9. As we can see, the blue bubbles, which represent server 3 and it has no overlap from server 1 and server 2. Server 2 and server 1 overlapped a little. According this graph, server 3 is perfectly distinguishable as server 3 has no overlap in the Pearson correlation distribution. If features overlap, then, we can switch to another dimension. As the nature of multi-dimension space, each dimension is independent, which allows us to check every dimension randomly. If they all overlapped, then, we import a Posterior Probability system to make a decision based on the statistically reliable function.

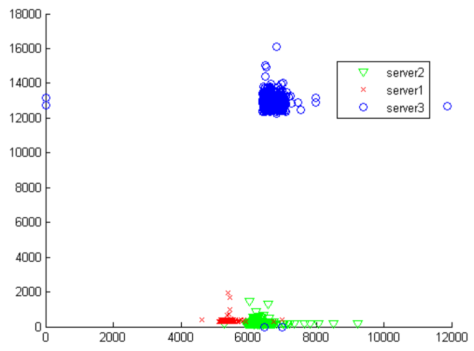


**Figure 2-Pearson correlation distribution of three servers**

## B. *intra and outra sample variance*

It is important to choose features that have minimal variation between executions of the same service operation regardless of the input arguments and, equally as important, the platform on which the services are hosted. This is termed intra-sample variation. Furthermore, the features of one operation should exhibit a significant degree of difference from features extracted from that of another, which is called inter-sample variation. These two characteristics will provide the best separation between classes and result in optimal performance.

## C. *Multi-level mapping*

As some feature distributions (Figure 3) are showed unusual and incorporating these features is difficult especially in a multi-dimensional space. One of the overriding criteria required from the system is to allow all features to be combined together to form a single encryption key. Encompassing all forms of feature distribution within a single over-arching model thus becomes desirable. In order to solve that, we introduced a multi-level mapping [7] system to generate a single regular normal distribution. Basically, the multi-level mapping system will map feature values into a new regular coordinate to make a new regular distribution. For some random distribution, we need to see real ones to decide how to integrate them but the general idea is to have a multi-level mapping, mapping them initially into a parameterized distribution.

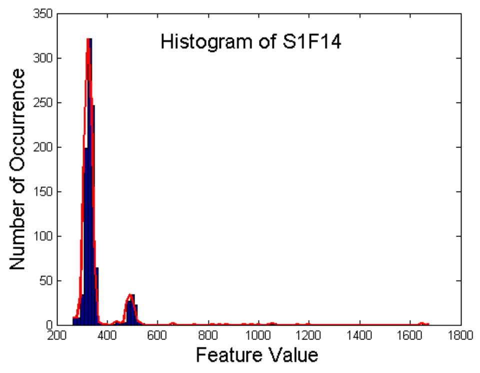


**Figure 3-Feature S1F14 shows bimodal distribution**

## *D. Space distance*

Feature distance describes distribution of the features in a multi-dimensional space. This concept is used to analyze how the feature is located in the space. For instance, in Figure 2, if we observer the graph from left, then, server 3 is distinguishable. If we look from the bottom, then, server 2 is distinguishable. If we put a new feature in a new dimension and drag the server 3 to a different location, then, they are all distinguishable. This method is used to find best feature combinations of correlated data. To calculate the feature distance, we start by selecting a random feature from the group of correlate features. Then, we use Euclidean distance to calculate the distance of each other. After that, we pick another feature and calculate Euclidean distance with previous one together. If the distance increased, then, we remove the feature. After that, we iterate above procedures until we find a best combination of features. To calculate the distance, we previously find the center points of each group of features, but due to the anomaly of data, the center points cannot represent a real distance. So, we calculate the distance based on closest points between each group of data. It is difficult select feature when data overlapped in the multidimensional space. Currently, we ignored the overlapped data but pick up the closest one without the overlapped data. This method is used to select best feature combinations that can show greatest differences between each other.

## *E. Feature normalization and quantization*

The proposed system works in the phase process, firstly analysing typical feature values for Cloud servers to produce a normalization map for the feature and subsequently employing the normalization maps to produce a code for a potentially unknown Cloud server. A conventional simple strategy for generating an encryption key from a given feature distribution may involve quantising the distribution into fixed subsets with each value within a given subset mapping to a single value. To generate proper quantisation intervals we undertook the following tests. The goal of quantisation is to normalise feature data, so the best quantisation interval should exhibit the biggest inter sample variance between cloud servers. Figure 4 represents the number of bins versus variance of the features. Bins represent the number of intervals employed. A high number means more segmentation between feature value ranges. As can be seen from Figure 4, when the number of bins reaches around 100, the variance stops increasing. Each feature will potentially have a different number of bins. In our multi-dimensional feature space, they are independent, so, each feature could have different quantisation strategy.

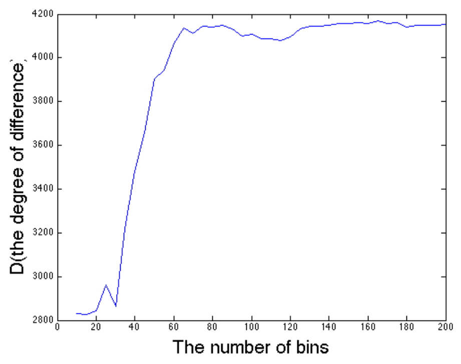


**Figure 4-Number of bins versus variance of feature values of feature 1 between servers1 and server2**

## IV. CONCLUSION

This paper provides a new technology that can be used to encrypt components of services located within the Cloud using properties or features derived from their own construction and behavior to form a digital signature capable of assuring both their authenticity and freedom from malware whilst simultaneously allowing the flexibility for it to operate within their designed specification and execute on an arbitrary platform. The properties of ICmetrics features in Cloud environment have been explored and we listed the following 3 requirements for ICmetrics to be available in a Cloud environment. Firstly, the data should correlate to each other because correlated features improve the robustness of the system and raise the feasibility of raw feature data. Next, the combination of the data need to present a certain amount of discrimination in a multi-dimensional space, which means it should as less overlap as much in the multi-dimensional space. Then, the data should show high ultra-sample variance and low intra-sample variance as high ultra-sample variance can significantly improve the performance of the system. The Euclidean distance is used to detect best feature combinations that show greatest differences between each other. Finally, we evaluated normalization and quantization of the feature values. Overall, this paper outlines the methodologies of analysis and mathematical implementation. At this step, we finished the data analysis and future work will focus on solutions of implementation.

## V. ACKNOWLEDGEMENT

The authors gratefully acknowledge the support of the UK Engineering and Physical Sciences Research Council under grant EP/K004638/1 and the EU Interreg IV A 2 Mers Seas Zeeën Cross-border Cooperation Programme – SYSIASS project: Autonomous and Intelligent Healthcare System (project's website http://www.sysiass.eu/).

## REFERENCE

[1] D. N. Use, “Global Phishing Survey : Trends and Domain Name Use in 2H2012 July-December 2012 G l o b al R es p o n s e Industry Advisory,” no. April, pp. 1–30, 2013.

[2] M. Fatindez-Zanuy, “On the vulnerability of biometric security systems,” *IEEE Aerosp. Electron. Syst. Mag. (June 2004)*, pp. 3–8, 2004.

[3] R. Tahir and K. McDonald-Maier, “Improving Resilience against Node Capture Attacks in Wireless Sensor Networks using ICMetrics,” in *Emerging Security Technologies (EST), 2012 Third International Conference on*, 2012, pp. 127–130.

[4] E. Papoutsis, G. Howells, a. Hopkins, and K. McDonald-Maier, “Key Generation for Secure Inter-satellite Communication,” *Second NASA/ESA Conf. Adapt. Hardw. Syst. (AHS 2007)*, pp. 671–681, Aug. 2007.

[5] B. Ye, G. Howells, and M. Haciosman, “Investigation of Properties of ICmetric in Cloud,” in *Emerging Security Technologies (EST), 2013 Fourth International Conference on*, 2013, pp. 107–108.

[6] R. Tahir, H. Hu, D. Gu, K. McDonald-Maier, and G. Howells, “Resilience against brute force and rainbow table attacks using strong ICMetrics session key pairs,” in *Communications, Signal Processing, and their Applications (ICCSPA), 2013 1st International Conference on*, 2013, pp. 1–6.

[7] G. Howells, E. Papoutsis, A. Hopkins, and K. McDonald-Maier, “Normalizing Discrete Circuit Features with Statistically Independent values for incorporation within a highly Secure Encryption System,” in *Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on*, 2007, pp. 97–102.

[8] X. Zhai, K. Appiah, S. Ehsan, W. M. Cheung, G. Howells, H. Hu, D. Gu, and K. McDonald-Maier, “Detecting Compromised Programs for Embedded System Applications,” in *Architecture of Computing Systems--ARCS 2014*, Springer, 2014, pp. 221–232.

[9] X. Zhai, K. Appiah, S. Ehsan, H. Hu, D. Gu, K. McDonald-Maier, W. M. Cheung, and G. Howells, “Application of ICmetrics for Embedded System Security,” in *Emerging Security Technologies (EST), 2013 Fourth International Conference on*, 2013, pp. 89–92.

[10] Y. Kovalchuk, W. G. J. Howells, H. Hu, D. Gu, and K. D. McDonald-Maier, “A practical proposal for ensuring the provenance of hardware devices and their safe operation,” 2012.

[11] R. Tahir and K. McDonald-Maier, “An ICMetrics based Lightweight Security Architecture using Lattice Signcryption,” in *Emerging Security Technologies (EST), 2012 Third International Conference on*, 2012, pp. 135–140.

[12] M. Desnoyers and M. R. Dagenais, “The LTTng tracer: A low impact performance and behavior monitor for GNU/Linux,” in *OLS (Ottawa Linux Symposium)*, 2006, vol. 2006, pp. 209–224.

[13] Y. Kovalchuk, H. Hu, D. Gu, K. McDonald-Maier, D. Newman, S. Kelly, and G. Howells, “Investigation of Properties of ICmetrics Features,” in *Emerging Security Technologies (EST), 2012 Third International Conference on*, 2012, pp. 115–120.

[14] K. Appiah, X. Zhai, S. Ehsan, W. M. Cheung, H. Hu, D. Gu, K. McDonald-Maier, and G. Howells, “Program Counter as an Integrated Circuit Metrics for Secured Program Identification,” in *Emerging Security Technologies (EST), 2013 Fourth International Conference on*, 2013, pp. 98–101.

[15] A. Hopkins, K. Mcdonald-Maier, and G. Howells, “Device to generate a machine specific identification key.” Google Patents, 2013.

[16] Y. Kovalchuk, K. McDonald-Maier, and G. Howells, “Overview of ICmetrics Technology-Security Infrastructure for Autonomous and Intelligent Healthcare System.,” *Int. J. U-& E-Service, Sci. Technol.*, vol. 4, no. 3, 2011.

[17] X. Zhai, K. Appiah, S. Ehsan, W. M. Cheung, H. Hu, D. Gu, K. McDonald-Maier, and G. Howells, “A Self-Organising Map Based Algorithm for Analysis of ICmetrics Features,” in *Emerging Security Technologies (EST), 2013 Fourth International Conference on*, 2013, pp. 93–97.

[18] R. Tahir, H. Hu, D. Gu, K. McDonald-Maier, and G. Howells, “A scheme for the generation of strong cryptographic key pairs based on ICMetrics,” in *Internet Technology And Secured Transactions, 2012 International Conferece For*, 2012, pp. 168–174.

# Cloud Computing Benchmarking: *A Survey*

**C.Vazquez, R. Krishnan, and E. John**
Department of Electrical and Computer Engineering, The University of Texas at San Antonio,
San Antonio, Texas, U.S.A

**Abstract –** *Cloud computing gives service-oriented access to computing, storage and networking resource. Often, these resources are virtualized. The prospect of being able to scale computing resources to meet user demand has clearly caught the attention of developers and organizational IT leaders over the recent years. Considering the number of cloud computing providers and the different services each provider offers, cloud users need benchmark information that specifically addresses the unique properties of the cloud computing environment such as dynamic scaling. This paper compares five prominent tools (CloudCmp, CloudStone, HiBench, YCSB, and CloudSuite) that present workloads and/or methods for quantitatively comparing cloud computing offerings.*

**Keywords:** Cloud computing, Workload, Benchmarking, Performance evaluation

## 1 Introduction

The increase in popularity of cloud computing in recent years is driven by the advantages offered by the dynamically scalable, pay-as-you-go model. This enables organizations to focus on providing services to their customers while consuming the requisite computing resources as a utility. By eliminating the need for on-premises equipment, organizations avoid large capital expenses and instead focus resources towards faster deployment. The pay-as-you-go model allows an organization to grow naturally with customer demand. Since cloud computing resources scale elastically, utilizing cloud computing reduces the risk of over provisioning, wasting resources during non-peak hours, and reduce the risk of under provisioning, missing potential customers [32]. Success stories of start-ups like Instagram, which built-up a user base of over 150 million users in less than four years using only public cloud solutions [38], exemplify the potential for fast growth that utilizing cloud computing can provide.

Considering the number of cloud computing providers and the different services each provider offers, a customer shopping for an appropriate solution for their organization requires benchmark information that specifically addresses the unique properties of the cloud computing environment. A benchmark must provide an accurate representation of the workload the consumer intends on running. A benchmark targeting social networking sites should differ from a benchmark targeting database systems. Different applications running on the same computing platform can have different requirements in terms of computing, storage, and networking, and modern web applications can have wide disparities between peek and average demand [32]. A developer must ensure that the cloud provider's services can scale to meet their end-users' demand. Long response times from a cloud application can lead to limited adoption of an application since there are often competitors offering similar products.

Although standard methods for reporting the performance of cloud resources are still not available, tools have been suggested to give the consumer the ability to quantitatively compare the offerings of cloud providers. This paper identifies five such tools: CloudCmp [1], CloudStone [2], HiBench [3], YCSB [4], and CloudSuite [5].

## 2 Background

### 2.1 Cloud Computing

Cloud computing is a large-scale, distributed computing paradigm which is driven by economies of scale. Providers of cloud computing offer abstracted, virtualized, dynamically scalable, and managed resources on demand to external customers over the Internet [33]. These resources include compute, storage and networking. Cloud computing providers benefit from economies of scale in that they assemble massive datacenters operating tens of thousands of servers which service a wide customer base. Large-scale operation more effectively absorbs operational costs through the benefits of increasing the utilization of equipment, bulk discounts on purchased equipment, and reducing the cost of cooling and powering equipment [6]. The demand for large-scale computing resources continues to grow as Internet users generate larger sets of data to be processed.

The essential characteristics of cloud computing [7] are:

- On-demand self-service – The ability to provide computing capabilities as needed automatically, when needed.
- Broad networks access – Cloud services are available over the network and accessed through standard mechanisms.
- Resource pooling – Physical and virtual resources are dynamically assigned to serve multiple consumers using a multi-tenant model.
- Rapid elasticity – Capabilities are elastically provisioned and released quickly without perceived bound.
- Measured service – Cloud services automatically control resource use by leveraging appropriate metering capability (pay-per-use).

## 2.2 Virtualization

Virtualization is a fundamental component of cloud computing, allowing for pooling and dynamically allocating hardware resources. A server in a datacenter acting as a host machine is installed with a hypervisor which can simultaneously run instances of virtual machines or guest machines. These virtual machines are operating system instances managed by a separate controlling computer which loads them into respective host machines. With the controlling computer managing the computing resources of many servers, a cloud computing provider thus unifies the datacenter's resources into an encapsulated pool which can be allocated and released according to user demand.

## 2.3 Services

The NIST definition of cloud computing [7] categorizes the services that providers offer into three service models: infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS), or a software-as-a-service (SaaS).

- An IaaS provides access to instances of unified resources including computing, storage, and networking. Providers offer flexible computing resources for a usage-based price. These resources are distributed as instances on demand which are treated like physical hardware. The user is left with the responsibility for demanding and initializing new instances when scaling is required.
- A PaaS provides many of the same resources as an IaaS but through an integrated environment which reduces the development burden of using the resources but also restricts features. PaaS providers offer a variety of computing and storage resources in a more constrained environment that can be accessed through APIs. Many application specific tools are pre-built and available to users such as web hosting, data management, business analytics, etc.
- SaaS, such as e-mail and Google Docs, are special-purpose software services which are used remotely by the end user. They are often built using PaaS and IaaS tools, but their implementation details are hidden from the end-user.

## 2.4 MapReduce

Since cloud computing now offers wide horizontal scaling, end-users are taking the opportunity to process massive sets of data, a service which was previously only available to users with a dedicated datacenter. Apache Hadoop [8], an open-source version of Google's MapReduce [9] and GFS [10], is a parallel processing framework used for many cloud-based batch-processing projects. A data set in a file system or a database is processed as follows:

1. Initialize - A list of key-value pairs is distributed over the nodes in a cloud.
2. Map phase – Each node performs a specified operation on the key-value pairs to produce new key-value pairs.
3. Shuffle phase – The new data is rearranged on the nodes according to a partition function which groups data.
4. Sort phase – Each node assigns new key-value pairs.
5. Reduce phase – Key-value pairs are merged to a data-set.

# 3 Cloud Benchmarking Tools

## 3.1 CloudCmp

CloudCmp is a proposed framework designed to estimate the performance and cost of a legacy application running on a cloud without the expense or effort of porting and deploying the application. To achieve this goal, CloudCmp uses an approach composed of three phases: service benchmarking, application workload collection, and performance prediction.

In the service benchmarking phase the services of six cloud providers (including Google AppEngine [11], Amazon AWS [12], Microsoft Azure [13], GoGrid [14], and Rackspace [15]) are selected based on their ability to provide cloud computing services necessary for web application development on a cloud. These cloud computing services include access to an elastic compute cluster, persistent storage, intra-cloud networking, and wide-area delivery networking. Each cloud service's performance and cost are estimated by running a collection of benchmarking tasks designed to exercise each of the characteristics of cloud computing services.

- Elastic compute cluster efficiency– Different compute clusters were tested with SPECjvm2008 [16] Java tasks. Java tasks were selected because of Java's portability. The performance of each cluster was measured by the finishing time of each task while the cost effectiveness was measured by the cost per task.
- Elastic compute cluster scaling – Scaling was measured by the latency between the time an instance was requested and when the instance was ready. The applicability of this metric is limited by the fact that not all services allow for scaling via instance request.
- Persistent storage services – To test the performance of a persistent storage service the latency to insert or fetch a random to and from a data table was measured. The test was carried out with table sizes of 1000 entries and 100,000 entries. The results showed that the operation and table size had a significant effect on the performance.

TABLE I
COMPARISON OF CLOUD BENCHMARKING TOOLS

| | CloudCmp | CloudStone | HiBench | YCSB | CloudSuite |
|---|---|---|---|---|---|
| **Target** | Estimate the performance and costs of running a legacy application on a cloud | Capture "typical" Web 2.0 functionality in a cloud computing environment | Hadoop (MapReduce) programs including real-world applications | Performance comparisons of the new generation of cloud data serving systems | Characterize scale-out workloads |
| **Cost** | • Cost per task per instance type | • Cost per user per<br>• month | • Not covered | • Not covered | • Not covered |
| **Scaling** | • Latency to allocate new instance | • Load balancer – Apache default or user defined | • None specific | • Scaleup<br>• Elastic speedup | • None specific |
| **Storage** | • Latency to insert/fetch a random entry from pre-defined data table | • User's choice of relational database | • Aggregated bandwidth delivered by HDFS | • Adjust possible operations, data size, and distribution to target specific workloads | • Uses YCSB to assess serving systems |
| **Networking** | • Intra-cloud –TCP throughput between instances<br>• Wide-area delivery network – send ping packets from distributed locations | • None specific | • None specific | • None specific | • None specific |
| **Computing performance** | • Latency of various SPECjvm2008 tasks | • Response time of request made by load generator | • Speed – job running time<br>• Throughput – tasks completed per minute<br>• System resources utilization | • Read/Update Latency | • Execution cycle profile<br>• Instruction cache miss rate<br>• IPC/MLP<br>• Memory bandwidth utilization |
| **Test environment** | • Multiple instance types | • Amazon EC2 instances | • Hadoop cluster | • Data serving system | • Server |
| **Service** | • IaaS<br>• PaaS | • IaaS | • PaaS | • PaaS | • IaaS |
| **Workload** | • User-defined application's request traces and each request's execution path | • Olio driven by Faban | • Sort<br>• WordCount<br>• TeraSort<br>• Web search<br>• Machine learning<br>• File system | • Random operations on random data based on selected distributions | • Data serving<br>• MapReduce<br>• Media Streaming<br>• SAT Solver<br>• Web hosting<br>• Web search |

- Intra-cloud network – The available bandwidth between two instances in the cloud was tested by measuring the average TCP throughput of instances in the cloud using the iperf [17] tool for many pairs. This test is limited only to cloud providers which allow explicit intra-cloud communication.
- Wide-area delivery network – The latency of a cloud provider's delivery network was measured by sending ping packets from different geographic locations.

The goal of the application workload collection phase is to obtain a workload representation of a user's legacy application. It is proposed that this can be achieved by collecting the application's request traces and deriving an execution path for each request. In the performance prediction phase, the profiles of each cloud service and the workload representation of the legacy application would be used to estimate the total running time and total cost of running the application.

## 3.2 CloudStone

CloudStone is a toolkit for characterizing the workload of a typical social networking website. The goal of CloudStone is to give developers tools to investigate different implementation decisions which affect the performance and price of running a social networking website. These tools can currently only be utilized on a cloud service which can use Amazon EC2 instances. The three components of CloudStone are: Olio, automation tools for running Olio experiments, and a methodology for computing a suggested metric.

Olio features two complete implementations of a social-event calendar application and utilizes a time-varying workload generator, Faban [18]. The two application implementations, in both PHP and Ruby-on-Rails, provide an identical user experience allowing for a direct comparison of each development stack. Faban simulates multiple users simultaneously by running parallel agents on different which are controlled by one central coordinator. The central coordinator can also change the number of active users during a run. Faban also collects the latency of each request and utilization data.

Performing an experiment with CloudStone involves selecting a configuration for the Olio deployment, selecting a workload profile to be generated by Faban, and deploying the instances. The performance of the configuration of Olio will differ depending on the different tuning mechanism each implementation provides such as database caching, load balancer, etc. The results of the experiment are suggested to be expressed in terms of a metric of dollars per user per month.

## 3.3 HiBench

HiBench is a benchmark suite targeting the components of the Hadoop framework. The use of many realistic workloads fully exercises Hadoop's parallel computing component (MapReduce) and database component (HDFS). The benchmarking tasks selected can be categorized as micro-benchmarks, web search tasks, machine learning tasks, and HDFS benchmark.

- Micro-benchmarks include Sort [19], WordCount [20], and TeraSort [21]. Sort, which simply sorts a large collection of data, is intended to represent a class of MapReduce problem which transforms a data set. Similarly, WordCount is intended to represent a class which extracts a small amount of data from a large data-set. TeraSort is another sorting task but with a larger data-set. All of the micro benchmarks use tools included in HiBench to generate their input data-sets.
- Web search benchmarks, which include Nutch Indexing [22] and PageRank [23], test the ability to handle search-indexing systems. Nutch Indexing workload generates inverted index files from an input of web page links. PageRank calculates ranks of web pages according to the number reference links.
- Machine learning tasks include two workloads, Bayesian Classification and K-means clustering, from the Mahout library [24] which are used to test Hadoop's machine learning processing capabilities. Bayesian classification, a popular algorithm for data mining, is used on processed portions of Wikipedia [25]. The K-means algorithm, also popular for data-mining, is used to iteratively compute an approximation of the centroid of a multi-dimensional array which is randomly generated by HiBench.
- HDFS uses Extended DFSIO, an enhanced version of the DFSIO [26] program which is part of Hadoop. Extended DFSIO is file system benchmarks for finding the throughput of simultaneous read and write operations.

## 3.4 Yahoo! Cloud Serving Benchmark

Yahoo! Cloud Serving Benchmark (YCSB) is a tool developed by Yahoo! to benchmark their PNUTS [27] serving system. This benchmark focuses on scalable serving systems which provide read and write access to data. YCSB separates the task of benchmarking serving systems into two different tiers.

Tier 1 encompasses general performance as measured by the latency of a request when the database is under load. To test the balance of throughput and latency, the latency of a request is monitored as the throughput is increased. Tier 2 examines scaleup and elastic speedup, the serving system's ability to scale with increased load. This is achieved by observing the impact that adding more machines to the system has on the performance of the system. The ability of the system to scaleup well is described by system's latency remaining constant across multiple tests where the workload and server count are both increased. Elastic speedup measured test the impact of additional servers while a fixed size workload is running.

To test the performance and scalability of a serving system, YCSB uses a randomly generated workload instead of modelling a specific application. The YCSB client generates a dataset and operations according to a workload profile. The workload profiles contain user specifications for random distributions which are used to generate which operations will occur on which record.

## 3.5 CloudSuite

CloudSuite is a collection of benchmarking tasks which were used to characterize the inefficiencies in the micro-architecture of modern server CPUs used in a cloud computing environment. The benchmarking tasks were identified as some of the more common tasks which are handled using cloud computing. These tasks included data serving, MapReduce, media streaming, SAT solving, web hosting, and web search.

- Data serving – Cassandra [28] database exercised with a read-heavy YCSB workload.
- MapReduce - The Mahout library's Bayesian classification algorithm was run on a Hadoop cluster. The algorithm is used to process a portion of Wikipedia to guess the country tag for each article.
- Media streaming – The Darwin Streaming Server receiving request from simulated users generated by Faban.
- SAT solving – Cloud9 [30] parallel symbolic execution engine's Klee SAT solver
- Web hosting – CloudStone including Olio and Faban.
- Web search – Nutch/Lucene [31] index serving node receiving request from simulated users generated by Faban.

TABLE II
CLOUD COMPUTING BENCHMARK WORKLOADS

| **Target Application** | **Workload** |
|---|---|
| Database | YCSB |
| Legacy application | CloudCmp |
| MapReduce | HiBench<br>Mahout Bayesian classification |
| Media streaming | Darwin Streaming Service |
| Web 2.0 | CloudStone |

## 4 Conclusion

Cloud computing offers organizations the ability to scale to the size of their user base more efficiently and thus offers a competitive advantage if the proper services are selected. In this paper, we have presented available benchmarking tools for cloud computing services. CloudCmp offers an approach to benchmarking the individual cloud computing services offered by a provider. CloudStone provides a social networking application with simulated user interaction to test Web 2.0 applications. HiBench collects realistic workloads for the MapReduce processing framework. YCSB tests the performance and scalability serving systems with generated workloads. Finally, CloudSuite suggests workloads to capture the behaviour of the more common tasks in a cloud computing environment.

## 5 References

[1] Li, Ang, et al. "CloudCmp: comparing public cloud providers." *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010.

[2] Sobel, Will, et al. "Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0." *Proc. of CCA*. 2008.

[3] Huang, Shengsheng, et al. "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis." *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*. IEEE, 2010.

[4] Cooper, Brian F., et al. "Benchmarking cloud serving systems with YCSB."*Proceedings of the 1st ACM symposium on Cloud computing*. ACM, 2010.

[5] Ferdman, Michael, et al. "Clearing the clouds: a study of emerging scale-out workloads on modern hardware." *ACM SIGARCH Computer Architecture News*. Vol. 40. No. 1. ACM, 2012.

[6] Hennessy, John L., and David A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2012.

[7] Mell, Peter, and Timothy Grance. "The NIST definition of cloud computing (draft)." *NIST special publication* 800.145 (2011): 7.

[8] Hadoop homepage. http://hadoop.apache.org/

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," USENIX OSDI, December, 2004.

[10] Ghemawat, Sanjay, Howard Gobioff, and Shun-Tak Leung. "The Google file system." *ACM SIGOPS Operating Systems Review*. Vol. 37. No. 5. ACM, 2003.

[11] Google AppEngine. http://code.google.com/appengine.

[12] Amazon Web Service. http://aws.amazon.com.

[13] MicrosoftWindows Azure. http://www.microsoft.com/windowsazure.

[14] GoGrid Cloud Hosting. http://gogrid.com.

[15] Rackspace Cloud. http://www.rackspacecloud.com.

[16] SPEC Java Virtual Machine Benchmark 2008. http://www.spec.org/jvm2008

[17] Iperf. http://iperf.sourceforge.net.

[18] Faban Documentation. Sun Microsystems. 2009. http://faban.sunsource.net/0.9/docs/toc.html

[19] Sort program. Available in Hadoop source distribution: src/examples/org/apache/hadoop/examples/sort

[20] WordCount program. Available in Hadoop source distribution:src/examples/org/apache/hadoop/ examples/WordCount

[21] Hadoop TeraSort program. Available in Hadoop source distribution since 0.19 version: src/examples/org/apache/hadoop/examples/terasort

[22] Nutch homepage. http://lucene.apache.org/nutch/

[23] P. Castagna, "Having fun with PageRank and MapReduce," Hadoop User Group UK talk. Available: http://static.last.fm/johan/huguk-20090414/paolo_castagna-pagerank.pdf

[24] Mahout homepage. http://lucene.apache.org/mahout/

[25] Wikipedia Dump. http://en.wikipedia.org/wiki/index.php?curid=68321

[26] DFSIO program. Available in Hadoop source distribution: src/test/org/apache/hadoop/fs/TestDFSIO

[27] B. F. Cooper et al. PNUTS: Yahoo!'s hosted data serving platform. In VLDB, 2008.

[28] The Apache Cassandra Project. http://cassandra.apache.org/.

[29] Darwin Streaming Server homepage. http://dss.macosforge.org/

[30] Liviu Ciortea, Cristian Zamfir, Stefan Bucur, Vitaly Chipounov, and George Candea. Cloud9: a software testing service. ACM SIGOPSOperating Systems Review, 43:5–10, January 2010

[31] Lucene homepage. http://lucene.apache.org

[32] Armbrust, Michael, et al. "A view of cloud computing." *Communications of the ACM* 53.4 (2010): 50-58.

[33] Foster, Ian, et al. "Cloud computing and grid computing 360-degree compared."*Grid Computing Environments Workshop, 2008. GCE'08*. Ieee, 2008.

[34] Grossman, Robert L. "The case for cloud computing." *IT professional* 11.2 (2009): 23-27.

[35] Binnig, Carsten, et al. "How is the weather tomorrow?: towards a benchmark for the

cloud." *Proceedings of the Second International Workshop on Testing Database Systems*. ACM, 2009.

[36] Alexandrov, A., et al. "Benchmarking in the Cloud: what it should, can, and cannot be." *4th TPC Technology Conference on Performance Evaluation and Benchmarking (TPCTC), VLDB*. 2012.

[37] Huppler, Karl. "The Art of Building a Good Benchmark." *Performance Evaluation and Benchmarking*. Springer Berlin Heidelberg, 2009. 18-30.

[38] Kavis, Michael J. *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*. John Wiley & Sons, 201

# Big Parameter Data Analysis for Semi-conductor Manufacture

**Jain-Shing Wu, Ming-Chun Tsai, Sheng-Wei Chu, and Chung-Nan Lee**
Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan

**Abstract -** *Statistics help the manufacturers to maintain the quality in manufacturing process, especially in the mechanical and engineering areas. Monitoring and analyzing the manufacturing data in real time, the quality can be increased almost instantly. However it is a hard task due to huge data or records gathered from wafer manufacturing logs. Fortunately, big data analytics can explore the granular details of the enormous manufacturing data with a variety of parameter values to uncover the abnormal parameters, unknown correlations and other useful information. In this paper, the Gaussian distribution method and cloud genetic algorithm are used to analyze and find out products/machines with significant defects from manufacturing logs. Experimental results show that our method is efficient in figuring out the abnormal machines and parameters, and comparing with standalone machine, the proposed algorithm has 4.32 times faster.*

**Keywords:** Big data analytics; machine protection; cloud Genetic algorithm

## 1 Introduction

In the automation control era, increasing production yield rate has become an important issue since the higher production yields on behalf of the company's manufacturing capacity is higher, and thus enhances the company competitiveness and increases the potential customer base. But human resource is often unaffordable for monitoring more and more complex production processes of a factory. Engineering Data Analysis (EDA) systems have been therefore employed to collect, process, and monitor a large number of parameters from production equipment and diagnose tool health [1, 2].

Traditionally, statistical tools for EDA are facilitate to maintain the quality in manufacturing process, especially in the mechanical and engineering areas [3]. By using the right process for statistical tracking and real time feedback, the quality can be increased almost instantly. Due to traditional statistical data analysis has become inadequate at providing equipment fault detection and diagnosis [4], however, many high-tech manufacturers have had a hard time completing a real time traditional statistics program due to huge data or records gathered for tracking a product [5]. For example, the health data of the semiconductor wafer machine often involves highly correlated parameters and time-varying behaviors [2].

In addition, the investigation showed that a sudden breakdown accounts for 60% of machine maintenance costs [6]. There is therefore a significant requirement for the development and application of efficient and effective approaches to monitor the health state of equipment and predict unscheduled failure, especially for semiconductor industry [7]. In fact, many researchers from academia as well as industries are getting involved into identifying the most probable causative factors in manufacturing field [8].

In brief, owing to advances of modern information technologies and new applications, intelligent and statistical techniques should be integrated to explore the granular details of the enormous manufacturing data with a variety of parameter values for fault detection to enhance the yield [5]. Especially, thousands of parameters of each product may need to be stored properly and accessed in-time. This cannot be easily accomplished by the traditional computing architecture. Fortunately, big data analytics with Cloud Computing (CC) can be employed as soon as possible to uncover the abnormal parameters, unknown correlations and other useful information for a factory automation environment.

To sum up, there are two main problems to be solved. The first problem is how to access and storage huge data quickly, even how to recover it, when the storage disk is broken. The second one is how to figure out the impacting factor for the yield rate of automation product in a short time. In this study, we adopt MapReduce programming model [9] proposed by Google on Hadoop distributed platform [10], which is one of CC platforms, to automatically parallelize the computation across large-scale clusters of machines that makes efficient use of the network and disks. Base on the CC platform, statistical indicators (such as mean, standard deviation, maximum, minimum, and range) through control charts and Artificial Intelligence (AI) technologies are integrated to implement for early warning of key equipment excursion.

In the Section 2, we will give some essential background materials. The proposed algorithm is given in the proposed method section. Experimental results are given in the experiment section. And we give a conclusion in the last section.

## 2 Background Materials

The wafer fabrication process for producing integrated circuit (IC) consists of a lengthy sequence of complex physical and chemical processes. Nowadays, semiconductor fabrication facilities have already collected the parameters of the fabrication processes, materials, and equipment involved in the product manufacturing [4]. The recorded parameters in different fabrication processes may have implicit correlations. For example, temperature and humidity are the main parameters in the first and second processes, respectively. Except previously known that high temperature with high humidity makes product fail very easily, we can expose the relationship by analyzing the large amount of log data on CC platform.

Motivated by real needs, Hadoop, a popular open-source framework for CC, implements a MapReduce engine and a distributed user-level file system named Hadoop Distributed File System (HDFS). Written in Java for portability across a variety of platforms, such as Linux, Mac OS/X, and Windows etc. and only require commodity hardware, Hadoop benefits a wide range of commercial and academic users for big data processing. MapReduce is a parallel programming model for processing and generating large datasets. Programmers express the computation as two functions: map and reduce. The former takes an input pair and produces a set of intermediate key/value pairs, and then applying the latter to all the values that shared the same key in order to combine the derived data appropriately. Hadoop automatically parallelizes the computation across large-scale clusters of machines, schedules inter-machine communication, and handles machine failures.

For big data analysis of semiconductor manufacturing data, it is a very appropriate way to specify the computation in terms of a map and a reduce function owing to the following characteristics of Hadoop:

*1. High reliability* - Each computing slave node registers its status to master node at designated times. For high computing quality guarantee, if the slave node does not return computing results in a pre-defined time, master node will re-allocate the jobs and data fragments to other slave node.

*2. Fault tolerance* - Master node automatically storages regular progresses in order to prepare to response recovery request. In addition, HDFS, the all input/output data handle system of Hadoop, splits data into fixed fragment size (default is 64MB) and keeps backup copies to different data nodes (default is 3 copies). Based on the design of Hadoop, the master-slave architecture provides very good fault tolerance mechanism.

*3. Load balance* - Master node dynamically allocates computing jobs to slave nodes in order to trade off overall execution time saving and efficient resource utilization.

*4. Virtualization and dynamic resource allocation* - The Cloud infrastructure offers virtual private links and allows for the provision of resources on-demand, thus resources are allocated in an elastic way, according to consumers' needs [11].

## 3 Proposed Method

In this section, the Hadoop MapReduce technique is employed in order to provide a CC platform for semiconductor manufacturing data analysis. Figure 1 shows the flow diagram of proposed system architecture for the big data analytics.

First, we receive the data from machine log files which are stored in HBase. And then, we correct the missing values in data. And then, we send the data to the abnormal parameters detection module. This module checks outliers in the parameters by using Gaussian distribution method via MapReduce technique. The key and value pair used in mapper is <parameter #, parameter value>. And then, the reducer collects key-value pairs and calculates the means and deviations. The outliers are the data whose distance to the means is more than 2 times deviations. The parameters that contain lots of outliers are treated as abnormal parameters.

Before performing the Cloud Genetic Algorithm (CGA) [12, 13], the wafer logs are first fixed the missing values and then normalized in order to eliminate the difference between scales of different parameters. The normalizing function $\mathrm{Nor}(x_i^j)$ is used to normalize all wafer data using the same recipe, and it is defined as follows:

$$Nor(x_i^j) = \frac{x_i^j - x_{i,min}}{x_{i,max} - x_{i,min}}, \forall i = 1,2,\ldots,n \ and \ j = 1,2,\ldots,m \qquad (1)$$

where $x_i^j$ represents i-th parameter value of j-th wafer, and m is the total number of wafers using the same recipe, and n is the total number of parameters. $x_{i,max}$ and $x_{i,min}$represent the maximum and minimum parameter values of i-th parameter for all wafers adopting the same recipe. After normalization, the CGA performs. The individuals of CGA are set to a binary string that shows the parameters are used to be further classified by K-nearest neighbor (KNN) [14] or not. We use the KNN to classify the data into normal and abnormal wafers and calculate accuracy as fitness values in the evaluation process. One-point crossover and mutation are adopted in CGA. The crossover rate and mutation rate are 0.8 and 0.01.

Also we used the MapReduce technique for calculating the distance between different data when performed KNN.

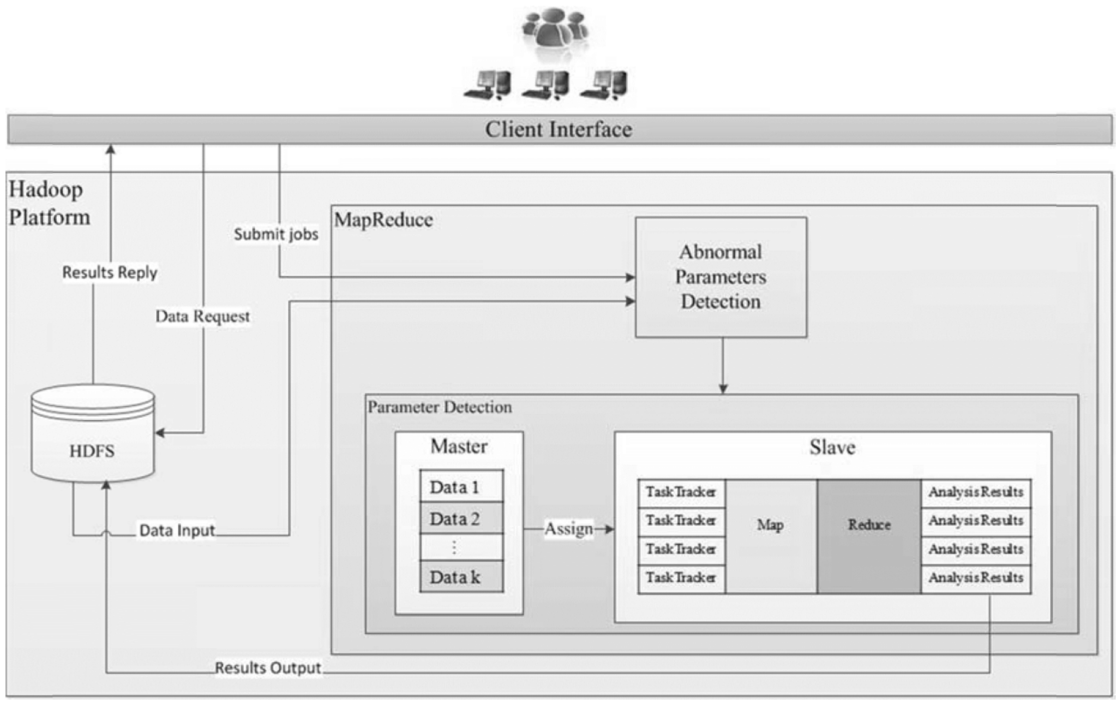


Figure 1. The flow diagram of the proposed system architecture for the big data analytics.

## 4 Experimental Results

The proposed system is developed by using Java language and MapReduce technique on Hadoop. The experimental environment adopts 2 clusters of computing nodes, each clusters contains 1 master node and 2 slave nodes, and 1 node for central receiver. Table 1 shows the information of a single node.

Table 1 Specification of the computing nodes.

| Parameter | Specification |
|---|---|
| Number of nodes | 6 |
| OS | Ubuntu 12.04 (32-bit) |
| Memory | 4GB |
| CPU | Virtual Pentium D (dual core) 2.8GHz*4 |

The 6 computing nodes are partitioned into 2 different Hadoop clusters as the environment of ICGA. In each Hadoop cluster, there are 3 computing nodes that one is master and the other two are slaves.

In this experimental data, we use two different log files; one is composed of 99 samples and the other is composed of 2488 samples. Each has 915 parameters selected for detection tasks.

First of all, we performed the outlier detection on one Hadoop cluster. Our goal is to select the sensitive parameters that are affected by more than 10% outlier tools whose standard deviations are more than 2 standard deviations. As listed in Table 2, we detect 592 parameters from 99 sample log and 528 parameters from 2488 sample log.

Table 2 Number of the sensitive parameters in wafer logs.

| | Number of sensitive parameters |
|---|---|
| No. of parameters from 99 samples | 592 |
| No. of parameters from 2488 samples | 528 |

In the anomalous detection, feature selection is carried out before implementing ICGA. The S2N technique is adopted to remove the unrelated parameters and select the top 25% parameters from 915 parameters as the preliminary significant parameters. And then, these parameters are used in ICGA for selecting the most critical parameters.

In order to increase the exploration ability, we perform different cloud GA on different Hadoop clusters (islands). One cluster applies single-point crossover and single-point mutation. The other one adopts two-point crossover and two-point mutation. The best solutions on respective island are exchanged through the central receiver. The best solution is selected among the sent solutions and sent back to each island. The stop criterion is the classification accuracy is 100% and the number of parameters is less than or equal to 25% of the number of the preliminary significant parameters. The parameters for ICGA are listed in Table 3.

We use the ten-fold cross-validation to verify our mining methods combined with the outlier detection and ICGA. In each fold, we performed the outlier detection and ICGA, and then we averaged the convergence time and accuracy. For the smaller data, which is composed of 99 samples, the experiment shows that ICGA obtain the critical parameters in fewer generations than that in the standalone GA. It takes more time to perform ICGA than to run the standalone GA in small case, since MapReduce technique requires more time to separate and distribute data. However, it shows a great performance when handling with big volume of data. Hence, for the 99 sample case, the data is not big enough to show the true power of the proposed system. For testing the accuracy for small volume of data, both ICGA and the standalone GA have 100% accuracy rate. The comparison is listed in Table 4.

For 2488 sample case, the experimental result shows ICGA obtains the critical parameters in fewer generations than the standalone GA. Moreover, it shows a significant improvement in efficiency that the convergence time in ICGA is 4 times less than that in the standalone GA. For testing the accuracy, both ICGA and the standalone GA have 100% accuracy. The results are given in Table 5.

Table 3 Parameter values used for different islands.

| | Island 1 | Island 2 |
|---|---|---|
| Population Size | 100 | 100 |
| Crossover Rate | 0.8 | 0.8 |
| Mutation Rate | 0.1 | 0.1 |
| Way of Crossover& Mutation | Single-point | Two-point |
| Island Exchange Rate | 50 generations | 50 generations |

Table 4 A comparison of efficiency with ICGA and standalone GA in 99 samples.

| | ICGA | Standalone GA |
|---|---|---|
| Convergence Time | 154 generations (81 seconds) | 187 generations (75 seconds) |
| Accuracy Rate | 100% | 100% |

Table 5 A comparison of efficiency with ICGA and standalone GA in 2488 samples.

| | ICGA | Standalone GA |
|---|---|---|
| Convergence Time | 160 enerations (2.682 hours) | 185 generations (11.58 hours) |
| Accuracy Rate | 100% | 100% |

## 5 Conclusions

In this paper, we have proposed a novel ICGA that can not only efficiently process big data but also have a high accuracy in detecting the most discriminative parameters. With the MapReduce technique, performance speeds up by more than 4 times in big data. Furthermore, the proposed ICGA based on the MapReduce technique has avoided the unnecessary map-reduce procedure so that it can enhance the efficiency.

## 6 Acknowledgements

The research is based on work supported by the NSC 102-2410-H-110-083- project from National Science Council, Taiwan.

## 7 References

[1] A. Chen and J. Blue, "Recipe-independent Indicator for Tool Health Diagnosis and Predictive Maintenance," IEEE Transactions on Semiconductor Manufacturing, vol. 22, pp. 522-535, 2009.

[2] A. Thieullen, M. Ouladsine, and J. Pinaton, "Application of PCA for Efficient Multivariate FDC of Semiconductor Manufacturing Equipment," in 24th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC), Saratoga Springs, NY, 2013, pp. 332-337.

[3] I. Saleem, M. Aslam, and M. Azam, "The use of Statistical Methods in Mechanical Engineering," Research Journal of Applied Sciences, Engineering and Technology, vol. 5, pp. 2327-2331, 2013.

[4] M. P.-L. Ooi, E. K. J. Sim, Y. C. Kuang, S. Demidenko, L. Kleeman, and C. W. K. Chan, "Getting More From the Semiconductor Test: Data Mining With Defect-Cluster Extraction," IEEE Transactions on Instrumentation and Measurement, vol. 60, pp. 3300-3317, 2011.

[5] C.-Y. Hsua, C.-F. Chienb, and P.-N. Chen, "Manufacturing Intelligence for Early Warning of Key Equipment Excursion for Advanced Equipment Control in Semiconductor Manufacturing," Journal of the Chinese Institute of Industrial Engineers, vol. 29, pp. 303-313, 2012.

[6] J. Lee. (2005). Intelligent Maintenance Systems (IMS) Technologies. Available: http://www.imscenter.net/Resources/IMS Chinese Introduction.pdf

[7] L. Bechou, D. Dallet, Y. Danto, P. Daponte, Y. Ousten, and S. Rapuano, "An Improved Method for Automatic Detection and Location of Defects in Electronic Components Using Scanning Ultrasonic Microscopy," IEEE Transactions on Instrumentation and Measurement, vol. 52, pp. 135-142, 2003.

[8] B. Hu, C. K. Pang, M. Luo, X. Li, and H. L. Chan, "A Two-Stage Equipment Predictive Maintenance Framework for High-Performance Manufacturing Systems," in 7th IEEE

Conference on Industrial Electronics and Applications (ICIEA), Singapore, 2012, pp. 1343-1348.

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Communications of the ACM, vol. 51, pp. 107-133, 2008.

[10] J. Shafer, S. Rixner, and A. L. Cox, "The Hadoop Distributed Filesystem: Balancing Portability and Performance," in 2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS), White Plains, NY, 2010, pp. 122-133.

[11] G. E. Gonçalves, P. T. Endo, T. D. Cordeiro, A. V. A. Palhares, D. Sadok, J. Kelner, et al., "Resource Allocation in Clouds: Concepts, Tools and Research Challenges," in XXIX Brazilian Symposium on Computer Networks and Distributed Systems, Brazil, 2011, pp. 197-240.

[12] M. Mitchell, An Introduction to Genetic Algorithms. Cambridge, England: MA: MIT Press, 1996.

[13] F. Ferrucci, M.-T. Kechadi, P. Salza, and F. Sarro. (2013). A Framework for Genetic Algorithms Based on Hadoop. Available: http://arxiv.org/pdf/1312.0086.pdf

[14] N. S. Altman, "An Introduction to Kernel and Nearest-neighbor Nonparametric Regression," The American Statistician, vol. 46, pp. 175-185, 1992.

## SESSION

## CLOUD COMPUTING AND Grid - RELATED ISSUES

### Chair(s)

**TBA**

# A Highly Available Generic Billing Architecture for Heterogenous Mobile Cloud Services

**P. Harsh**[1], **K. Benz**[1], **I. Trajkovska**[2], **A. Edmonds**[1], **P. Comi**[3], **and T. Bohnert**[1]
[1]InIT Cloud Computing Lab, Zurich University of Applied Sciences, Winterthur, Kanton of Zurich, Switzerland
[2]Dpto. Ingeniería de Sistemas Telemáticos, Universidad Politécnica de Madrid, Madrid, Spain
[3]Innovation & Research, Italtel S.p.A., Castelletto, Milan, Italy

**Abstract**—*Rating, Charging, Billing (RCB) is the fundamental activity that enables a business to generate revenue stream depending on the resource consumption by their consumers. Traditionally, telecom operators have used custom designed, vertically integrated solution for RCB which often results in a complex system that is difficult to adapt to new service offerings. With telecom operator's desire to capitalize on cloud computing by using their vast amount of infrastructure, the need for a RCB solution that serves the needs of cloudified telcos is needed.*

*In this paper we present an approach to implement a generic rating, charging, and billing engine that serves the business and technical needs of both cloudified telecom services and those of cloud service providers. Key to this is a generic accounting process to drive the design of the generic RCB architecture. We show how RCB as a service can be offered catering to not only traditional telco services, the new cloud services they wish and will offer, but packaged cloudified services to the consumers and application developers as well. Finally, we detail how our architecture can be distributed and key services replicated to ensure high-availability.*

*The end result of this paper is a solution that can enable telecom service providers to leverage the rapidly growing and accelerating cloud service market.*

**Keywords:** economics, rating, charging, billing, high-availability, cloud

## 1. Introduction

In the telecoms' domain, the RCB process has been very tightly-coupled and vertically-integrated with their services. Therefore any new value addition (e.g. cloud services) on top of the offered service necessitates a complete overhaul of the RCB strategy, and many times technological ones, by the businesses. In this era of mash-ups and composed services, there is a real need of a completely generic RCB platform that can potentially support any composed service today and in the future.

We are conducting this research as part of Mobile Cloud Networking (MCN) project. MobileCloud goal is the convergence of the telecom and cloud worlds. Essentially it equates to: *Mobile Network + Decentralized Computing + Smart Storage* offered as one service based on cloud computing principles [1] e.g. on-demand, elastic, pay-as-you-go model.

Telecom services are normally offered over vertically integrated systems, comprising of Radio Access Network (RAN), Enhanced Packet Core (EPC), and IP Multimedia Subsystem (IMS). These services are supported by standards such as Diameter [2] and Radius [3] that provides Authentication, Authorization, and Accounting (AAA) support. More details of the current state of the art can be found in Section VI. With the emergence of smart-phones and always connected mobile devices, more and more value is created by mobile application developers on top of cloud services. Traditional telecom operators are being increasingly delegated to simply provide a dumb data pipe for such rich-experience mobile apps.

With a departure from traditional service models, the tightly integrated RCB solutions used currently, are rendered insufficient in dealing with the new models in MCN that will support dynamic service compositions using elements from both traditional telecom domain to be offered as a service plus elements of clouds offered as a service. Each composed-service being offered to the user (the application developer, or a Mobile Virtual Network Operator), can be offered by a single provider in its entirety, or individual services could be offered by independent operators.

This work plans to address this issue by providing an architecture that adapts to existing ones and models (Section IV) which help telecom operators embrace cloud computing principles and make an operator efficient through the use of clouds. In addition, MCN also aims at enabling new business models by extending the cloud, so that, an operator can provide customized bundled platforms comprised of cloud services and telecom features such as EPC to application developers. This would enable application developers to create next generation of fully integrated, rich mobile applications through custom provisioned app-development environments, customizing not just the traditional data-center elements, but also the elements of the telecom service stack.

And therein lies the motivation and need of a model for developing a RCB solution which is generic in nature so as to support requirements (Section III) of composed services in a completely uniform manner. The proposed solution in this paper aims to be fully extendible in order to support new services that will be offered in the near-future.

Regardless of the nature of service offered, a business must conduct an internal accounting process in order to bill it's customers, and this process should be general across businesses and agnostic to the services offered. Hence in this paper we investigate how we can exploit this financial process for creating a completely generic rating-charging-billing model, as detailed in Section II.

With the this approach, RCB as a service can be offered to any generic service provider and support both the traditional monolithic service models, as well as new cloud-based atomic and composed service paradigm.

## 2. Accounting Process and Pricing Models

In order to comprehend the architectural design requirements on a generic RCB system, it is important to look into the overall accounting process and different pricing models that an organization could use in their billing process.

### 2.1 Accounting Process

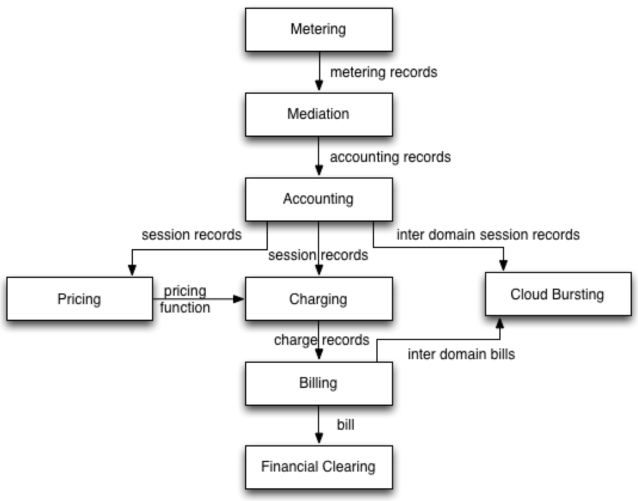


Fig. 1: General Accounting Process

In [4], the authors have captured the financial process for accounting cloud services. Figure 1 provides the overview of such an accounting process. It explains the general workflow and relations from the metering phase to the financial clearing process where the customer settles the invoice after the payment is processed. For our purpose, *we slightly adapted the concepts to support cloud bursting*. In our slightly adapted approach, various phases in the accounting process are -

- **Metering** - the process of collecting the various resource usage metrics of the consumers. This process is critical as without the raw metered data we can not properly customize our billing strategy. Without metering, businesses could offer their services essentially at a flat rate regardless of how high or low the customer's consumption is.
- **Mediation** - the process of assimilating and transforming the usage records that comes from different meters into a meter-agnostic format which could be processed by other modules in the accounting process cycle.
- **Accounting** - this part of the overall process is normally tasked with secured long term storage of accounting records generated by the mediation module, until at least the legally required timeframe. It also analyzes the accounting records and generates the session records for further processing. The stored accounting records come in handy in case of any billing dispute from the customers.
- **Pricing** - depending on the resource type, the pricing strategy will vary, e.g. - a provider may offer a flat rate for up-to 1 TB of storage, but the network bandwidth pricing could be based on the units of data sent/received. This function, depending on the resource type, outputs the appropriate pricing function to be applied to the accounting records.
- **Charging** - this is the process of applying the appropriate pricing functions to the accounting records to generate the charge records. Charge records contains the monetary value associated with the resource usage by the customer.
- **Roaming / Cloud Bursting** - This component is inspired by the roaming charges that one has to pay in the telecom domain. Similarly, if there is a cloud bursting scenario, then one has to consider that in the overall accounting process. This aggregation of billing information from external organization could be governed by special arrangements between providers. All these aspects can be handled at this phase in the overall process.
- **Billing** - this is the process of consolidating all the charge records since the last billing cycle. This stage also takes into account any discounts that were applicable in the cycle. Bills are generated for the customers as an output of this phase.
- **Financial Clearing** - generating bills is one aspect, sending the bills out to the customers and processing the payments through financial clearing houses is the main task supported in this phase of the financial process.

### 2.2 Pricing Models

In [4] [5], the authors also covered popular pricing models. In this section we summarize their findings. Pricing models are key in realizing an optimal revenue stream for the services being offered. The most common pricing models are[1] *time-based, volume-based, QoS based, flat-rate, Paris-metro model, priority-based, smart-market model, edge, responsive, proportional-fairness, cumulus, session-oriented, one-off and time-of-day based*. The correct choice of the pricing function for charging the resources could help differentiate one's service from the competition.

Depending on the business scenario, one may have to adapt the generic pricing models. Some of the variations commonly used today are *free of charge, periodic-fees, discounts, pre-paid, online-accounting, offline-accounting, static-pricing, dynamic-pricing*, etc.

The generic RCB implementation, if to be used as a service by several customers, must be capable of supporting most of the pricing models and common variations used today. We will see later how our proposed architecture addresses the challenge.

## 3. Design Requirements

RCB system requirements in MCN are influenced by the general architectural requirements. The MCN architecture is service centric. Core telecom functions such as EPC, RAN, and BBU are offered as services. Some services have a built-in legacy "rating-charging" component, in which scenario, the proposed RCB architecture must utilize the charging data. In other cases the data format and message flows are to be designed in a completely service agnostic manner. Furthermore, RCB is the key process that leads to revenue generation, such a system should be highly available.

### 3.1 MCN Global Architecture

The MCN architecture follows a service oriented architecture. In the MCN architecture, all functional elements are modelled as services. The key architectural entities of the MCN architecture are:

- **Service Manager (SM)**: It provides an external interface to the user both programmatic and/or visual. It offers multi-tenant capable services to that user. The SM has two dimensions; the business which encodes business agreements, and the technical that manages the different Service Orchestrators of a particular tenant.
- **Service Orchestrator (SO)**: It embodies how the service is actually implemented. Generally, one SO per SM domain is instantiated per tenant. It oversees the complete (end-to-end) orchestration of a service instance (SI). It is implemented as a domain specific component and manages the service instance, which it creates, including scaling of the instance. The SO is managed by the SM and the SO monitors SI specific metrics related to the service instance. Although SIs are domain-specific, they are composed of service instance components (SIC).

[1]for details please refer to the original study

- **CloudController (CC)**: Supports the deployment, provisioning, and disposal of SOs. To the SOs it also provides both atomic and support services through a Service Development Kit (SDK).

Below is a diagram of their relationships:

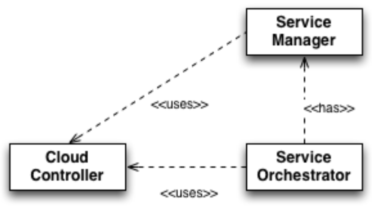


Fig. 2: Mobile Cloud Networking Architectural Entities and Relationships

Each architectural entity and service within MCN shares a common lifecycle model. The lifecycle model used in MCN is divided into two complementing phases, the business and the technical. For the business life cycle phase, the following stages are defined:

- **Design**: the service that will be offered is formulated and understood how it can be created from internal and outsourced services.
- **Agreement**: with a set of services identified, agreements related to service level agreements (SLA), pricing and access (AAA) can be entered with those service providers.

For the technical life cycle phase, the following stages are defined:

- **Design**: at this stage the service's technical design is carried out.
- **Implement**: with a service design the service is implemented. This entails the implementation of a SM and SO.
- **Deploy**: In order for the SM to take requests to create new service instances, the SO needs to be deployed using the CC.
- **Provision**: this phase is where the SO is instantiated and begins to create the services necessary to satisfy the SO's needs.
- **Runtime and Operation**: the SO has completed its job of providing the tenants service instance and is now monitoring and managing the service instance. It is during this step where scaling in and out of components is carried out.
- **Disposal**: the service instance's sub-components are destroyed and deleted.

To be integrated in MCN, the RCB architecture has to implement the SM and SO MCN architectural entities. As there is an existing CloudController within MCN, RCB as presented here can simply reuse it through the SDK.

## 4. RCB Architecture

A high level RCB architecture is shown in figure 3. It showcases all the functional elements needed to handle different stages of a complete financial process. However it does not show in detail how the various elements of the overall architecture can be distributed and does not describe the communication interfaces between various modules.

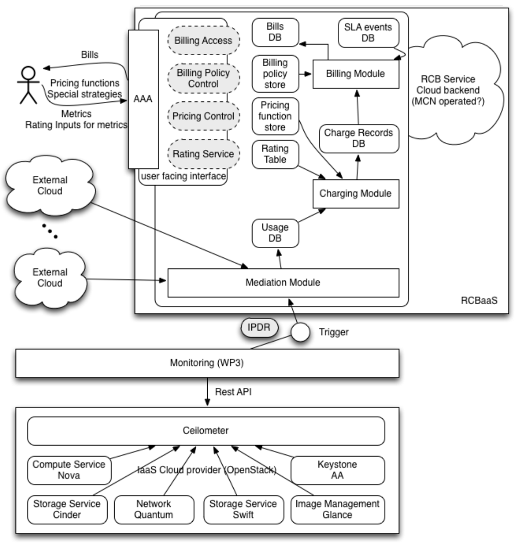


Fig. 3: Generic Rating, Charging, Billing Architecture

The figure 3 shows OpenStack [6] and Ceilometer [7] monitoring as an example environment over which RCB could be deployed. The overall architecture is general enough to handle any service type as long as it can send necessary metrics data to the RCB service instance. The metric records from various services could be represented in any data representation format standard. A likely candidate is IP Detail Records (IPDR) [8] standard.

In the overall architecture diagram, the various metrics taken from numerous (internal and external) channels come into the *Mediation Module*, whose task is to standardize the data format - translate from various supported data formats into a uniform format for other modules to consume.

The *Mediation Module* output i.e. the translated data records are then processed by an analytics engine (not shown in the overall architecture) to generate the usage records which are stored in the usage database for future retrieval and processing.

The *Charging Module* takes in a rating strategy and pricing function and processes the usage records to generate charge records. These charge records must be in a resource neutral format at this stage. The charge records could be generated periodically - as frequently as needed (configuration dependent) and stored in a secure database for future retrieval and processing by other modules.

The rest of the components' functionality is self-explanatory. The overall architecture shown is very easy to distribute. With a cloud service provider with several data-centers, the architecture can be split into two, collect usage and generate charge data locally at each data-center; collect the charge records from multiple locations, process and generate the bills in one data-center.

Since in MCN, RCB is to be provided as a service, we will present the design discussions from the implementation and deployment perspective.

### 4.1 Key Architectural Components

In this section we will describe key architectural elements that could be implemented as a standalone module which would interact with rest of the RCB architectural elements via secure message bus.

#### 4.1.1 Mediation Submodule

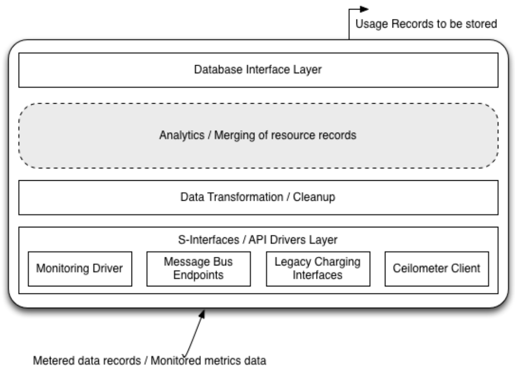

Usage Records to be stored
Database Interface Layer
Analytics / Merging of resource records
Data Transformation / Cleanup
S-Interfaces / API Drivers Layer
Monitoring Driver
Message Bus Endpoints
Legacy Charging Interfaces
Ceilometer Client
Metered data records / Monitored metrics data

Fig. 4: Mediation Submodule

Figure 4 describes in details the internal components of the mediation module. This module can be implemented as a highly available standalone service. The mediation module would be composed of -

- **S-Interfaces / API Drivers** - The southbound interface implements drivers for popular monitoring systems (Nagios [9], Ganglia [10], Zabbix [11], etc.) through which resource usage data can be filtered. It also implements the message-bus endpoints management for services that wish to send usage data directly to RCBaaS. Ceilometer is another optional client that could be supported.
- **Data Transformation / Cleanup** - The data coming through the southbound interfaces could be in disparate formats, they must be transformed in a common format for other modules to process in a uniform manner. They could be transformed into IPDR [8] records.
- **Analytics** - The monitored usage records in some situation needs to be combined together as part of a single user session. The analytics module analyzes the

individual data records and performs the classification and statistical aggregation. The analytics engine can be implemented as an extendible engine where the users could supply their own analytics logic (ex. Datahero [12], Quantopian [13]).

- **DB-Interface** - Several popular data-store interfaces must be supported so as to provide flexibility with the choice of target store where the processed *usage records* could be kept for a configured time period.

### 4.1.2 Charging Module

The charging module uses the *usage record* from the *Usage Records DB* and applies the pricing function together with the rating strategy depending on the resource type to generate the *charge records* which is stored in *Charge Records DB* for future retrieval and analysis by other modules. Similar to *usage records*, the *charge records* are represented in a neutral, standard format, agnostic to the resource that resulted in such a record. This way the high-level modules are shielded from the low-level resources (ex. CPU, Disk, Network I/O, etc.).

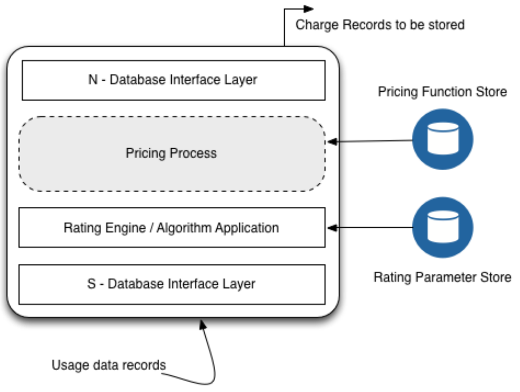

Charge Records to be stored
N - Database Interface Layer
Pricing Function Store
Pricing Process
Rating Engine / Algorithm Application
Rating Parameter Store
S - Database Interface Layer
Usage data records

Fig. 5: Charging Submodule

Figure 5 shows the charging-module components. The *S - Database Interface Layer* connects to the *Usage Records DB* to retrieve the data records for further processing. The *Rating Engine* governed by the rating process parameters / configuration values, processes the usage data and sends an intermediate data record to the *pricing process* for application of appropriate pricing function from the *Pricing Function Store*. The selection of the pricing function could be governed in-part by the *Rating Engine*. The *N - DB Interface* implements popular database drivers in order to send the *charge records* which contains the monetary value for the usage of a particular resource by the consumer, for secure storage in *Charge Records DB*. These records could be retrieved in future for further processing by other RCB modules.

### 4.1.3 Billing Module

Figure 6 describes the billing module that can be implemented as an independent package running on a separate node while interacting with other nodes using standardized interfaces.

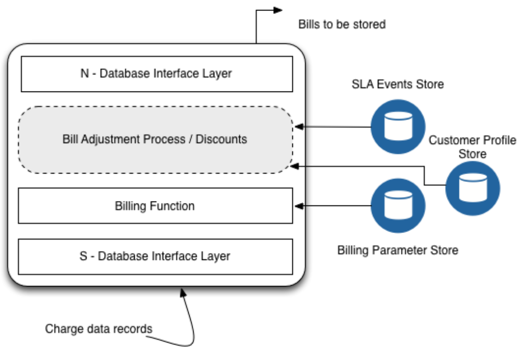


Fig. 6: Billing Submodule

The *S-Database Interface* implements the popular API drivers for connecting to the *Charge Records DB* and retrieving the data records from it. The charge records are aggregated by the *Billing Function* module, which simply generates the basic billed amount for various resources consumed. Depending on the individual consumer profile, the billed amount may need to be readjusted depending on pending discounts, penalties due to SLA violations, etc. This is taken care of by the *Billing Adjustment* process. The northbound database interface implements popular database API drivers for storing the generated bills in a secure *Bills DB*.

### 4.1.4 User / Management Interface

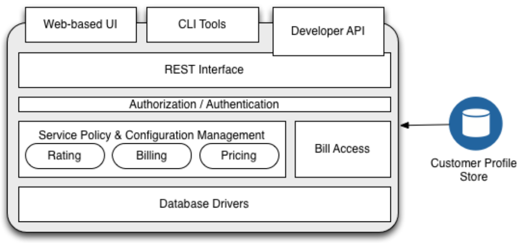


Fig. 7: User-Interface Submodule

Figure 7 shows in detail the user-interface module of the overall RCB architecture. It could provide multiple means of access to the service user: a web based UI, command line interface, and/or developers' kit in the form of an API - each built upon the underlying RESTful [14] interface. A standardized OCCI [15] billing interface could also be implemented to support interoperability. All user requests coming through the REST interface must go through authentication / authorization checks. Normally this module would allow service users to configure all aspects of the RCB process including policies and settings of rating engine, charging strategy, pricing model to be used for various resources consumed. It also allows them to access the generated bills to be forwarded to the collection centers or payment gateways. The interface presented to the service user would be governed by their profile settings.

#### 4.1.5 Supporting Services

Authentication / Authorization service will be implemented as a cross module facilitator since every module in the RCB architecture needs proper authorization to talk to other modules of the service. There are numerous authentication and authorization solutions that could be utilized [16].

Individual modules deployed in a distributed environment needs a common messaging platform to synchronize the processes. Several open source messaging solutions (RabbitMQ [17], ZeroMQ [18], ActiveMQ [19]) could be utilized to implement the RCB messaging service.

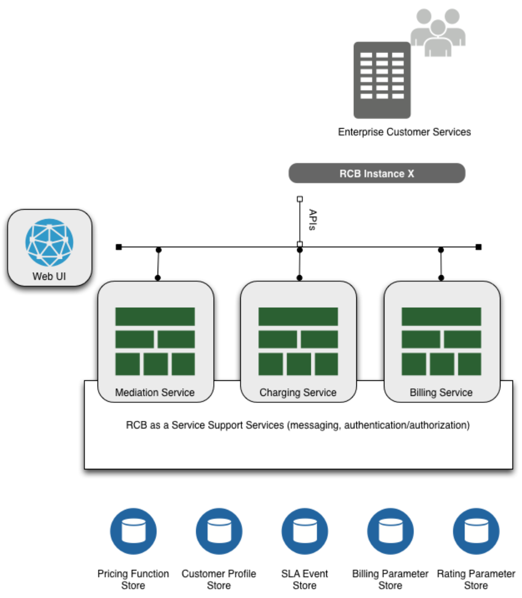


Fig. 8: RCB Overall Service Orchestration

Figure 8 shows the rating-charging-billing overall organization consisting of all supporting services and essential modules that could be easily distributed and made highly available if needed.

### 4.2 Strategies for RCB as a Service

Now that we have seen all the components of the generic RCB software architecture, how can it be used to support *as a service* concept? There are two possible strategies - individual instances per tenant which would require bringing up separate VM/OS-Container instances running all the above mentioned modules along with completely separate backend data-stores. In such a situation, the service user will have an instance isolated from other instances. The other solution would be to offer RCB instance by splicing the overall service. Different user's configurations and policies would be stored in the overall configuration and policy stores segregated using strict access control. The same would hold true for data-stores too. They could be offered out of the same database server or any data-store back-end to different users.

Whenever a new tenant is created, a management end point could be returned back to the user that would allow them to configure all the stages of the full financial process thus offering maximum level of tenant control. The service users could be billed in a numerous manner, ex. number of bills generated each month, metrics ingress rate, etc.

#### 4.2.1 MCN Overall Architecture Alignment

The proposed RCBaaS fits nicely in the overall MCN service architecture. One would have to implement a SM representing the entry point of the RCB service. For each new tenant, a SO instance will be provisioned. The SO will then handle the deployment and run-time management of the RCB instance for that tenant.

## 5. Ensuring High Availability

The RCB overall architecture separates data from program logic. The data is stored in database files, stores and tables. The program logic is provided by software submodules, a web user interface and other control elements.

"High Availability" (HA) architectures exploit the fact that data is separated from program logic in IT processes. They make IT processes highly available by using a clustering technology (for increasing availability of data) and a distributed program logic (for automated failover). RCB can be turned into a HA system by clustering RCB data and by using a distributed program logic to control the RCB services.

HA clustering technology is based on replication and distribution of data on several redundant machine nodes (which form the cluster). In order to keep data consistent, it must be synchronized between all cluster nodes.

The distributed program logic is achieved by running distributed failover software over redundant computer nodes and by allowing the software to control the processes that run on each computer.

### 5.1 Degree of availability

Redundancy is the essence of High Availability. A non-redundant RCB architecture can not ensure high availability levels. If one of the RCB services fails, the whole RCB platform fails too. Though the usage of data replication and distributed failover software does enhance availability of the RCB system (compared to usage of non-redundant data and IT services), the actual degree of availability vastly depends on the number of cluster nodes and the configuration of the failover and data clustering technologies. At this point we must restrict our analysis to the very generic RCB High Availability architecture we see in figure 9. The architecture whose implementation produces a particularly high availability level can only be evaluated by exploration and tests of actual implementations. In the following subsections we want to describe the details of the generic RCB HA architecture.

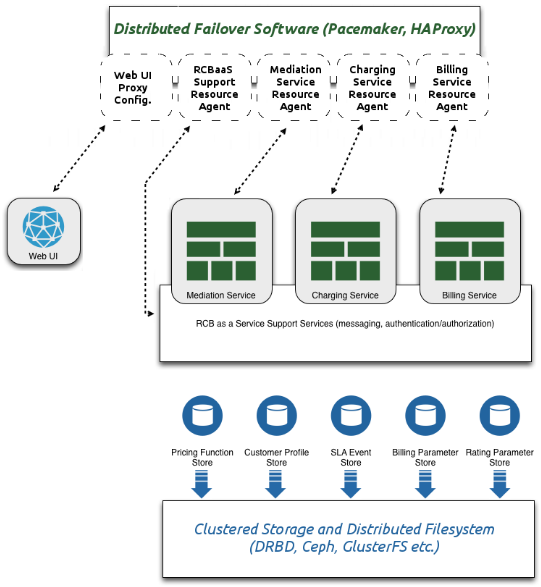

Distributed Failover Software (Pacemaker, HAProxy)
Web UI Proxy Config.
RCBaaS Support Resource Agent
Mediation Service Resource Agent
Charging Service Resource Agent
Billing Service Resource Agent
Web UI
Mediation Service
Charging Service
Billing Service
RCB as a Service Support Services (messaging, authentication/authorization)
Pricing Function Store
Customer Profile Store
SLA Event Store
Billing Parameter Store
Rating Parameter Store
Clustered Storage and Distributed Filesystem (DRBD, Ceph, GlusterFS etc.)

Fig. 9: Generic HA architecture for Rating, Charging, Billing

As shown in figure 9, the first HA component is the distributed failover software (top component) which uses resource agents to monitor execution of the core RCB services. The second HA component are a distributed storage device and file system (bottom component) which redundantly stores the different RCB stores.

## 5.2 Clustering Technology

HA clustering technologies typically *use redundant storage devices* (e.g. disk partitions), *federate them into a single HA cluster* and *create some kind of distributed storage on top of the cluster*. The HA clustered devices are then threated by each node as if they were one single storage device. In order to access the clustered device through a single entry point, usually the cluster gets labelled with a "virtual" IP address. The virtual IP is a an IP address which is shared between cluster nodes and assigned to the cluster node which is currently actively managing the clustered device.

In order to make RCB data highly available, it must be stored on a HA clustered storage. Therefore all database files, stores and tables of the RCB must be transferred to the clustered storage. Then configuration files must be changed in order to locate the data in the clustered storage. In the RCB HA architecture diagram (figure 9) the transfer of RCB data to a clustered storage is depicted with blue arrows.

## 5.3 Distributed Program Logic

While data can be made highly available by replicating it over several nodes, availability of a software program depends not only on redundant data, but also on uninterrupted execution of services that operate with the data. This can be achieved by *executing the services that constitute the RCB platform redundantly on several nodes* and *switching the control flow to those services which are currently available* in case of failure. These tasks must be performed by *a distributed application which monitors and controls execution of RCB services*. Such an application should track the execution state of RCB services and direct the control flow of the RCB platform only to available services.

Program logic of the RCB application can be made highly available by installing identical RCB component services on multiple nodes and deploying a distributed failover software on all nodes. The redundant RCB services are then connected via "proxy configurations" or "resource agents" to the distributed failover software. The resource agents and the proxy configurations allow the distributed failover program to control execution of RCB services. The distributed failover application will check if an RCB service fails and process failover actions to recover from service outages. In figure 9 the connections between RCB services and resource agents or proxy configurations are depicted with dotted arrows.

Candidate technologies that allows us to create a HA RCB platform is covered in section 6.

## 6. Technology Specifics, General Concepts and Related Work

## 6.1 Telecom and 3GPP

For wireless telecommunication charging is mainly covered by 3GPP specifications of 32.x series. These are 3GPP TS 32.240 "Charging architecture and principles" [20] and 3GPP TS 32.299 "Diameter charging application" [21]. The 3GPP standard supports both *offline* as well as *online* charging models. In offline charging, the resource usage is reported from the network to the Billing Domain (BD) after the resource usage has occurred. In online charging, a subscriber account, located in an online charging system, is queried prior to granting permission to use the requested network resource(s). 3GPP and Diameter approach to RCB is not suitable for a mash-up, composed service as the usage demands significant integration with the offered service which is time consuming. A more loosely coupled approach is needed which is offered by our model.

## 6.2 Research Trends in RCB

In this section we describe the existing approaches for RCB in the context of telcos, cloud providers and service providers. In the domain of the 3GPP telecommunication networks, a study by Grgic et al. [22] offers an extensive overview of the charging process. The authors propose: (a) signaling aspect, (b) inter-domain aspect and (c) service- and component-based aspect of online charging with respect to information utilization. They diagnose a lack of information specification and structuring, sharing issues and user privacy issues as research challenges for online charging systems with respect to the information access. A good report that classifies the pricing schemes for IP and ATM networks is presented in [23]. It analyses the current issues, advantages and disadvantages in the both pricing models and compares the pricing approaches. In his paper [24], Kelly describes a system model of charging, routing and flow control for broadband multiservice networks. The system assignees utility functions to the users and capacity constraints to the network. An example shows how the fairness criteria are associated with a particular utility function. The authors demonstrate that when users' choices of charges per unit time and the network's choice of allocated rates are in equilibrium, a system optimum is achieved. A new E-Charging API was proposed to Parlay and 3GPP OSA [25]. This API isolates the charging as a separate process offered by the Payment Service Provider and it is addressed for both the application service providers and the network operators. It permits the application service providers reach to the subscriber base of the payment service provider. Koutsopoulou et al. [26], propose a platform addressed for the next generation mobile network, for sophisticated and reconfigurable support of charging, accounting and billing process (CAB) as a discrete service. This platform reuses the existing network components according to the recommendations of the standardization groups. Apart from one stop billing, it supports separation of charging events based on transport, service and content usage. A set of APIs is provisioned for pricing related reconfigurations and deployment of charging services.

The emerging cloud computing market opens new possibilities for the telecommunication companies to maximize their revenue. In this context, Tselios et al. point out the closed-garden mentality of the telecommunication companies, their slow business model adoption and the lack of credibility to be a major handicap for cloud infrastructure adoption. A charging and billing layer is required, according to the authors, to capture the traffic records and provide the necessary charges to both departmental and individual levels. They conclude the urge for a new business model for increased price competition and improved customer service that will most likely enforce cloud adoption by the telcos [27]. The CGI group [28], identifies flexibility in billing as the missing link for cloud providers that would allow them to aggregate data and to understand usage patterns for better capacity planning and analysis of sales and marketing. We identified in the literature, an example of a system implementation of a model for deployment of different cloud business models based on the Internet Economics process [4]. The authors use jBiling as an accounting platform and IPDR protocol to fit different pricing schemes and tariffs, as well as better accounting on the usage of cloud services. Deelman et al. [29], take an approach towards using the cloud for science and analyze how a scientific application, given the availability of the clouds, can make the right cost-performance trade-off. In this context they study the cost of various workflow execution models and provisioning plans for cloud resources and prove the cloud to be a cost-effective choice for data-intensive applications. A monetary-based incentive for accounting and billing in Grid networks is presented in [30]. The novelty in this model is the support for multiple virtual organizations and multiple network operators. In addition, the authors present a Grid Economic architecture as a solution to the Large File Transfer problem, that is essentially scalable, efficient and feasible over the Internet. Besides the standard AAA and Billing components, this architecture provides a Pricing, Metering and Security elements based on a price-wise or trust-wise service provisioned by the Grid node.

Two interesting applications of online charging are presented by Zuber [31] and Nagahara et al. [32]. The first one is a patented method for automatic tagging of documents and communications with filing and billing information for online social networks. This information can be further associated with each document and the communication can be customized to include categories most applicable to the business of the user. The second invention resolves the conventional charging system's limitation by facilitating children to use imaginary accounts for accessing on-line services such as: on-line shopping and video-on-demand. Finally Bhushan et al. [33], present a standardization-based work in B2B environment–a federated accounting management architecture for charging and billing.

The described approaches address particular segment or the entire RCB process, from individual provider's point of view. RCB paradigm is yet a novel service for cloud providers and therefore migrating this service to a higher level that will embrace heterogeneous providers and services is currently a challenging process. We have not registered so far a generic RCB solution aimed for composed services. What also distinguishes our approach in the RCB domain, is the consideration of high availability concept that is tightly coupled and highly important aspect for services' scalability.

## 6.3 Technologies for enabling HA

For highly available architectures, two type of enablers exist: technology for HA of data and technology for HA of software programs. A storage clustering technology is needed in order to make RCB data highly available. Typical examples of such HA clustering technologies are DRBD [34], Ceph [35] and GlusterFS [36]. The difference between these HA clustering technologies lies in synchronization of the clustered devices.

DRBD is a replicated copy of disk contents: after an initial synchronization of disk contents, disk writes can be performed by "primary" nodes only, and are propagated synchronously to all nodes [37]. DRBD is quite a simple and reliable mechanism, but because storage is rather copied than shared, DRBD storage does not scale very well.

Ceph is a more scalable solution, because (unlike in DRBD) consistency conditions can be relaxed and file writes need not be propagated synchronously to all nodes [35]. A major drawback of such relaxed replication is that a lookup service is needed in order to retrieve files and keep file data consistent [35]. Therefore Ceph separates file metadata from file contents.

An alternative to Ceph could be GlusterFS. Unlike Ceph, GlusterFS uses completely synchronous replication of files. Files are retrieved by a hash value which is assigned to them when they are written [38]. GlusterFS does not scale as good as Ceph, but it still scales better than DRBD, because the GlusterFS storage is not merely a copy of disks, but an abstraction of hardware devices [38].

The choice of the "right" clustering technology for RCB data depends on the requirements of the concrete RCB implementation: if a scalable solution is needed, Ceph is the "best" choice. If reliability is important, DRBD should be taken. If some compromise between scalability and reliability is needed (which is often the case in mobile networks), GlusterFS might be an adequate solution. RCB program logic can be made highly available by using a distributed failover software which manages the RCB services (e.g. the Billing submodule). Typical technologies which are capable of management of IT services are Pacemaker [39] and HAProxy [40].

Pacemaker is a distributed application which monitors "resources" (IT services) in a cluster and controls execution of these resources [39]. In contrast to Pacemaker, HAProxy is a HTTP/TCP load balancer which can detect service failures, failover unavailable services, and redirect user requests to currently available services [40]. HAProxy has the advantage that it couples user interactions to availability of IT services and transparently hides IT service outages to end users. A drawback is that it is mainly designed for HTTP/TCP-based applications. Pacemaker is more flexible than HAProxy: it can manage almost all possible kind of IT services. The major drawback of pacemaker is that it is not actively managing user requests.

As a HA solution for the RCB program logic, Pacemaker is suitable for management of core RCB services like Mediation, Charging, Billing and RCBaaS Support services. Pacemaker offers the flexibility which is required to manage those component services. For the Web UI service, it is better to use HAProxy because of its user request management and load balancing capabilities. Pacemaker and HAProxy have in common that they must be configured in order to observe availability of IT services and perform the required failover actions. In Pacemaker we need "resource agents" which tell Pacemaker how it can monitor, start or stop a particular service. In HAProxy a "proxy configuration" must be defined for each service which is monitored. For the connection of services with Pacemaker and HAProxy we plan to use custom resource agents for the RCB services (Mediation, Charging, Billing and RCBaaS Support) and a proxy configuration for the Web UI.

## 7. Conclusion

In this paper we have provided a highly available, generic RCB service architecture that supports a standard accounting model. Every functional element of the proposed model is configurable, thereby making our solution work in any business environment. As a result, this solution is ideal to be offered as a service to different service providers. In section 2, we provided an overview of accounting process to inspire the design. We presented a detailed architecture in section 4 and analyzed and prepared a high-availability strategy in section 5.

The next steps for us is to implement the architecture and integrate the solution over our OpenStack testbed. We will do more in depth study of different business models and check if our solution satisfies the RCB need of the control group.

## Acknowledgment

The authors would like to thank the MCN consortium for providing the draft of the overall architecture diagrams which helped them a lot to understand the RCB interaction with other components and in one way or other helped shape the RCB generic architecture. The work is supported by the European Community Seventh Framework Programme (FP7/ 2001âĂŞ2013) under grant agreement no.318109.

## References

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST special publication, vol. 800, p. 145, 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] V. Fajardo, J. Arkko, J. Loughney, and G. Zorn, "Diameter Base Protocol," RFC 6733 (Standard Track), Internet Engineering Task Force, Oct. 2012. [Online]. Available: http://tools.ietf.org/html/rfc6733

[3] C. Rigney, "RADIUS Accounting," RFC 2866 (Informational), Internet Engineering Task Force, June 2000, updated by RFCs 2867, 5080. [Online]. Available: http://www.ietf.org/rfc/rfc2866.txt

[4] I. Ruiz-Agundez, Y. K. Penya, and P. G. Bringas, "Cloud computing services accounting," International Journal of Advanced Computer Research (IJACR), pp. 7–17, 2012.

[5] ——, "A taxonomy of the future internet accounting process," in Proceedings of ADVCOMP 2010 : The Fourth International Conference on Advanced Engineering Computing and Applications in Sciences. Florence, Italy: IARIA, 25–30 October 2010, pp. 111–117, iSBN: 978-1-61208-000-0.

[6] O. Community. (2013, July) Openstack open source cloud computing software. [Online]. Available: http://www.openstack.org/

[7] O. Ceilometer Community. (2013, July) Ceilometer - openstack. [Online]. Available: https://wiki.openstack.org/wiki/Ceilometer

[8] S. Cotton, B. Cockrell, P. Walls, and T. Givoly, "Network Data Management - Usage (NDM-U) For IP-Based Services. Service Specification - Cable Labs DOCSIS 2.0 SAMIS," IPDR Service Specifications NDM-U, Nov 2004.

[9] D. Josephsen, Building a Monitoring Infrastructure with Nagios. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007.

[10] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: Design, implementation and experience," Parallel Computing, vol. 30, p. 2004, 2003.

[11] P. Tader, "Server monitoring with zabbix," Linux J., vol. 2010, no. 195, July 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1883478.1883485

[12] DataHero. (2013, July) DataHero. [Online]. Available: http://www.datahero.org/

[13] Quantopian. (2013, July) Quantopian Community. [Online]. Available: https://quantopian.com

[14] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, 2000, aAI9980887.

[15] A. Edmonds, T. Metsch, A. Papaspyrou, and A. Richardson, "Toward an open cloud standard," Internet Computing, IEEE, vol. 16, no. 4, pp. 15–25, 2012.

[16] O. Keystone Community. (2013, July) Keystone - openstack. [Online]. Available: https://wiki.openstack.org/wiki/Keystone

[17] J. Russell and R. Cohn, Rabbitmq. Book on Demand, 2012. [Online]. Available: http://books.google.ch/books?id=uO7IMgEACAAJ

[18] P. Hintjens, ZeroMQ: Messaging for Many Applications. O'Reilly Media, 2013. [Online]. Available: http://books.google.ch/books?id=TxHgtl_sFmgC

[19] B. Snyder, D. Bosanac, and R. Davies, ActiveMQ in Action. Greenwich, CT, USA: Manning Publications Co., 2011.

[20] "Telecommunication management; charging management; charging architecture and principles," 3rd Generation Partnership Project (3GPP), 650, route des Lucioles, Sophia-Antipolis 06921, France, Tech. Rep., 2013, TS 32.240, Rel-12. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/32240.htm

[21] "Telecommunication management; charging management; diameter charging applications," 3rd Generation Partnership Project (3GPP), 650, route des Lucioles, Sophia-Antipolis 06921, France, Tech. Rep., 2013, TS 32.299, Rel-12. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/32299.htm

[22] T. Grgic and M. Matijasevic, "An overview of online charging in 3gpp networks: new ways of utilizing user, network, and service-related information," International Journal of Network Management, vol. 23, no. 2, pp. 81–100, 2013. [Online]. Available: http://dx.doi.org/10.1002/nem.1816

[23] S. Bodamer, "Charging in multi-service networks," http://www.ikr.uni-stuttgart.de/Content/Publications/Archive/Bo_IB29_29702.pdf, 1998.

[24] F. Kelly, "Charging and rate control for elastic traffic," European Transactions on Telecommunications, vol. 8, no. 1, pp. 33–37, 1997. [Online]. Available: http://dx.doi.org/10.1002/ett.4460080106

[25] K. Luttge, "E-charging api: outsource charging to a payment service provider," in Intelligent Network Workshop, 2001 IEEE, 2001, pp. 216–222.

[26] M. Koutsopoulou, A. Kaloxylos, and A. Alonistioti, "Charging, accounting and billing as a sophisticated and reconfigurable discrete service for next generation mobile networks," in Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th, vol. 4, 2002, pp. 2342–2345 vol.4.

[27] C. Tselios, I. Politis, V. Tselios, S. Kotsopoulos, and T. Dagiuklas, "Cloud computing: A great revenue opportunity for telecommunication industry," in FITCE Congress (FITCE), 51st, 6, Poznan, Poland, 2012.

[28] CGI, "Billing in the cloud: The missing link for cloud providers," http://www.cgi.com/files/white-papers/billing-in-the-cloud-e.pdf, 2010.

[29] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: The montage example," in High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for, 2008, pp. 1–12.

[30] S. Kotrotsos, P. Racz, C. Morariu, K. Iskioupi, D. Hausheer, and B. Stiller, "Business models, accounting and billing concepts in grid-aware networks," in Networks for Grid Applications, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, A. Doulamis, J. Mambretti, I. Tomkos, and T. Varvarigou, Eds. Springer Berlin Heidelberg, 2010, vol. 25, pp. 27–34. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11733-6_4

[31] T. ZUBER, "Method for tagging documents and communications with filing and billing information," Patent Application, 05 2011, uS 2011/0106679 A1. [Online]. Available: http://www.patentlens.net/patentlens/patent/US_2011_0106679_A1/en/

[32] J. Nagahara, T. Nashida, H. Nakano, M. Niijima, Y. Sonoda, and Y. Kumagai, "Charging system in interactive on-line service," Patent, 04 2007, eP 0725376 B1. [Online]. Available: http://www.patentlens.net/patentlens/patent/EP_0725376_B1/en/

[33] B. Bhushan, M. Tschichholz, E. Leray, and W. Donnelly, "Federated accounting: service charging and billing in a business-to-business environment," in Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on, 2001, pp. 107–121.

[34] L. Ellenberg, "Drbd 9 and device-mapper: Linux block level storage replication," in Proceedings of the 15th International Linux System Technology Conference, 2008.

[35] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: a scalable, high-performance distributed file system," in Proceedings of the 7th symposium on Operating systems design and implementation, ser. OSDI '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 307–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=1298455.1298485

[36] E. B. Boyer, M. C. Broomfield, and T. A. Perrotti, "Glusterfs one storage server to rule them all," Los Alamos National Laboratory (LANL), Tech. Rep., 2012.

[37] F. Haas, P. Reisner, and L. Ellenberg, "The drbd user's guide," LINBIT HA Solutions GmbH, 2011.

[38] R. Hat, "Glusterfs: Red hat storage software appliance," 2011.

[39] M. Schwartzkopff, Clusterbau: Hochverfügbarkeit mit Pacemaker, Openais, Heartbeat und LVS. O'Reilly, 2010.

[40] V. Kaushal and V. Kaushal, "Autonomic fault tolerance using haproxy in cloud environment," International Journal of Advanced Engeneering Sciences and Technologies, vol. 7, 2010.

# Exploring Security and Privacy Risks of SoA Solutions Deployed on the Cloud

Abdullah Abuhussein, Harkeerat Bedi, Sajjan Shiva
Department of Computer Science
The University of Memphis
Memphis, USA
{bhussein, hsbedi, sshiva}@memphis.edu

*Abstract*— **It has been widely accepted that service oriented architecture (SoA), has been a promising approach for business development and growth. SoA principles (also known as SoA qualities) attempt to guide development, maintenance, and usage of the SoA. These principles provide benefits like: ease of reuse, service automation, and lowering integration costs. However, they can also lead to security issues. These issues are augmented especially when SoAs are deployed in multi-tenancy third party clouds. SoA has benefited from the existence of cloud computing (CC) as it provided SoA with a flexible deployment medium. However, the advantageous collaboration of SoAs and CC has led to a larger set of privacy and security issues (e.g. compliance issues, QoS issues). Additionally, we observe newer kinds of security and privacy risks that are now required to be monitored and mitigated. In this paper we highlight the security and privacy challenges associated with the utilization of the SoA principles on cloud based solutions. We identify the origin and severity of these issues followed by several recommendations to guide the utilization of SoA principles in off-premise clouds.**

***Keywords— service oriented architecture, cloud computing, security, privacy.***

## I. INTRODUCTION

Service oriented architecture (SoA) has provided the software development industry with flexibility and capabilities like bridging business and IT, lower cost by implanting reusability and providing autonomy in software services. SoA is defined as a set of architectural tenets for building autonomous yet interoperable systems [1]. SoA defines eight principles that guide its development, maintenance, and usage. These principles are: abstraction, autonomy, composability, discoverability, formal contract, loose coupling, reusability and statelessness [2].

SoA principles offer a number of advantages (e.g. reusability, reduce integration and maintenance costs) [3] and therefore they can also be represented as qualities of SoA. SoA principles played a significant role in the adoption of the SoA paradigm in the last decade [4]. The tightly coupled nature among services in systems preoccupied developers' minds. The SoA principles alleviated these issues and enabled the software developers to produce software components that are reusable, autonomous and customizable.

In some cases, SoA principles like abstraction and independency of services help to reduce services exposure to the outside world and therefore reduce security risks. However, SoA security in general remains an issue due to the medium they are deployed on and delivered through.

Deployment and delivery of SoA can be performed using several methods. At present, cloud computing (CC) has become the most prominent means of SoA deployment and delivery. CC provides benefits like resiliency, elasticity and reliability but also raises several security and privacy risks [5]. The combination of SoA and CC together produces a larger set of security and privacy risks. CSA Notorious 9 of 2013 stated that Clouds that share PaaS, SaaS, and IaaS are more vulnerable [6]. This is generally the case when deploying SoA solutions on public clouds. .

The future of SoA is tightly interlinked with CC due to the use of Internet, changing nature of the customers, and the impact of social networking (e.g. sudden high consumer demand/traffic that was not an issue before). To handle such situations that are very common now, SoA needs CC to cater the needs of this newer generation of consumers. Therefore, SoA benefits from CC features like agility, scalability, and reliability to operate and conveniently perform upgrades to meet the consumer's needs.

The current research is primarily geared towards finding the security and privacy issues of SoA [7]. Researchers in [8] and [9] have shown some of the security challenges in deploying SoA in the cloud. In this work we study the relationship between the utilization of the SoA principles and the emersion of security and privacy issues . We also show the origin of these security and privacy issues then provide recommendations on how to secure the deployment. SoA is widely practiced today. Now, most companies are focusing on building services that are independent, can be discovered and requested automatically by consumers, and are able to monitor and manage themselves. However, this requires an extensive effort towards balancing the utilization level of SoA principles, while minimizing exposure to security and privacy risks.

Section 2 explores SoA deployments over the past decade. We will also go over current different form of delivering SoA. In section 3 we illustrate how utilizing SoA principles in the cloud may lead to potential security and privacy vulnerabilities. We show the severity of such risks and describe how they are originated. We also present various recommendations to overcome these risks. Section 4 provides our observations on the presented problem, proposed solution and future work.

## II. SoA Deployment and Delivery

Traditionally, SoA solutions like Customer relationship management (CRM) , Enterprise resource planning (ERP), payroll, etc. were deployed on private machines that lie within the premises of the end user's organization (on-premises) or deployed within the SoA provider's organization (off-premises) and accessed by end users through the Internet. Emergence of CC served to meet developers' increasing demands of infrastructure for their SoA solutions. With the advent of CC, the entities responsible for development of SoA and those of infrastructure became separated. This leads to change to the nature, severity and/or existence of SoA vulnerabilities. It also leads newer kinds of issues and risks that were not present earlier (e.g. governance and compliance issues, etc.) [5].

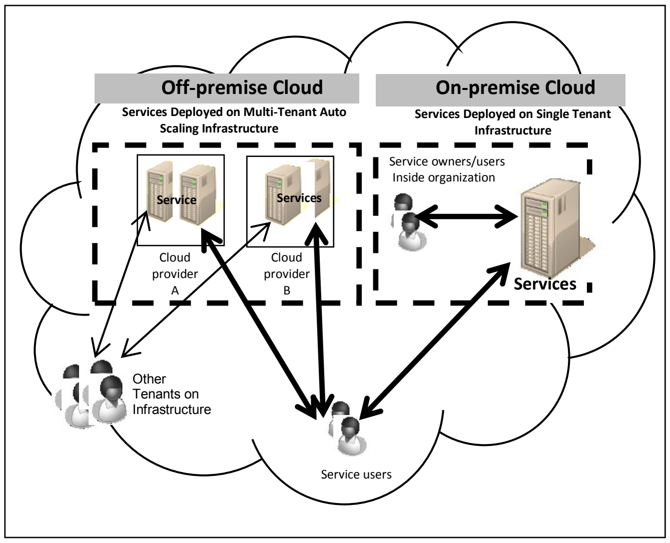

Off-premise Cloud
Services Deployed on Multi-Tenant Auto Scaling Infrastructure
On-premise Cloud
Services Deployed on Single Tenant Infrastructure
Service
Services
Cloud provider A
Cloud provider B
Service owners/users Inside organization
Services
Other Tenants on Infrastructure
Service users

Fig. 1. SoA deployed on off-premises versus on-premises cloud computing

Fig. 1 shows the two possible cases of deploying SoA on the cloud. On the right, SoA is deployed on a cloud model that is on-premise. Services are hosted by the organization's infrastructure and the infrastructure is provisioned and managed by the organization itself. Since the entity responsible for the development of SoA and the infrastructure are the same, the risks are limited.

The case on the left shows off-premise cloud computing infrastructure being used to host the SoA services. Infrastructure is provisioned and managed by the CC service provider. In this case, features like auto-scalability and multi-tenancy are offered to provide SoA developers with as much infrastructure as they need at low costs. However, SoA developers share the infrastructure with other tenants. Also, services might demand more resources and scale up on more VMs on the same physical machine or distant machines on different regions. Moreover, CC service brokers might recommend a different service provider every time additional infrastructure is requested. These scenarios lead to new kinds of security issues and thus risks that were not present before.

CC providers do offer isolated hardware for interested consumers. This in turn would overcome the multi-tenancy drawbacks although at higher prices [9]. Nevertheless, denial of service (DoS) attacks, which are the CC's fifth top threat in 2013 [10], are a serious concern in isolated hardware [11].

## III. Risks of CC on SoA Oriented Solutions and Recommendations

Despite the benefits that SoA principles add to the traditional software development life cycle, they bring new challenges. Some of these challenges are security and privacy issues that take place due to the technologies used in SoA based service development and operation. XML is the core of SoA and is not inherently secure. SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description Discovery and Integration ) are all based on XML. A well-known XML exploit is the XML rewriting attack. Although WS-Security [12], WS-Policy [13] and other standards aim to secure the XML based application and avoid these attacks, the national vulnerability database [14] showed 14 SOAP vulnerabilities, and 4 WSDL related ones in 2013.

Beside the security problems of SoA [15], the fact that CC is becoming one of the most prominent means of SoA deployment worsened matters. Table 1 shows the (8) SoA principles in the first column, application area in column two, alongside the technologies required to foster each one of principles in column three. The forth column highlights how the deployment of SoA in an off-Premise CC can change the nature of the SoA vulnerabilities and the severity of security issues and risks. In the same table, we map these risks to the CSA notorious nine cloud attacks observed in 2013.

Technically, the application of the 8 service-oriented-architecture principles can be segregated into two categories based on the part of SoA that they are utilized in. The first category is for the principles that can be utilized in service contract and registry like: abstraction, discoverability, and formal contract. Other SoA principles like: (composability, Autonomy, loose coupling, reusability, and statelessness) can be utilized in services themselves, which is the second category. We need categorization to enable exploring technical security and privacy issues. For instance, WSDL and UDDI together with SOAP are standards in service registry [16]. Knowing these standards we can look for security breaches that can be exploited using them. Matters can be even worse in an exposed off-premise cloud computing infrastructure.

Below we explain the SoA principles in brief, discuss the security and privacy issues related to the utilization of each principle and suggest several security recommendations.

TABLE I. SECURITY ISSUES RELATED TO THE UTILIZATION OF EACH SOA PRINCIPLE IN OFF-PREMISE CLOUD

| SoA | Applicable to | Vulnera-ble Tech | Risks due to CC [5] | CSA Notorious 9 Threats [6] |
|---|---|---|---|---|
| Abstraction | Service Contract | SOAP UDDI WSDL XML HTTP | 1. Exposure 2. Redundancy and integrity issues 3. Access control 4. Trust | 1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking |
| Discoverability | Service contract and Service Registry | SOAP UDDI WSDL XML HTTP | 1. Exposure to cloud risks 2. Authentication and access control | 1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking |
| Composability | Services | TCP communi-cation XML | 1. Lack of standards 2. QoS. 3. Availability 4.Trust 3. Compliance 4. Governance | 1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking 5.0 Denial of Service 6.0 Malicious Insiders |
| Autonomy | Services | TCP communi-cation XML | 1. Lack of standards and safe patterns 2. Exposure to cloud risks | 1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking 9.0 Shared Technology Vulnerabilities |
| Formal contract | Service Contract | SOAP UDDI WSDL XML HTTP | 1. Trust 2. QoS 3. Authentication and access control | 6.0 Malicious Insiders 5.0 Denial of Service |
| Loose coupling | Services | TCP communi-cation XML | 1.QoS 2.Exposure to cloud risks 3. Data Interception | 5.0 Denial of Service 4.0 Insecure Interfaces and APIs 1.0 Top Threat: Data Breaches |
| Reusability | Services | TCP communi-cation XML | 1. Compliance 2. QoS 3. Exposure to cloud risks 4. Authencation and access control | 6.0 Malicious Insiders 5.0 Denial of Service |
| Statelessness | Services | TCP communi-cation XML | 1.Exposure to cloud risks 2.Compliance 3.Trust | 1.0 Data Breaches 4.0 Insecure Interfaces and APIs 3.0 Account or Service Traffic Hijacking |

## A. *Abstraction*

This principle of SoA aims to hide the logic behind services from the outside world, while providing descriptions in the service contract. To utilize service abstraction service developers need to categorize service meta information into (1) Functional (2) Technology (3) Programmatic and (4) Quality of service categories. Service meta information is then used to create service contracts and service registry. After that, an access control procedure is applied to limit open access to service owners only and give controlled access to others including interested consumers.

Less abstraction indicates more information about the logic of the service and therefore more exposure and more vulnerability. However, over utilization of abstraction indicates over-hiding service logic and technology information and therefore, limits the reusability of the service which leads to creating similar services and raises redundancy and integrity issues.

Another possible problem is access control. Traditionally, service owners have access to all parts of the service, like design specifications, source code, etc. However, the off-premise CC nature exposes the SoA and increases the possibility of account hijacking. For example, an attacker who is successfully able to hijack the account of a service owner will have access to all the parts of a service.

Also as a result of abstraction, service consumers and program designers will not be aware of composite services. Due to this, service consumers won't know what is wrong with the service once a composing part of the whole service goes down as we will see later in the composability principle.

Due to the exposed nature of cloud computing these issues will have a bigger chance to occur. Thus it is recommend that service developers and implementers balance the amount of abstraction and monitor services for risks appropriately. One should look for CC with good access management to mitigate the risk of account hijacking.

## B. *Discoverability*

Service registry is a central repository of service meta data that is hosted on off-premise cloud. Service consumers access service registry to find desired functionalities. That's how a consumer discovers a desired service and then retrieves the service contract. Then the service will be ready for usage.

The discoverability principle enforces that services have communicative meta-data so that they can be efficiently discovered and interpreted. One of the ways this principle is implemented is through using the Web Proxy Auto-Discovery Protocol (WPAD). Browsers in an organization are required to be supplied with the same proxy policies. These polices are created and maintained by using a configuration file based on the Proxy auto-config (PAC) standard. WPAD is used to discover the URL of this configuration file so that proxy policies on all browsers in an organization can be set concurrently.

With SoA being implemented on the cloud, we are adding more exposure to these vulnerable PAC files. Previously an attacker had to be within a company's network to attack these PAC files. Now due to the ubiquitous nature of the cloud, this is not the case. The above two problems become difficult when the service broker comes into picture as this adds another layer of communication exposed to cloud vulnerabilities. In 2012 a summary by the national vulnerability database shows WPAD) functionality in Microsoft .NET Framework 2.0 SP2. WPAD was not validating configuration data that is returned during acquisition of proxy settings. This vulnerability may allow remote attackers to execute arbitrary JavaScript code.

In 2013, the same database reported a Cross-site scripting vulnerability in the UDDI administrative console in IBM WebSphere application server. UDDI is the core of the registry along with WSDL and SOAP [14].

Thus it is recommended to use some form of authentication among services, or between services and the service browser. Also, balance the amount of discoverability, and monitor the services. Another recommendation can be to enable automatic updating for your services to benefit from security frequent patches provided by SoA vendors.

### C. *Composability*

This principle encourages that services become effective participants for composition. It promotes composing new solutions by reusing existing services. However, lack of standards in how to securely and safely compose a service from other services on a cloud is a possible security issue due to the multi-tenancy nature of the cloud. As mentioned before, service contracts hide service composability details so; consumers can never tell whether the service is a standalone service or composed of others.

Availability of the composing services will affect the availability of the parent service. Moreover, quality of the service (QoS) depends on the QoS of the CC infrastructure.

Also, QoS of a composed service depends on the QoS of the sub-services and the infrastructure they are deployed on. Because of the multi region infrastructure of the cloud, compliance and distributed ownership security issues may also apply if the regulations in the countries of the composing sub-services do not match.

Moreover, too composability denotes more transit time due to communication among composing services. Attackers can steal or modify information if not protected while in transit. Again, the exposed nature of the off-premise cloud computing may worsen matters [17].

Therefore, it is recommended to follow safe service composition patterns when composing solutions [8]. It is also essential to review SLA of the underlying CC infrastructure and make sure that hosting countries have no problem with the content and the function of the service. Auditing the underlying CC service for hypervisor security is another recommendation. It helps to overcome multi-tenancy security issues. Encryption and digital signature of data on transit must be considered too in order to secure data in transit. Another recommendation is to balance the amount of composability, and monitor composed services and participant services.

### D. *Autonomy*

This principle of SoA aims to build services with self-control over the logic they contain. When services are made autonomous, they become independent of the underlying technologies, i.e., these services will be resilient to the issues in these technologies. But at the same time, since they can be implemented on more diverse platforms, we are also increasing their exposure to security flaws of these platforms. This will increase the possibility of compromising services due to variations on the underlying technologies.

Service autonomy implies greater emphasis on explicit management of trust between applications to avoid malicious modification and avoid service integrity issues especially due to the nature of public clouds [18].

The Autonomous nature of services implies that services communicate to maintain control over the resources and to coordinate with other components of the SoA. A significant increase in the messaging must occur as service autonomy increase which will also increase exposure to vulnerabilities on off-premise CC. The greater the number of resources, that are accessible for attack, the greater the attack surface and therefore, the more insecure the software environment [19].

The recommendations to overcome these issues are to do a thorough assessment of whether or not it is necessary to increase autonomy at the expense of exposure. It is also important to verify the security practices that can be applied to the underlying technologies. A strong input validation is required to verify input from other applications. Finally, apply WS-Security to achieve trust among autonomous services and applications.

### E. *Formal contract*

When a service is implemented as a Web service, the service contract is normally comprised of a WSDL definition, multiple XML schema, policy definitions, as well as supplementary documents, such as an SLA. This principle enables a standard design of services in terms of policies, WSDL, and XML Schema within the service inventory.

As aforementioned, the formal contract principle is utilized on service contracts. So, it is also subject to WSDL, XML, and SOAP security issues.

In cases where SLA parameter deals with response time and there is a delay, the service consumer would not know whether the problem lies with the service or the CC infrastructure. The service consumer will have to trust the SoA provider for the promised QoS. Moreover, the QoS could get worse if the service is a composed service [20].

Standardization of services within the inventory might give a pattern of how these services are built. This might lead to unveil information about the logic, and/or the technology used to build other services if one service is attacked.

To safely and securely apply this principle we present the following recommendations. The first one is to avoid automated tools when creating contracts as it might lead to inaccuracies. Verify the created contracts to make sure that the underlying infrastructure provides the promised QoS by the SLA. A good access control and authentication system is also required here to avoid illegitimate communication.

### F. *Loose coupling*

Loose coupling enforces that services are built in such a manner that they are decoupled from their surrounding environment. Services must be designed in such a way that it is not tightly coupled to other services or resource. Decoupling a service from its environment has several advantages (e.g.

Hiding service implementation from attackers) however; it increases the message exchanges between the service and the environment. Deploying services on CC makes it worse since the messages are transmitted through the Internet which adds latency to response time and reduces throughput. Also, messages passing between two services or between a service and the service container can be intercepted as mentioned before [21].

Thus, it is recommended to use secure communication by applying encryption on transmitted data. Another recommendation is to use compression techniques to reduce the bandwidth and latency overhead.

### G. Reusability

Services need to be as generic as possible so that they are of interest to multiple service consumers, however, larger granularity may lead to larger incompatibilities that might in turn lead to security issues. To utilize reusability, developers need to produce solution in forms of services with the intention of promoting reuse. Compliance issues can rise by producing reusable SoA services [22]. For example, rules and regulations in different countries can limit the extensibility of use of such reusable services. Another issue in enforcing reusability on off-premise deployed SoA is the difference in the CC infrastructure configurations. Different CC providers have difference configurations, thus QoS variance is expected. Also, changes to standards or upgrades applied to infrastructure may have a large impact on the security of services.

Therefore, it is recommended that SoA developers test services on various infrastructure configurations before releasing them to public. As suggested previously, a thorough walkthrough over the rules and regulation of countries hosing the CC infrastructure should be performed. It is also important to verify that the service data is lawful in all stages (input, process, output, and storage).

### H. Statelessness

This principle of SoA promotes minimizing resource consumption by services. This is achieved by deferring the management of state information when necessary. In other words developers should try to avoid service consumption of resources so that services can handle more requests in a reliable manner. Also saving state in an external component requires additional infrastructure. On the cloud, since the external component can be placed anywhere, it becomes necessary to ensure that the latency limits are met. While communicating to and from different clouds, we are exposing the state of the service and increasing the message exchange between the service and its infrastructure.

Thus, similar to the recommendations provided for loose coupling, it is suggested to use secure communication by applying encryption. Also it is recommended to use compression techniques to reduce the bandwidth and latency overhead and thus, increase service availability.

## IV. Conclusion

Service oriented architectures (SoA) and cloud computing (CC) are accelerating to provide consumers with reliable, resilient, and efficient solutions. Increasing the utilization of SoA principles indicates adding more qualities to applications however, it also exposes developed services to newer vulnerabilities. These vulnerabilities can occur due to the broad attack surface of these SoA solutions. In this paper we showed the importance of balancing and monitoring the services that utilize the SoA principles in off-premise cloud computing. We presented several security and privacy risks (challenges). We also provided recommendations that developers of SoA in public cloud computing need to consider to overcome these risks.

In this work, we have demonstrated how enforcing the eight principles of SoA can add risks when deployed on off-premise CC environments. Porting SoA to the cloud will not have only benefits, but it will also add some security risks that developers need to consider. Further investigation is needed on additional security risks and safe SoA patterns to strengthen the SoA industry. A common issue when developing SoA is the overhead of utilizing these principles. In addition, we have seen other security and privacy issues like exposure, QoS, trust, compliance, data interception, and availability. There are general recommendations [23] when porting SoA to an off-premise CC like (1) look for secure CC services which exhibit adequate security attributes [24, 25] to overcome the most possible security issues. (2) Test services on different infrastructures and different scenarios before releasing them to be used by public and (3) encourage SoA developers to find and publish safe SoA development patterns so that others can benefit from them.

We are now investigating the factors that affect the over-utilization of the SoA principles. We also intend to identify safe SoA utilization patterns that can help others in overcoming the security risks presented in the paper.

## References

[1] Jammes, François, Antoine Mensch, and Harm Smit. "Service-oriented device communications using the devices profile for web services." In Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, pp. 1-8. ACM, 2005.

[2] Thomas Erl: SOA Design Patterns, Prentice Hall PTR; 1 edition (2009)

[3] Bean, J. :SOA and web services interface design: principles, techniques, and standards. Morgan Kaufmann. (2009)

[4] Inaganti, S., & Aravamudan, S. (2007). SOA maturity model. BPTrends, April.

[5] NITS, "Guidelines on Security and Privacy in Public Cloud Computing", http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144. pdf

[6] The Notorious Nine Cloud Computing Top Threats in 2013, https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_ Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdF

[7] F5, SOA: Challenges and Solutions- White paper, http://www.f5.com/pdf/white-papers/soa-challenges-solutions-wp.pdf

[8] Wei, Y., & Blake, M.. : Service-oriented computing and cloud computing: Challenges and opportunities. Internet Computing, IEEE, 14(6), 72-75. (2010)

[9] Pal, P., Atighetchi, M., Loyall, J., Gronosky, A., Payne, C., & Hillman, R. : Advanced Protected Services-A Concept Paper on Survivable Service-Oriented Systems. In Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), 2010 13th IEEE International Symposium on (pp. 158-165). IEEE. (2010, May)

[10] Ristenpart, Thomas, et al. "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds." Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009.

[11] Amazon Web Services, EC2: http://aws.amazon.com/ec2/faqs/

[12] WSS: http://www.oasis-open.org/wss/

[13] Web Services Policy Framework (WS-Policy): http://www.ibm.com/developerworks/library/specification/ws-polfram

[14] National Vulnerability Database: http://nvd.nist.gov/

[15] Phan, C. (2007, October). Service oriented architecture (soa)-security challenges and mitigation strategies. In Military Communications Conference, 2007. MILCOM 2007. IEEE (pp. 1-7). IEEE.

[16] García-González, Juan Pablo, Verónica Gacitúa-Décar, and Claus Pahl. "Service registry: A key piece for enhancing reuse in SOA." (2010).

[17] Venkatasubramanian, Nalini. "Safe'composability'of middleware services." Communications of the ACM 45, no. 6 (2002): 49-52.

[18] Cabrera, Luis Felipe, Christopher Kurt, and Don Box. "An introduction to the web services architecture and its specifications." Microsoft, Microsoft Technical Article, Oct (2004).

[19] Manadhata, Pratyusa K., Kamie M. C. Tan, Roy A. Maxion, and Jeannette M. Wing. " An Approach to Measuring a System's Attack Surface." CMU Technical Report CMU-CS-07-146, August 2007.

[20] Bianco, P., Lewis, G. A., & Merson, P.. Service Level Agreements In Service-Oriented Architecture Environments (No. Cmu/Sei-2008-Tn-021). Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst. (2008)

[21] Joshi, Rajive. " Data-Oriented Architecture: A Loosely Coupled Real-Time SOA." Real-Time Innovations, Inc., August 2007.

[22] Is SOA Being Pushed Beyond Its Limits? , from: http://msdn.microsoft.com/en-us/architecture/aa699422.aspx

[23] Microsoft: Security Fundamentals for Web Services, http://msdn.microsoft.com/en-us/library/ff648318.aspx#WebServicesSecurityPrinciples

[24] Abuhussein, Abdullah, Harkeerat Bedi, and Sajjan Shiva. "Evaluating security and privacy in cloud computing services: A Stakeholder's perspective." Internet Technology And Secured Transactions, 2012 International Conferece For. IEEE, (2012)

[25] Softwareag.com. " Best Practices for SOA Governance User Survey." Software AG, Summer 2008.

# Modeling Cloud/Grid Applications using Functional Representations

**Noé Lopez-Benitez**
Department of Computer Science, Texas Tech University, Lubbock, TX 79403, USA

**Abstract**— *Without altering the functionality of the intended flow of execution, a sequence of compositions can be derived from a scientific workflow or directly from the functional code; this sequence can then be used to manage the execution of the workflow in a grid/cloud oriented environment. A cloud execution model is proposed that considers the submission of composition patterns that can be expanded to integrate interface functions involving orchestration steps such as resource matching, location, and communication information. Expanding the functionality of a given language may potentially provide a holistic programmable approach to develop, deploy, and execute complex applications efficiently using grid/cloud resources.*

**Keywords:** Scientific workflows, partitioning, orchestration, grid/cloud applications, execution management

## 1. Introduction

Functional languages are characterized by the ability to express sets of related serialized functions as well as independent functions that can be executed in parallel. Large workflows describe not only a collection of component functions, but also their dependencies, which predefine a constrained order of execution. Scientific workflows can describe not only computational and service requirements but also the location of such services or computational units. Instruments in scientific laboratories, robots in remote inaccessible areas, a satellite unit in outer space, a set of databases, storage units as well as computational units, all provide services that must be orchestrated to satisfy an overall scientific objective. In this paper the feasibility of a high-level composition and orchestration via functional languages is explored. A functional description of workflows is proposed as an alternative abstraction that can provide the basis for a dynamic management of service requests as well as a paradigm to organize and easily develop entire applications via the use of functional languages to explore not only fine-grained parallelisms, but also functional dynamic parallelisms suitable for grid and cloud execution environments.

Functional languages such as Haskell [1], Parallel Haskell [2], and SequenceL [3] have been proposed to specify and/or generate possible parallel operations for fine-grained computational platforms. Haskell has been extended to Cloud Haskell to provide message passing support [4]. The translation of functional code may lead to fine-grain parallelism; sequenceL, for example, generates n-tuples of independent computations that can be mapped into multiple independent threads of execution [5]; consequently, fine-grain parallelisms can be mapped into high-level language such as MCUDA constructs [6] suitable for multi-core execution platforms.

Section II of the paper discusses functional descriptions as alternate representations of workflows. Reduced functional descriptions are addressed in section III. Section IV illustrates the use of alternate sequences of submissions that have the potential to minimize data transfers. A cloud-based execution model is described in section V. Work related is discussed in section VI. The paper concludes in section VII.

## 2. Functional Description of Workflows

A workflow is described as a typical task graph (DAG) by a tuple $(V, E)$ where $V$ is a set of vertices representing individual functional units with data dependencies described by the set of edges $E$ [7]. As dependencies dictate the order of execution it is possible to regard each task as a functional unit depending on the execution of its predecessors. For example, if a task C is preceded by independent tasks A and B in the workflow, then a notation $C[A, B]$ will model not only such dependencies but also that tasks A and B can be executed in parallel. For an effective execution management of tasks in a grid environment, mapping a DAG model into a functional description is reported in [8]. A workflow $W$ can be alternatively described using its functional description as follows:

$$W = \{T_{F1}(), T_{F2}(), \ldots, T_{Fn}()\} \tag{1}$$

Where $T_{Fi}()$ represents a collection of nodes describing a path of dependent functions. If we let $T_{Fi}$ represent a terminal node (function) in the workflow, then each $T_{Fi}$ in (1) can be described as:

$$T_{Fi}[T_x[T_y[, \ldots, ] \ldots]], i = 1, \ldots, n \tag{2}$$

Where $T_x$ and $T_y$ are nodes in one of the execution paths leading to terminal node $T_{Fi}$. Thus, each node in the workflow is expressed as a function of all previous nodes in its execution path. Furthermore, the collection of functions in a single path corresponds to the set of dependent functions that must be executed in sequence and are queued accordingly. Each sequence of functions includes at least one initial task. Examining all paths in equation (2) all common tasks (queue heads) are identified as root tasks that correspond to at least one initial task. Initial root tasks are functions of the initial parameters of the workflow. A set of root tasks corresponds to a set of independent tasks that can be scheduled for execution in parallel.

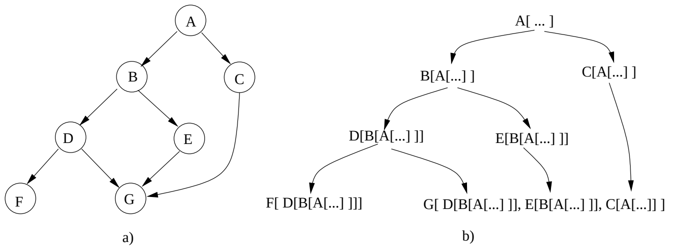


Fig. 1: Mapping a workflow into a functional description

Consider for example a workflow $W$ as shown in Fig. 1a. A functional representation can be derived from the workflow shown in Fig. 1b, where each node is expressed as a function of all previous computing nodes in its path. Grouping the terminal nodes in Fig. 1b, the following functional statement is generated:

$$W = \{\ F[D[B[A]]],\ G[D[B[A]], E[B[A]],\ C[A]]\ \} \quad (3)$$

As this expression shows, W is now described in terms of functions, eliminating the need for explicit representation of data items, except for input and output data.

## 2.1 Dynamic Execution Support

Removing, for readability, the internal brackets from equation (3) each path forms a queue in a set: $\{FDBA,\ GDBA,\ GEBA,\ GCA\}$, where the right most function in each queue corresponds to the Òhead of the queueÓ or the ÒrootÓ function. These structures can be easily obtained applying well-known depth-first search algorithms. Systematically separating functions or tasks to the right identifies those ÒrootÓ tasks that can be submitted for execution as reported in [8]. The model proposed in this paper relies on the generation of compositions, i.e., sets of functions that can be executed in sequence or in parallel. To describe the model, parallel functions are separated by commas; otherwise, a sequential execution is indicated. Using square brackets enforces serialization. Alternate partitions of the workflow described in (3) are generated by extracting to the right all sets of root functions leading to a set of compositions that describe a specific order of submission. The process generates possible representations (partitions) of the original workflow. For example the workflow in (3) can be represented in the following possible ways:

1) $\{FDB, GDB, GEB, GC\}[A]$
2) $\{FD, GD, GE\}[B, C][A]$
3) $[F, G][D, E][B, C][A]$

Note that partition 1) and 2) show the separation of compositions to the right. The remaining tasks in the workflow show the status of the remaining queues. Note also, that when a full sequence of sets of roots is reached, it corresponds to a partition in which sets contain unique components (not contained in any other set) as exemplified by partition 3); this sequence can then be used to manage the order of submissions for execution.

## 2.2 Functional Languages and Extensions

Functional languages can potentially be used to specify complex applications that naturally lead to a functional description of the corresponding workflow. Equation (3) is isomorphic to the workflow represented in Fig. 1a. The idea of generating functional descriptions from functional language code, or even from the profiling of legacy code [9] is still open to investigation. Functional descriptions such as equation (3), expose possible parallel operations for fine-grained computational platforms and can be used to manage execution of functions, or a set of functions, in a service provider execution environment. Consider for example SequenceL, which relies on CSP (Consume-Simplify-Produce) and NT (Normalize-Transpose) semantics to identify all possible parallelisms. The CSP paradigm is based on the generation of a series of tableaus each holding partial results for each step in the evaluation process. The NT paradigm, on the other hand, can lead to the identification of parallelisms using two steps: the normalize step, which is used to expand operands into consistent sizes, and the transpose step which aligns pairs of operands to describe operations that can be performed in parallel. The application of these semantics to a factorial function leads to an optional parallel implementation of a set of product functions:

$$\begin{aligned}
fact(n) &= fact(n-1) * n \\
&\quad when\ n\ >\ 1\ else\ 1 \\
&= fact(n-2) * (n-1) * n \\
&\quad when\ n-1\ >\ 1\ else\ (n-1) * n \\
&= fact(n-3) * (n-2) * (n-1) * n \\
&\quad when\ n-2\ >\ 1\ else\ (n-2) * (n-1) * n \\
&= fact(n-4) * (n-3) * (n-2) * (n-1) * n \\
&\quad when\ n-3\ >\ 1 \\
&\qquad else\ (n-3) * (n-2)(n-1) * n \\
&= \ldots \\
&= 1 * 2 * 3 * * (n-1) * n = prod(1 \ldots n)
\end{aligned}$$

To illustrate the NT semantic let $n = 5$, then pairing operands with the $*$ operator:

$$\begin{aligned}
fact(5) &= prod(1 \ldots 5) = *(*(2,3), *(4,5)) \\
&= *(6,20) = 120
\end{aligned}$$

Furthermore, let $P1 = *(2,3), P2 = *(4,5)$ and $P3 = *(P1, P2)$. A functional description can be used to express a fine-grain workflow where P3 is a product function that depends on product functions P1 and P2. The same expressiveness, not yet exploited, can be used for higher levels of granularity.

## 3. Reduced Functional Represenions

In a grid, the scheduler forwards all incoming functions (tasks) to different execution nodes. Condor's Dagman provides such functionality. Using a functional description of an application, execution can be controlled by the user directly, or, take advantage of suitable middleware such as Condor, or several other web-based tools to orchestrate the execution of the work submitted.

In a cloud-based service environment, a request to a single site could be issued for a pattern of several functions at a time, which is also possible in a cluster/grid computational environment. These patterns may correspond to a sequential or a parallel order of execution. We surmise that a systematic extraction and submission of these patterns may lead to a significant reduction in communication delays. Therefore, a set of rules are proposed to express a reduced functional description in which a sequential or parallel pattern is identified and the overall functionality of the workflow is preserved.

For a general description of sequential and parallel patterns, the notation used assumes that $x_i,\ i = 1, \ldots, n$, identifies the ith node in a pattern with $n$ number of nodes (functions). Again, the use of square brackets enforces a dependency on the execution of whatever composition is enclosed in brackets. The use of parenthesis will be used to express dependencies on data items. Reduction rules SC1 and SC2 refer to embedded sequential patterns, and PC1 and PC2 refer to embedded parallel patterns.

### 3.1 Sequential Composition

This composition describes a functional description of a workflow in which all nodes $x_1$ to $x_n$ must be executed in a sequence identified as follows:

$$[x_n x_{n-1}\ \ldots\ x_2\ x_1] \tag{4}$$

In this pattern $x_n$ is the terminal function in the workflow, and it is dependent on the sequential execution of all the functions in the set $\{x_{n-1}, \ldots, x_2, x_1\}$. As this pattern can also be embedded in large workflows a rule can be applied that leads to a potentially reduced representation:

#### 3.1.1 Reduction Rule SC1

A sequential composition (SC) of functions in the set $\{x_{n-1}, \ldots, x_2, x_1\}$ can be embedded in a functional description such as:

$$x_n[x_{n-1}\ \ldots\ x_2\ x_1], x_n[y] \tag{5}$$

where $y \notin \{x_{n-1}, \ldots, x_2, x_1\}$. Then the following composition is functionally equivalent to (5):

$$x_n[x'_1, y]$$

where $x'_1$ denotes the sequential composition $[x_{n-1}\ \ldots\ x_2\ x_1]$. This equivalence follows by observing that (5) can be submitted in the following sequence:

$$x_n[x_{n-1}\ \ldots\ x_2 x_1, y]$$

#### 3.1.2 Reduction Rule SC2

This rule applies if the sequential composition of the set $\{x_{n-1}, \ldots x_2, x_1\}$ is embedded in a functional description of the form:

$$x_n[x_{n-1}\ \ldots\ x_2\ x_1], y[x_{n-1}\ \ldots\ x_2\ x_1] \tag{6}$$

if $x'_1$ denotes the sequential composition $[x_{n-1} \ldots x_2 x_1]$ and $y \notin \{x_1, x_2, \ldots, x_{n-1}\}$ then the description in (6) is functionally equivalent to the following sequence:

$$[x_n, y][x'_1]$$

This equivalence follows by observing that (6) can be submitted in the following sequence:

$$[x_n, y][x_{n-1}\ \ldots\ x_2 x_1]$$

### 3.2 Parallel Composition

The orchestration of an embedded parallel pattern within a larger workflow will lead to a possible n-degree parallelism just for this structure, provided that the remote site is able to execute up to $n - 1$ functions in parallel. Submitting an embedded parallel pattern seeks to reduce communication delays, but will also reduce the potential dynamic parallelism expressed by additional independent paths in the rest of the workflow. A parallel pattern can appear in two forms: a *join* or a *fork* structure.

A *join* structure in the workflow is identified with the following structure:

$$y[x_n, \ldots,\ x_2, x_1]$$

In this composition $y$ is a terminal function that depends on the possible parallel execution of all functions in the set $\{x_n, \ldots, x_2, x_1\}$. An embedded join structure is generated by separating to the left a common $n$-degree node.

The following pattern identifies a *fork* composition:

$$[x_n, \ldots,\ x_2, x_1]y$$

This composition shows that the set of functions $\{x_n, \ldots,\ x_2, x_1\}$ can be submitted to execute in parallel but only after $y$ (which is not in the parallel set) reports a successful completion. Note that the orchestration of a fork corresponds to a master-slave configuration for a large class of parallel applications and appropriate for execution in a virtual cluster with several nodes.

### 3.2.1 Reduction Rule PC1

A functional description of a workflow given as follows:

$$[x_n, \dots,\ x_2, x_1]z,\ x_k y \tag{7}$$

contains and embedded fork composition $z' = [x_n, \dots,\ x_2, x_1]z$. If $x_k \in \{x_n, \dots,\ x_2, x_1\}$, and $y \notin \{x_n, \dots,\ x_2, x_1\}$ then the following sequential composition is a reduction functionally equivalent to (7):

$$[z'][y] \tag{8}$$

To show that this reduction is valid, notice that (7) can be submitted in two parallel sequences as follows:

$$[x_n, \dots,\ x_2, x_1][z, y]$$

Which indicates that the topology of the workflow allows for the parallel execution of $z$, and $y$ before the set $\{x_n, \dots,\ x_2, x_1\}$ is submitted for execution. Since $z$, and $y$ are independent of each other then they can be executed in sequence without affecting the computational flow and:

$$[x_n, \dots,\ x_2, x_1][z, y] = [x_n, \dots,\ x_2, x_1]zy = z'y$$

This equivalence is also true if $y$ is connected to more than one node, and even to the entire parallel set $\{x_n, \dots,\ x_2, x_1\}$.

### 3.2.2 Reduction Rule PC2

If a functional description of a workflow is given as follows:

$$z[x_n, \dots,\ x_2, x_1], yx_k \tag{9}$$

Then for any pair $x_k \in \{x_n, \dots,\ x_2, x_1\}$ and $y \notin \{x_n, \dots,\ x_2, x_1\}$, the following sequential composition is functionally equivalent to (9):

$$y[z'] \tag{10}$$

Where $z' = z[x_n, \dots,\ x_2, x_1]\ x'_i$ denotes the join composition shown in (9). Since $y \notin \{x_1, x_2, \dots, x_n\}$, then a parallel sequence is given as follows:

$$[y, z][x_n, \dots,\ x_2, x_1]$$

If the join composition $z'$ is used, the sequence becomes:

$$[y][z][x_n, \dots,\ x_2, x_1] = [y][z']$$

Which serializes $z'$, and $y$, but since they are independent, the functionality does not change and the composition in (10) holds.

## 4. Dynamic Data Alignment

Submitting composition sets implies that input data, if needed, must also be attached; the output data returned is likely to be used by the next composition to be submitted. Consider, for instance, the workflow shown in Fig. 2.

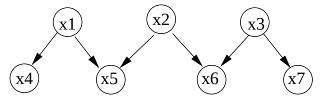


Fig. 2: An interactive workflow

The workflow in Fig. 2 exhibits the following set of paths:

$$\{x_4x_1, x_5x_1, x_5x_2, x_6x_2, x_6x_3, x_7x_3\} \tag{11}$$

Aligning the first set of root functions for a parallel execution leads to the following sequence of submissions:

$$[x_4, x_5, x_6, x_7][x_1, x_2, x_3] \tag{12}$$

These compositions show two groups of parallel tasks which, as reported in [15], can be submitted to minimize transfer delays. At this point, no information as to data dependencies in the workflow is available. If the second submission happens to be sent to the same remote site, data dependency information needs to be forwarded to this site to maintain computational integrity. If data is needed in a different site, both data and dependency information need to be forwarded to that remote site. I/O data dependencies can be addressed by using the dependency information shown by the queue structures in (11), and supplementing the submission information in (12), with the input data and data flow generated. Thus, the full description of (12) is given as:

$$\begin{aligned}(o_4, o_5, o_6, o_7) \quad \leftarrow \quad & [x_4(d_1), x_5(d_1, d_2), x_6(d_2, d_3), x_7(d_3)] \\ & [x_1(i_1), x_2(i_2), x_3(i_3)]\end{aligned} \tag{13}$$

Where $d_j \leftarrow x_j(i_j)$ associates the input data $i_j$ to the original root function $x_j$ in the workflow, which in turn, generates a data item $d_j$ that will be needed by some function in the next submission. The output $o_j$ is generated by the terminal node $x_j$. Note that the parenthesis used in (13) indicates the dependencies on the data generated. In the case of terminal functions this data is sent to the user.

Consider now an alternative functional description as follows:

$$[x_4, x_5]x_1, x_5x_2, x_6x_2, [x_6, x_7]x_3 \tag{14}$$

This description shows two fork structures from nodes $x_1$ and $x_3$. By applying rule PC1 the following compositions are derived:

$$[x_4, x_5]x_1x_2, [x_6, x_7]x_3x_2 = [x_1', x_3']x_2 \tag{15}$$

Integrating data requirements as before results in the following sequence:

$$(o_4, o_5, o_6, o_7) \leftarrow [x_1'(d_2), x_3'(d_2)]x_2(i_2)$$

Compared to the composition in (13) the only intermediate data transfer requirements is the one generated by $x_2$.

Consider once more the set of paths for the workflow $W$ shown in Fig. 1b.

$$\{FDBA, GDBA, GEBA, GCA\} \tag{16}$$

As root functions are separated the right the following partition (partition 2 shown in section 2.1) identiies a fork compostion:

$$\{FD, GD, GE\}[B, C][A]$$

In this partition the fork composition consumes the data generated by $A$, and produces the data generated by $B$ and $C$. I/O data can be integrated as follows:

$$\{FD, GD, GE\}(d_B, d_C)[B, C][A](i_A)$$

The remaining queues show an embedded join composition:

$$\{FD, GD, GE\} = \{FD, G[D, E]\}$$

Applying rule PC2 results in the sequence of submissions:

$$FG[D, E][B, C][A] = FG'A'$$

where $G' = G[D, E]$ and $A' = [B, C]][A]$. Observe that the join $G'$ consumes $d_B$ and $d_C$ but must return $d_D$ which is consumed by $F$. The overall submission sequence can be described as follows:

$$(o_F, o_G) \leftarrow F(d_D)G'(d_B, d_C)A'(i_A)$$

Finally, a set of data outputs delivered to the user is given by the set $(o_F, o_G)$ which corresponds to the outputs, if any, provided by the terminal nodes $F$ and $G$. Submission sequences are not unique as they depend on the topology of the workflow and the applied reduction criteria.

## 5. A Cloud-based Execution Model

Using the functional description abstraction of an application, a composition can be submitted one a time to a web interface tool to generate a concrete model of execution for that composition. If the tool requires the use of a specific language then the functional description can be used to generate the corresponding code.

Compositions can be extended and integrated into a single model to include orchestration information and deliver requests from the user machine directly to the cloud site. Fig. 3 describes a preliminary model in which three possible stages are involved in the cloud execution of a workflow:

1) A *Functional Description Abstraction* stage provides a functional description of the entire workflow, generated from the workflow graph, or directly from a functional code of the application.
2) A *Web-based orchestration* stage is intended to work with each submission (one composition at a time, or the entire workflow description), to generate the necessary requests to the already identified cloud sites.
3) A *Cloud Service* stage will process requests to execute the compositions received; individual tasks will be scheduled using local criteria. Each server will return the expected output for each composition executed.

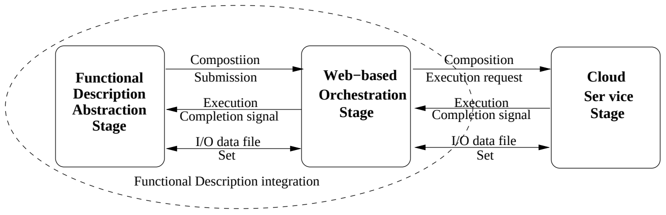

Functional Description Abstraction Stage
Compostiion Submission
Execution Completion signal
I/O data file Set
Web−based Orchestration Stage
Composition Execution request
Execution Completion signal
I/O data file Set
Cloud Ser vice Stage
Functional Description integration

Fig. 3: Stages of a Cloud Execution Model

At the user site, one composition can be submitted at a time; each composition set may contain a single function (task), a sequential, or a parallel composition. The web-based orchestration stage locates a cloud site, formulates and sends the execution request along with any input data required. At the cloud service stage, a composition is submitted to the physical resources, an execution completion signal is generated, which is sent back along with any output data generated. The web-based stage relays a completion signal and output data files to the submitting machine.

If one function at a time is submitted, full management of the workflow remains at the user site; if the composition submitted contains a fork or a join set, workflow management is shared with the cloud: global dependencies are managed at the user site but management at the composition level is migrated to the cloud site. If the entire workflow structure is submitted to a single cloud site, then full execution management is also transferred to the remote site.

By submitting one composition at a time, the premise of dynamic management still holds. Once a composition is executed, all its nodes are not part of the remaining workflow from where a new composition can be formed for the next submission.

## 6. Related Work

Mapping scientific workflows for grid execution has been a topic of considerable attention that has resulted in useful tools and management systems. One of these systems is

Pegasus [10] that maps into the grid abstract workflows generated by Chimera [11]. Specifications of the application are written using chimera's SQL-based Virtual Description Language; Chimera provides an XML description of a DAG of the abstract workflow, which Pegasus transforms into a concrete workflow for submission to CondorÕs DAGMan for execution. Workflow design is based on abstract models that represent applications using DAG structures [12]. Abstract-to-concrete workflow is a transformation that prevails in a cloud environment [13], where an orchestration process completes the mapping into a concrete web-based executable workflow.

In [14] a series of workflow transformations referred to as *sequence, and split* and *and join* patterns lead to a single node reduction. These transformations provide the basis for the reduction schemes described in this paper. The grouping of tasks reported in GridSolve [15] are intended to minimize transfer delays by either having a multiple-resource site executing all tasks or supporting transfer of data between different parallel tasks executing in different servers. Furthermore, extensive work has been reported on workflow optimization for grid environments, on scheduling parallel clusters through Condor in [16], on schedule-based workflow balancing [17], on performance and overhead of high-performance applications [18], on task clustering for balanced workflows in [19]; task clustering is of interest because the functional description models described in this paper are based on partitioning an entire workflow into sub-workflows, similar to the heuristics reported in [20] and complemented with the integration of resource provisioning as reported in [21].

## 7. Conclusion and Future Work

The thrust of the work reported in this paper exploits the intrinsic parallelisms existing in scientific workflows by manipulating alternate structures amenable for execution management at any level of granularity. With the latest advances in technology, cloud-based computing is becoming more accessible, and in addition, parallel processing is expected not only at client nodes but also in grid/cloud-based service platforms. The representation based on functional descriptions addresses coarse levels of granularity present in small (desktop) applications written using a functional language to 1) enhance execution management such that a maximum dynamic parallelism is possible, 2) explore suitable partitioning schemes that generate a sequence of submissions such that data communication delays could potentially be reduced, and 3) generate for each submission a reduced number of service requests, and consequently engage and release less number of resources in a shorter time. Current work involves coding applications using Scala with a number of functions large enough to test parsing and generation of submissions to a suitable grid site. Future efforts include testing functional descriptions of large work-flows exercising fully orchestrated submissions to several cloud-based service sites using the cloud execution model discussed.

## References

[1] P. Hudak, J. Hughes, S. P. Jones, P. Wadler, *A History of Haskell: Being Lazy with Class*, The Third ACM SIGPLAN History of Programming Languages Conference (HOPL-III), San Diego, California, June 9-10, 2007.

[2] S. Marlow, *Parallel and Concurrent Programming in Haskell, version 1.2*, Mircrosoft Research Ltd., Cambridge, U.K., May 2012.

[3] D. E. Cooke, N. J. Rushton, B. Nemanich, R. G. Watson, P. Andersen, *Normalize, Transpose, and Distribute: An Automatic Approach for Handling Nonscalars*, ACM Transactions on Programming Language Systems, 30, 2, Article 9, March 2008.

[4] J. Epstein, A. P. Black, S. Peyton-Jones, *Towards Haskell in the Cloud*, ACM Haskell'11, Tokyo Japan, September 2011.

[5] Per Andersen, Daniel E. Cooke, *Assessment of SequenceL as a High-Level Parallel Programming Language*, 15th Intern'l Conference on Parallel and Distributed Computing, November 3-5, 2003, Marina del Rey, CA.

[6] J. A. Stratton, S. S. Stone, W. W. Hwu, *MCUDA: An Efficient Implementation of CUDA Kernels on Multi-cores*, IMPACT-08-01, A Technical Report, Center for Reliable and High-Performance Computing, University of Illinois at Urbana-Champaign, 2008.

[7] N. Lopez-Benitez, J. Djomehri, R. Biswas, *Task Assignment Heuristics for Parallel and Distributed CFD Applications*, International Journal of Computational Science and Engineering, Vol 3, No. 2, 2007.

[8] N. Lopez-Benitez, P. Andersen, *Dynamic Structures for the Management of Complex Applications in Grid Environments*, Proceedings of the 2009 International Conference on Grid Computing and Applications (GCA'09), pp. 80-85, Las Vegas, Nevada, July 2009.

[9] N. Lopez-Benitez, A. Rai, K. Kothari, S.E. Poduslo, *Parallelization and Analysis of the Linkmap Program*, in Parallel and Distributed Scientific and Engineering Computing, Nova Science Publishers, N.Y., Advances in Computation: Theory and Practice, V. 15, Editors: Yi Pan and L.Tianruo Yang, 2004.

[10] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, Daniel S. Katz, *Pegasus: A framework for Mapping Complex Scientific Workflows onto Distributed Systems*, Scientific Programming Journal, Vol 13(3), 2005, Pages 219-237.

[11] I. Foster, J. Vockler, M. Wilde, Y. Zhao, *Chimera: A Virtual Data System?for Representing, Querying, and Automating Data Derivation, Scientific and Statistical Database Management*, 14th International Conference 2002, pages 37-46.

[12] J. Yu, R. Buyya, *A Taxonomy of Scientific Workflow Systems for Grid Computing*, SIGMOD Record, Vol. 34, No. 3, September 2005, pages 44-49.

[13] G. Juve, E. Deelman, *Scientific Workflows in the Cloud, in Grids, Clouds and Virtualization*, M Cafaro, G. Alisio (editors), Compute Communications and Networks, Springer-Verlag, 2011, pages 71-91.

[14] Jaeger, M.C. Rojec-Goldmann, G. Muhl, G., *QoS Aggregation in Web Service Compositions, e-Technology, e-Commerce and e-Service*, 2005. IEEE '05. Proceedings. The 2005 IEEE International Conference, pp.181,185, 29 March-1 April 2005.

[15] Li Yinan, YarKhan Asim, Dongarra Jack, Seymour Keith, and Hurault AurŔlie, *Enabling workflows in GridSolve: request sequencing and service trading*, the Journal of Supercomputing, Springer US, June 2013, Volume 64, 3:1133-1152.

[16] G. Singh, C. Kesselman, E. Deelman, *Optimizing Grid-Based Workflow Execution*, Journal of Grid Computing (2006) 3:201-219.

[17] S. Rajakumar, V. P. Arunachalam, V. Selladurai, *Workflow Balancing Strategies in Parallel Machine Scheduling*, International Journal for Advanced Manufacturing Technology, 23, 2004, 366-374.

[18] P. Mehrotra, J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff, S. Saini, R. Biswas, *Performance Evaluation o Amazaon Elastic Compute Cloud for NASA High-performance Computing Applications*, Concurrency and Computation Practice and Experience, (2013).

[19] W. Chen, R. Ferreira da Silva, E. Deelman, R. Sakellariou, *Balanced Task Clustering in Scientific Workflows*, 9th International Conference on eScience, Beijin, China, October 23-25, 2013.

[20] W. Chen, E. Delman, *Partitioning and Scheduling Workflows across Multiple Sites with Storage Constraints*, Workshop on Scheduling for Parallel Computing, 9th Intl. Conf. on Parallel Processing and Applied Mathematics, Sept. 2011.

[21] W. Chen, E. Delman, *Integration of Workflow Partitioning and Resource Provisioning*, 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp 764-768.

# Host load Prediction-based GMDH-EA and MMTP for Virtual Machines Load Balancing in Cloud Environment

**Chenglei Peng, Qiangpeng Yang, Yao Yu, Yu Zhou, Ziqiang Wang, Sidan Du**
School of Electronic Science and Engineering, Nanjing University, Nanjing, China

**Abstract** - *Virtual machines (VMs) dynamic consolidation is effective to improve the utilization of resources and energy efficiency in cloud environment. However, the obligation of providing high quality of service to customers leads to the necessity in dealing with the energy performance trade-off, as aggressive consolidation may lead to performance degradation. Current solutions to the problem of host load detection are generally heuristic based. We propose a novel load balancing approach that combines the Group Method of Data Handling (GMDH) based on Evolutionary algorithm (EA) for host load prediction and the Minimum Migration Time policy (MMTP) for VMs migration. The GMDH-EA algorithm could predict the actual host load in each consecutive future time interval. We evaluate our method using the host load traces in the Google data centers with thousands of machines. The proposed algorithms significantly reduce energy consumption, while ensuring a high level of adherence to the Service Level Agreements (SLAs).*

**Keywords:** Cloud Computing, Dynamic Consolidation, Host Load Prediction, Group Method of Data Handling, Minimum Migration Time

## 1 Introduction

The proliferation of Cloud computing has resulted in the consuming of enormous amounts of electrical energy. One of the ways to address the energy inefficiency is to leverage the capabilities of the virtualization technology [1]. The virtualization technology allows Cloud providers to create multiple VMs instances on a single physical server. And the reduction in energy consumption can also be achieved by switching idle hosts to low-power modes (i.e., sleep, hibernation), thus eliminating the idle power consumption.

However, efficient resource management in Clouds is not trivial, as modern service applications often experience highly variable workloads causing dynamic resource usage patterns. Therefore, aggressive consolidation of VMs can lead to performance degradation when an application encounters an increasing demand resulting in an unexpected rise of the resource usage. Ensuring reliable Quality of Service (QoS) defined via Service Level Agreements (SLAs) established between Cloud providers and their customers is essential for Cloud computing environments; therefore, Cloud providers have to deal with the energy-performance trade-off – the minimization of energy consumption, while meeting the SLAs.

The focus of this work is on energy and performance efficient resource management strategies that can be applied in a virtualized data center by a Cloud provider (e.g. Google App Engine). We investigate performance characteristics for the problem of energy and performance efficient dynamic VM consolidation considering multiple hosts and multiple VMs. Effective host load prediction is conducive to dynamic resource provisioning [2], virtual machine migration [3], server consolidation and energy management. Therefore, accurate host load prediction is essential for load balancing.

In this paper, we propose an effective host load prediction method with comparatively less prediction errors and acceptable prediction interval length. The main idea of our approach is to use GMDH-EA method based on evolutionary algorithm for host load prediction and apply Minimum Migration Time policy (MMTP) to the VM selection stage.

The GMDH method is a self organizing method first developed by Ivakhnenko [4] and it has been applied to solve many prediction problems with success. Zadeh et al. [5] proposed a new GMDH-type neural network where evolutionary algorithm is deployed to design the whole architecture of the network. We have combined the Phase Space Reconstruction (PSR) and GMDH for host load prediction in previous work [6]. The MMTP migrates a VM that requires the minimum time to complete a migration, firstly proposed by Beloglazov A. et al. [7].

We evaluate the proposed algorithms by extensive simulation using the Cloudsim toolkit and the cluster workload traces from 29 days of the resource usage by about 11k machines in Google data centers.

In this paper, we make the following contributions:

1. *Our proposed method could predict the actual host load rather than the mean load only, the performance of our method has been investigated by different time intervals, i.e. 0.5h to 3h.*
2. *To the best of our knowledge, this is one of the first works to combine the GMDH-EA and MMTP approaches for host load balancing in the context of Cloud Computing.*
3. *An extensive simulation-based evaluation and performance analysis of the proposed algorithms.*

The remainder of the paper is organized as follows. In Section 2 we discuss the related work. We present a thorough analysis of the VM consolidation problem in Sections 3. In

Section 4 we introduce the system model used in the development of heuristics for the dynamic VM consolidation problem. We propose our algorithms in Section 5, continuing with an evaluation and analysis of the obtained experiment results in Section 6. We discuss future research directions and conclude the paper in Section 7.

## 2 Related work

Many efforts [8][9][10] have been made in host load prediction in Grids or HPC systems. C. Dabrowski et al. [8] perform the host load prediction by leveraging the Markov model via a simulated environment. S. Akioka, et al. [9] combine the Markov model and seasonal analysis to predict the host load for one-step ahead in a computational Grid. Y. Wu et al. [10] use hybrid model for multi-step ahead host load prediction, which combines the Auto Regressive (AR) model and Kalman filter. Although the previous methods have achived high accuracy for host load prediction in Grids, the Cloud host load holds a different scenarios. Google's traces show that the Cloud host load has more drastic fluctuation and higher noise, which we can see in [11].

B. Guenter [12] proposed a simple linear prediction scheme which predicts the host load for the next time. Q. Zhang [13] used the Auto-Regressive Integrated Moving Average (ARIMA) model to predict the host load. In [12], the ARIMA model could predict the load over a time window H by iterated the one step prediction. In [14], D. Yang et al. proposed a multi-step-ahead prediction method for CPU load. Their method contains three consequent steps. The first step is to find a fit function for the change range sequence of the original sequence. The second step is to predict the multi-step-ahead change pattern. However, the length of the immediately preceding sequence that is used to find the same sequence and derive the change patterns from the history data is not discussed.

S. Di et al. [15] firstly use the Bayesian model to predict the host load in the Cloud. They proposed 9 novel features to characterize the recent load fluctuation in the evidence window, and could predict the mean load over consecutive time intervals. However, their method has two limitations. The first one is that the training period in evaluation type B should contain the test period, which is not suitable for the Cloud environment. The other is that they use exponentially segmented pattern, which means the length of the segment increases exponentially. With the growth of the segment length, the mean load could not fully reflect the fluctuation of the host.

Srikantaiah et al. [16] have studied the problem of request scheduling for multi-tier web applications in virtualized heterogeneous systems to minimize energy consumption, while meeting performance requirements. The authors have found that the energy consumption per transaction results in a "U"-shaped curve, and it is possible to determine the optimal utilization point. To handle the optimization over multiple resources, they proposed a heuristic for the multidimensional bin packing problem as an algorithm for the workload consolidation. However, the proposed approach is workload type and application dependent, whereas our algorithms are independent of the workload type, and thus are suitable for a generic Cloud environment.

In contrast to the discussed studies, we combine the GMDH-EA and MMTP algorithms for dynamic adaption of VM allocation at run-time according to the current utilization of resources applying live migration, switching idle hosts to the sleep mode, and thus minimizing energy consumption. The proposed approach can effectively handle strict QoS requirements, multi-core CPU architectures, heterogeneous infrastructure and heterogeneous VMs.

According to the experiment results, our method achieves higher accuracy than the previous methods in mean load prediction. And what's more, our method can predict the actual load variation with a lower MSE over a long time interval, which is very important to the VMs consolidation. While combined with MMTP, our algorithms can adapt the behavior according to the observed performance characteristics of VMs.

## 3 The VM consolidation problem

In this section we analyze the problem of dynamic VM consolidation considering multiple hosts and multiple VMs. VM consolidation is the key problem that IaaS provider or data center operators often face. They need develop appropriate resource management and scheduling strategies to meet SLAs, improve load balancing capability and reduce energy consumption. Before the VM selection stage, we need know which host is overloaded. Then the next step is to select particular VMs to migrate from this host.

We define that there are $n$ homogeneous hosts, and the capacity of each host is $A_h$. Although VMs experience variable workloads, the maximum CPU capacity that can be allocated to a VM is $A_v$. Therefore, the maximum number of VMs allocated to a host when they demand their maximum CPU capacity is $m = \frac{A_h}{A_v}$ The total number of VMs is $nm$. VMs can be migrated between hosts using live migration with a migration time $t_m$. Obviously, SLA violation occurs when the total demand for the CPU performance exceeds the available CPU capacity $A_h$. The cost of power is $C_p$, and the cost of SLA violation per unit of time is $C_v$. Without loss of generality, we can define $C_p = 1$ and $C_v = s$, where $s \in R^+$. We assume that when a host is idle, i.e., there are no allocated VMs, it is switched off and consumes no power, or switched to the sleep mode with negligible power consumption. We call non-idle hosts active. The total cost C is defined as follows:

$$C = \sum_{t=t_0}^{T} (C_p \sum_{i=0}^{n} a_{ti} + C_v \sum_{j=0}^{n} v_{tj}) \qquad (1)$$

Where $t_0$ is the initial time; $T$ is the total time; $a_{ti} \in \{0,1\}$ indicating whether the host $i$ is active at the time t; $v_{tj} \in \{0,1\}$ indicating whether the host $j$ is experiencing an SLA violation at the time $t$. The problem is to determine when, which VMs and where should be migrated to minimize the total cost $C$.

## 4 The system model

In this paper, the targeted system is an IaaS environment, represented by a large-scale data center consisting of *N* heterogeneous physical hosts. Each host *i* is characterized by the CPU performance defined in MIPS, amount of RAM and network bandwidth. The storage is provided as an NAS to enable live migration of VMs. Multiple independent users submit requests for provisioning of *M* heterogeneous VMs characterized by requirements to processing power defined in MIPS, amount of RAM and network bandwidth. The fact that the VMs are managed by independent users implies that the resulting workload created due to combining multiple VMs on a single physical host is mixed. The mixed workload is formed by various types of applications which utilize the resources simultaneously. The users establish SLAs with the resource provider to formalize the QoS delivered. The provider pays a penalty to the users in cases of SLA violations.

The software layer of the system is tiered comprising local and global managers (Figure 1).

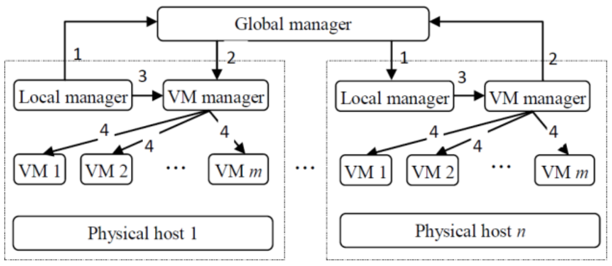


Fig. 1. The tiered system model

The local managers reside on each host as a module of the VM manager. Their objective is the continuous monitoring of the host's CPU utilization, resizing the VMs according to their resource needs, and deciding when and which VMs should to be migrated from the host (3). The global manager resides on the master host and collects information from the local managers to maintain the overall view of the utilization of resources (1). The global manager issues commands for the optimization of the VM placement (2). VMMs perform actual resizing and migration of VMs as well as changes in power modes of the hosts (4).

Based upon the above model, we propose an new hybrid control system (Figure 2) whose core components include: host load analyzer, host load scheduler, and VM monitor.

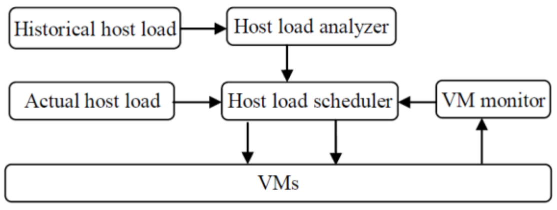


Fig. 2. The hybrid control system model

The Analyzer analyzes the changes in the load, using GMDH-EA algorithm to predict the future host loads; the scheduler mainly focus on the integrated management and scheduling, according to the actual load, predicted load , state parameters and other information resources.

The most important feature of this control system is based on the hybrid control mechanisms by combination of active control of prediction and passive control of feedback. Through the hybrid control system, we can not only be informed of the fluctuations of host load in advance by the prediction technique so that can allows the scheduler to implement more calmly the VMs migration policies. Therefore, the system can target to advance to play a preventive role. But we can also be informed of the actual implementation of the scheduling policy through feedback technique so that can play a role in real-time corrective control action.

## 5 The algorithms for VM consolidation

In this section, we propose several algorithms for dynamic consolidation of VMs based on an analysis of historical data of the resource usage by VMs. We split the problem of dynamic VM consolidation into four parts: (1) determining when a host is considered as being overloaded to migrate of one or more VMs from this host; (2) determining when a host is considered as being under-loaded to migrate all VMs from this host and switch the host to the sleep mode; (3) selection of VMs that should be migrated from an overloaded host; and (4) finding a new placement of the VMs selected for migration from either the overloaded or under-loaded hosts.

### 5.1 Host load prediction

Beloglazov A. et al. [17] apply an approach based on the idea of setting fixed utilization thresholds. However, fixed utilization thresholds are not efficient for IaaS environments with mixed workloads that exhibit non-stationary resource usage patterns. Also they use Local regression algorithm first proposed by Cleveland [18]. The main idea of the method of local regression is fitting simple models to localized subsets of data to build up a curve that approximates the original data.

In this section, we propose the GMDH-EA method for the host load prediction.

#### 5.1.1 The overview of GMDH-EA

The GMDH network is a feed-forward network that can be represented as a set of neurons, of which different pairs in each layer are connected through a quadratic polynomial and thereby produce new neurons in the next layer. The coefficients of the neuron are estimated using the Least Squares Method. The most popular base function used in GMDH is the gradually complicated Kolmogorov-Gabor polynomial:

$$\hat{y} = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ijk} x_i x_j x_k + \cdots \quad (2)$$

where *n* is the number of the data in the dataset; $A = (a_0;\ a_1;\ a_2;\ \ldots)$ and $X = (x_i;\ x_j;\ x_k;\ \ldots)$ are the vectors of the coefficients and input variables of the multi-input single-output system; and $\hat{y}$ is the output of an individual host. However, in the GMDH algorithm, the infinite Kolmogorov-Gabor polynomial is estimated by a cascade of a second order polynomials using only pairs of variables in the form of

$$\hat{y} = a_0 + a_1x_i + a_2x_j + a_3x_ix_j + a_4x_i^2 + a_5x_j^2 \qquad (3)$$

The basic form of the GMDH algorithm has several limitations, e.g., each host can only have two input variables, and the neurons in each layer are only connected to the host in its adjacent layer. Therefore, we choose GMDH-EA to remove these restrictions, as each neuron in GMDH-EA can have a different number of input variables as well as a different order of polynomial.

### 5.1.2 The presentation of GMDH-EA network

The representation of the GMDH-EA network should contain the number of input variables for each neuron, the best type of polynomial for each neuron, and which input variables should be chosen for each neuron. Therefore, the chromosome for each individual should contain three subchromosomes. Each subchromosome in our algorithm is represented as a string of integer digits.

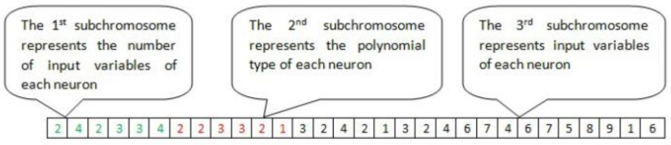

The 1st subchromosome represents the number of input variables of each neuron
The 2nd subchromosome represents the polynomial type of each neuron
The 3rd subchromosome represents input variables of each neuron
2 4 2 3 3 4 2 2 3 3 2 1 3 2 4 2 1 3 2 4 6 7 4 6 7 5 8 9 1 6

Fig. 3. The chromosome represents the GMDH-EA network

Figure 3 shows an example of a chromosome which represents an GMDH-EA network. This GMDH-EA network consists of three layers, and the neurons number of each layer are 3, 2 and 1. The number of input variables of each neuron ranges from 2 to 4, and the type of polynomials ranges from 1 to 3.

### 5.1.3 Estimate the coefficients of each neuron

In the GMDH-EA network, the coefficients of each neuron are derived by minimizing the mean squared error between $y$ and $\hat{y}$.

$$e = \frac{1}{Nt}\sum_{i=1}^{Nt}||y_i - \hat{y}_i||^2 \qquad (4)$$

Where $Nt$ is the size of the training set, and $y_i$ and $\hat{y}_i$ are the vectors of the actual and predict values. Using the training set, this gives rise to the set of linear equations

$$XC = Y \qquad (5)$$

The coefficients of each neuron are derived in the form

$$(X^TX)^{-1}X^TY \qquad (6)$$

Where $Y = [y_1, y_2, \dots, y_{Nt}]^T$, and the values of $X$ and $C$ are according to the number of input variables and the order of the polynomial.

### 5.1.4 The fitness function

The fitness function is very important to the GMDH-EA network, as it determines the performance of the model. In this paper, we use the locally weighted mean square error as the fitness function.

$$\Phi = \frac{1}{Nv}\sum_{i=1}^{Nv}W_i||y_i - \hat{y}_i||^2 \qquad (7)$$

Where $Nv$ is the size of the validation set, W is the weighting function. There are many weighting functions proposed by the researchers [15]. In this paper, we use the tricube kernel weighting function as follows:

$$W_i = (1 - (\frac{D_i}{\sum_{i=1}^{Nv}D_i})^3)^3 \qquad (8)$$

Where $D_i$ is the Euclidean distance of the input variables between the data in the validation set and the prediction set, which is used to indicate the similarity between the load in the validation set and the prediction set.

### 5.1.5 Crossover and mutation operations

The crossover and mutation operation are used to produce offsprings from two parents, which are chosen using the roulette wheel selection method. The crossover operation for the first and the second subchromosome is simply accomplished by exchanging the tail of each two subchromosomes from a random point. The change of the third subchromosome follows the change in the first one. The mutation operation is similar to the crossover operation.

## 5.2 VM selection

Once the system get the predicted load, it has been decided which host is overloaded or under-loaded. So the next step is to select particular VMs to migrate from this host. In this section we propose two policies for VM selection. The described policies are applied iteratively. After a selection of a VM to migrate, the host is checked again for being overloaded. If it is still considered as being overloaded, the VM selection policy is applied again to select another VM to migrate from the host. This is repeated until the host load is considered as being at the normal value.

### 5.2.1 The minimum migration time policy

The Minimum Migration Time policy (MMTP) migrates a VM $v$ that requires the minimum time to complete a migration relatively to the other VMs allocated to the host. The migration time is estimated as the amount of RAM utilized by the VM divided by the spare network bandwidth available for the host $j$. Let $V_j$ be a set of VMs currently allocated to the host $j$. The MMT policy finds a VM $v$ that satisfies conditions formalized in (9).

$$v \in V_j | \forall a \in V_j, \frac{RAM_u(v)}{NET_j} \leq \frac{RAM_u(a)}{NET_j} \qquad (9)$$

Where $RAM_u(a)$ is the amount of RAM currently utilized by the VM $a$; and $NET_j$ is the spare network bandwidth available for the host $j$.

### 5.2.2 The Random Selection Policy (RSP)

The Random Selection Policy (RSP) selects a VM to be migrated according to a uniformly distributed discrete random variable $X \overset{\text{def}}{=} U(0, |V_j|)$, whose values index a set of VMs $V_j$ allocated to a host $j$.

## 5.3 VM placement

The VM placement can be seen as a bin packing problem with variable bin sizes and prices, where bins represent the physical hosts; items are the VMs that have to be allocated; bin sizes are the available CPU capacities of the hosts; and prices correspond to the power consumption by the hosts. As the bin packing problem is NP-hard, to solve it we choose a modification of the BFD algorithm denoted Power Aware Best Fit Decreasing (PABFD) proposed by Beloglazov A. et al. [6], we sort all the VMs in the decreasing order of their current CPU utilizations and allocate each VM to a host that provides the least increase of the power consumption caused by the allocation. The pseudo code for the algorithm is presented in Algorithm 1. The complexity of the algorithm is nm, where n is the number of hosts and m is the number of VMs that have to be allocated.

**Algorithm 1:** Power Aware Best Fit Decreasing (PABFD)

```
1  Input: hostLst, vmLst Output: allocation of VMs
2  vmLst.sortDecreasingUtilization()
3  foreach vm in vmLst do
4      minPower ← MAX
5      allocatedHost ← NULL
6      foreach host in hostLst do
7          if host has enough resources for vm then
8              power ← estimatePower(host, vm)
9              if power < minPower then
10                 allocatedHost ← host
11                 minPower ← power
12     if allocatedHost ≠ NULL then
13         allocation.add(vm, allocatedHost)
14 return allocation
```

# 6 Performance evaluation

## 6.1 Experiment setup

The CloudSim toolkit [19] has been chosen as a simulation platform, as it is a modern simulation framework aimed at Cloud computing environments. It has been extended to enable energyaware simulations, as the core framework does not provide this capability. Apart from the energy consumption modeling and accounting, the ability to simulate service applications with dynamic workloads has been incorporated.

We have simulated a data center that comprises 1000 heterogeneous hosts. The number of VMs is 1600. All the load traces are real data coming from the 29 days of the resource usage by about 11k machines in Google datacenters [20]. We have randomly choosen 10 times load traces to form 10 data centers. For host load prediction method, we evaluated GMDH-EA and gave the actual prediction in different time intervals, i.e. 0.5h to 3h. The GMDH-EA parameters are shown in Table I, which are optimized to get the best performance for the load prediction.

TABLE I. THE PARAMETERS OF GMDH-EA

| Parameters | value |
|---|---|
| Max generation | 50 |
| Population size | 35 |
| Crossover rate | 0.9 |
| Mutation rate | 0.15 |
| Number of layers | 4 |
| Number of neurons of each layer | 9,6,3,1 |
| Number of inputs to be selected | 2-4 |
| Polynomial type | 1-3 |

## 6.2 Test metrics

In order to compare the efficiency of the algorithms we use several metrics to evaluate their performance. One of the metrics is the total energy consumption (EC) by the physical servers of a data center caused by the application workloads. The second metric is the level of SLA violations (SLAV). Another metric is the number of VM migrations initiated by the VM manager during the adaptation of the VM placement. All these three metrics results can be found in the CloudSim output.

## 6.3 Host load prediction

The accurate prediction of host load in a Cloud computing data center is very important to improve resource utilization, lower data center costs and ensure the job performance. The previous methods [12][13] for multi-step ahead prediction usually iterate the result of the one-step ahead prediction, which will generate cumulative errors.

However, the output of our proposed method is a vector of the host load, which will not generate cumulative errors regardless of the step length, as the current predict value has nothing to do with the last predict value. We quantified the performance of actual load prediction with mean squared error (MSE).

$$MSE = \frac{1}{H}\sum_{i=1}^{H}(A_i - F_i)^2 \qquad (10)$$

Where $H$ is the step length, $A_i$ and $F_i$ are the actual value and forecast value.

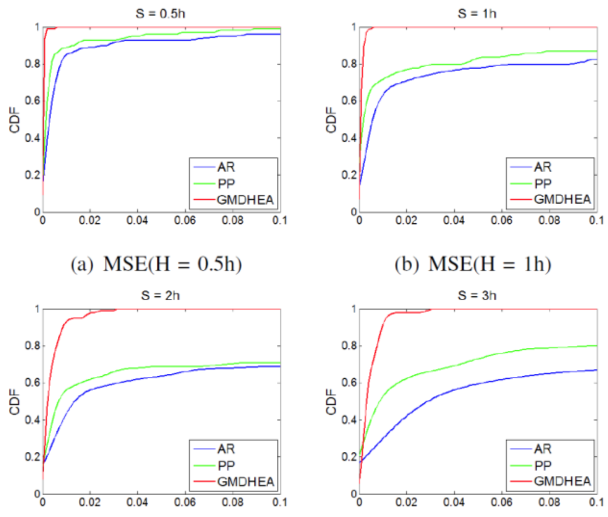


Fig. 4. MSE of actual load prediction

In Figure 4, we compare our method with the AR method and the Pattern Prediction (PP) method porposed by Yang [14]. The average MSE of our method in 3h ahead prediction is 0.0046, which is much lower than the other two methods. What's more, we can find that our proposed method keeps a good performance with the prediction step increases, while the performance of the other two methods has a large degree of decline.

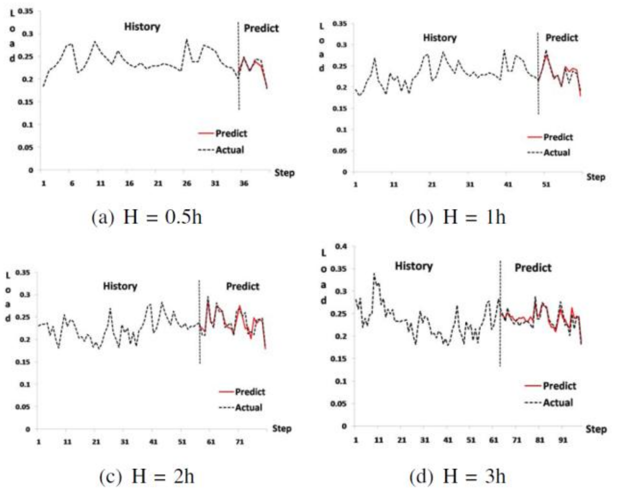


Fig. 5. Actual load prediction.

Figure 5 shows the load prediction results of hosts in the Google data center. As the interval in Google trace is 5 min, the step length of 0.5h to 3h is 6 to 36. And the y-label in Figure represents the CPU utilization, which has been normalized.

Our prediction result shows that the proposed method could achieve high accuracy although the host load fluctuates more drastically. As we can see in Figure 6, 7 and 8, our proposed method can still get a satisfactory performance.

## 6.4 Simulation results

To make a simulation-based evaluation applicable, it is important to conduct experiments using workload traces from a real system. For our experiments we have used data coming from the cluster workload traces of Google datacenters. The interval of utilization measurements is 5 minutes. We have randomly chosen record of 1600 tasks running on 1000 hosts of 29 days from the workload traces collected from May 2011 [20]. During the simulations, each VM is randomly assigned a workload trace from one of the VMs from the corresponding day. In the simulations we do not limit the VM consolidation by the memory bounds, as this would constrain the consolidation, whereas the objective of the experiments is to stress the consolidation algorithms.

TABLE II. AVERRAGE RESULTS

| Algorithms | Energy (KWH) | SLA Violation (%) | VM migration ($\times 10^3$) |
|---|---|---|---|
| LR-RSP | 84.94 | 4.38 | 17.98 |
| LR-MMTP | 83.82 | 4.32 | 17.33 |
| GMDH-RSP | 83.54 | 4.30 | 16.77 |
| GMDH-MMTP | 81.93 | 4.26 | 13.37 |

The average results of 10 data centers of the combinations of each host load detection algorithm and the MMT policy are shown in Table II.

We have simulated all combinations of the host load detection algorithms (LR and GMDH) and VM selection policies (MMTP and RSP).The results produced by the selected algorithms are shown in Figure 6, 7 and 8.

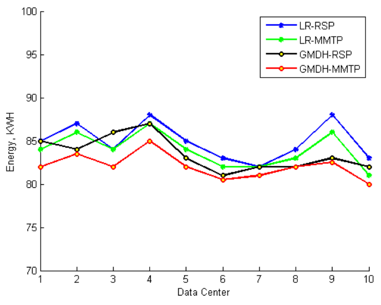


Fig. 6. Energy consumtion

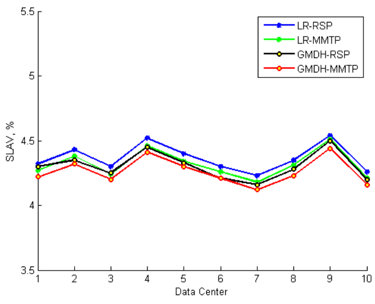


Fig. 7. SLA violations

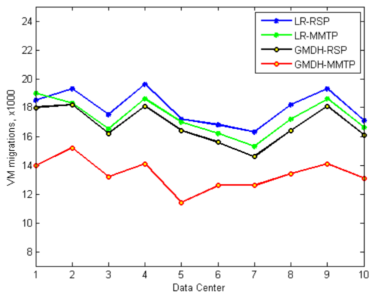


Fig. 8. VM migrations

From the observed simulation results, we can make several conclusions: (1) the GMDH-EA algorithm outperforms the local regression algorithm; (2) the MMTP policy produced better results compared to the RSP policy, meaning that the minimization of the VM migration time is more important;(3) the combination of GMDH-EA with MMTP algorithms outperform others.

## 7 Conclusion

To maximize ROI, Cloud providers have to apply energy-efficient resource management strategies, such as dynamic VMs consolidation and switching idle servers to power-saving modes. However, such load balancing is not trivial, as it can result in the SLA violations. In this paper we have conducted competitive analysis of the VM load balancing problems.

We proposed to combine GMDH-EA and MMTP algorithms for optimal online deterministic algorithms for these problems. According to the results of the analysis, we have proposed novel adaptive heuristics that are based on an analysis of historical data. We have also evaluated the proposed algorithms through extensive simulations on a large-scale experiment setup using workload traces from more than 11k machines in Google data centers. The results of the experiments have shown that the proposed GMDH-EA prediction algorithm combined with the MMTP selection policy significantly outperforms other VM consolidation algorithms in regard to the MSE metric due to a lower value in a long time interval and a substantially reduced level of SLA violations and the number of VM migrations.

In order to evaluate the proposed algorithm in a real Cloud environment, we plan to implement it by extending a real-world Cloud platform with commercial partner. Besides the reduction in infrastructure and on-going operating costs, this work also has social significance as it will decrease the carbon dioxide footprints and energy consumption by modern IT infrastructures.

## 8 Acknowledgment

This work was supported by the scientific and technical supporting programs of Jiangsu province, china, No.BE2011169.

## 9 References

[1] Xen and the art of virtualization. Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003), Bolton Landing, NY, USA.

[2] B. Urgaonkar, P. Shenoy, A. Chandra and P. Goyal, Dynamic Provisioning of Multi-tier Internet Applications, Proc of ICAC,2005.

[3] S. Osman, D. Subhraveti, G. su and J. Nieh, The Design and Implementation of Zap: A System for Migrating Computing Environments, Proc of OSDI, 2002.

[4] A. G. Ivakhnenko, Polynomial theory of complex systems, IEEE Trans. Syst. Man Cyber., vol. SMC-1, no. 4, pp. 364-378, 1971.

[5] N. Zadeh, A. Darvizeh, A. Jamali and A. Moeini, Evolutionary design of generalized polynomial neural networks for modeling and prediction of explosive forming process, J. Mater. Process. Tech., vol. 164-165, pp. 1561-1571, 2005.

[6] Q. Yang, C. Peng, Y. Yu, H. Zhao,Y. Zhou, Z. Wang, S. Du, Host Load Prediction Based on PSR and EA-GMDH for Cloud Computing System, in roceedings of the 2013 IEEE International Conference on Cloud and Green Computing(CGC 2013), pp. 9–15, 2013

[7] Beloglazov A., Buyya R., Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers, Concurrency Computat.: Pract. Exper. 2012; 24:1397–1420

[8] C. Dabrowski and F.Hunt, Using markov chain analysis to study dynamic behavior in large-scale grid systems, Seventh Australasian Symposium on Grid Computing and Research, ser. CRPT, vol. 99. Wellington, New Zealand:ACS, pp. 29-40, 2009.

[9] S. Akioka and Y. Muraoka, Extended forecast of cpu and network load on computational grid, in Proceedings of the 2004 IEEE International Symposium on Cluster Computering and Grid(CCGrid’ 2004), pp. 765-772, 2004.

[10] Y. Wu, Y. Yuan, G. Yang and W. Zheng, Load Prediction Using Hybrid Model for Computational Grid, Proceedings of 8th IEEE/ACM International Conference on Grid Computing, 2007.

[11] S. DI, D. Kondo and W. Cirne, Host Load Prediction in a Google Compute Cloud with a Bayesian Model, Procceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2012

[12] B. Guenter, N. Jain, and C. Williams, Managing Cost, Performance, and Reliability Tradeoffs for Energy-Aware Server Provisioning, INFOCOM, pp. 1332-1340, 2011.

[13] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, Dynamic Energy-Aware Capacity Provisioning for Cloud Computing Environments, Proceedings of the 9th international conference on Autonomic computing, 2012.

[14] D. Yang, J. Cao, C. Yu, and J. Xiao, A Multi-step-ahead CPU Load Prediction Approach in Distributed System, Sencond International Conference on Cloud Computing Green Computing, 2012

[15] S. DI, D. Kondo and W. Cirne, Host Load Prediction in a Google Compute Cloud with a Bayesian Model,Procceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2012

[16] H. S. Kim, R. Eykholt, J.D. Salas, Nonlinear dynamics, delay tiems and embedding windows, Physica D: Nonlinear Phenomena, 1999.

[17] Beloglazov A, Buyya R. Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, Bangalore, India, 2010; 4.

[18] Cleveland WS. Robust locally weighted regression and smoothing scatterplots. Journal of the American statistical association 1979; 74(368):829–836.

[19] Calheiros RN, Ranjan R, Beloglazov A, Rose CAFD, Buyya R. CloudSim: a toolkit for modeling and simulation of Cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience 2011; 41(1):23–50.

[20] https://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1

# SLA-Based Optimal Provisioning Method for Cloud Computing Environments

**Seunghwan Yoo**[1], **and Sungchun Kim**[1]
[1] Computer Science & Engineering Department, Sogang University, Seoul, South Korea

**Abstract** — *In cloud computing, cloud providers can offer cloud consumers two provisioning plans for computing resources, formal reservation and on-demand plans. Usually, cost of utilizing computing resources provisioned by reservation plan is cheaper than that provisioned by on-demand plan, since cloud consumer has to pay to provider in advance. With the reservation plan(Local Adjustment), the consumer can reduce the total resource provisioning cost. However, the optimal reservation of resources is difficult to be achieved due to fluctuation of consumer's future demand and providers' resource prices. So we propose a framework to improve their profits by maximizing the resource utilization and reducing the reconfiguration costs. Then a two-step runtime reconfiguration strategy. The SLA algorithm can provide computing resources for being used in multiple provisioning stages as well as a long-term plan(Global Adjustment), The Service Level Agreement (SLA) based scheduling approach promotes cooperative resource sharing. In this paper, minimizing both under provisioning and over provisioning problems under the demand and price uncertainty in cloud computing environments is our motivation to explore a resource provisioning strategy for cloud consumers. the results show that our framework is effective for maximizing the resource utilization and reducing the costs of the runtime reconfiguration.*

**Keywords:** Provisioning, Service Level Agreement, Optimize, Cloud Computing Platform

## 1 Introduction

Cloud computing[1] is basically an Internet-based network made up of large numbers of servers - mostly based on open standards, modular and inexpensive. Also, it is popular as a rising application paradigm, where resources, including software, platform and infrastructure, are provided and shared as services. Cloud providers are responsible for various resource demands by determining where to place VMs and how to allocate the resources. The virtualization-based cloud computing can improve resource utilizations, scalabilities, flexibilities and availabilities of applications. Also it can provide good application isolations in multiple levels. Due to these advantages, large-scale distributed applications are preferred to be hosted in a cloud platform.. A service-level agreement (SLA) is a part of a service contract where a service is formally defined. In practice, the term SLA is sometimes used to refer to the contracted delivery time (of the service or performance). As an example, internet service providers will commonly include service level agreements within the terms of their contracts with customers to define the level(s) of service being sold in plain language terms. This paper proposes a SLA-Based Optimal resource provisioning method (SAA provisioning Method). SAA provisioning method provides scalable processing power with dynamic resource provisioning mechanisms, where the number of virtual machine used is dynamically adapted to the time-varying incoming request workload. We evaluate the effectiveness of our initial VM deployment method used in sandpiper[19] and through runtime VM reconfiguration, we show the advantage in resource utilization.

The remainder of the paper is organized as follows. Section 2 presents survey related to our work. Section 3 describes our SAA provisioning method on the cloud computing platform. Section 4 presents our adaptive resource provisioning algorithm and its performance evaluation. Section 5 concludes the paper and points out some future research directions.

## 2 Related Work

Dynamic resource provisioning [4], which has been generally used in web hosting platforms, has proven to be useful in handling multiple time-scale workloads(VMs). However, dynamic provisioning in previous research has been more focus on physical resource allocation, which is not flexible enough for the effective delivering of services. Unlike other computing resources, VMs are flexibly deployed on physical machines, which can be automatically generated for different virtualized applications. Though existing physical capacity provisioning has long been used, overprovisioning or under-provisioning has been a common difficulty for most resource IT vendors. To solve this problem it is necessary to make full use of advantages of adaptive resource provisioning. We propose the design of a virtualized resource allocation framework using the cloud platform, which allocates VMs on demand in order to provide services, as well as minimizing the cost of using those virtual resources. Nowadays, some researches have focused on the issue of resource management and performance control in cloud computing platform[5,6]. However, new challenges are introduced while service providers benefit from the planning flexibility in technical and economic aspects. Some challenges and opportunities of automated control in cloud computing is discussed in [7]. And other researchers work to improve the resource utilization, such as resource virtualization [8,16], on-demand resource provisioning management based on virtual machines [9, 10], and QoS management of virtual machine [11].

Also, many researchers [12, 13] focus on improving resource utilization as well as guaranteeing quality of the hosted services via on-demand local resource scheduling models or algorithms within a physical server. However, most of them could not be good solutions to tradeoff between resource utilization and SLA. For example, [12] present a novel system-level application resource demand phase analysis and prediction prototype to support on-demand resource provisioning. The process takes into consideration application's resource consumption patterns, pricing schedules defined by the resource provider, and penalties associated with SLA violations. The authors in [13] improve resource utilization and performance of some services by hugely reducing performance of others. How to improve resource utilization, as well as guarantee SLA, is a challenge in a VM-based cloud data center

In the context of the dynamic resource provisioning, the author in [16] introduce three mechanisms for web clusters. The first mechanism, QuID [14], optimizes the performance within a cluster by dynamically allocating servers on-demand. The second, WARD [15], is a request redirection mechanism across the clusters. The third one is a cluster decision algorithm that selects QuID or WARD under different workload conditions.

For multi-tier internet applications, the modeling is proposed that a provisioning technique which employs two methods that operate at two different time scales : predictive provisioning at the time-scale of hours or days, and reactive provisioning at time scales of minutes to respond to a peak load[17].

In this section, we first discuss the service level agreements (SLAs) that we use in the paper. Then we give a high-level description of the test bed and three types of workload generators for our experimental studies. Finally, we describe the control system architecture that we use throughout the paper.

### 2.1 Service Level Agreements

Service level agreements (SLAs) are firm contracts between a service provider (IT Bender) and its clients (Users). SLAs in general depend on certain chosen criteria, such as latency, reliability, availability, throughput and security, and so on. In this paper, we focus on end-to-end latency, or maintain cost. Although SLA cost function may have various forms, we believe that a staircase function is a natural choice used in the real-world contracts as it is easy to describe in natural language [18]. We use a single step function for SLA in our paper as a reasonable approximation. We assume that if response time is shorter than arranged time, then the service provider will earn some revenue. Otherwise, the service provider will pay a penalty back to the client. As a result, in order to minimize the SLA penalty cost, our method should keep the response time right below arranged time

## 3 Proposed Scheme

### 3.1 SAA Framework

This section presents a scalable framework for virtualized applications on the cloud computing platform. The framework deals with the scenario that hosted on a cloud computing platform, handle many virtual machines simultaneously according to the incoming user requests. Since the amount of incoming requests changes with time and the cloud platform is a pay-per use service, the application has to dynamically assign the resources it uses to maintain guaranteed response time and reduce the total owner cost under various workloads. In the framework, server pool, combining a distinct computing server, is capable of processing multiple hybrid workload requests.

To efficiently utilize resources, there are two main issues considered in the cloud computing platforms. The first is finding the least loaded resource for dispatching incoming requests. The second issue deals with SAA provisioning for adaptively handling dynamic user's requests. With resource state monitoring, each workflow enactment request will be sent to the least loaded resource for service. The effectiveness of least load dispatching largely depends on how to accurately capture the computing load on each resource.

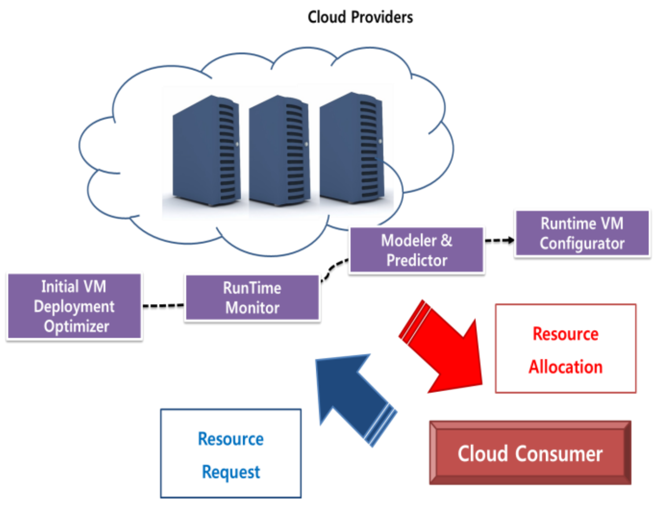


Fig. 1 Proposed SAA-Provisioning Framework

Fig. 1 shows an overview of the framework in handling user requests for Cloud Computing Environments. The architecture consists of four main components that Initial Deployment, Runtime Monitor, Modeler & Predictor, and Runtime VM Configurator which is a loops architecture. The goal is to meet the user requirements while adapting cloud architecture to workload variations. Usually, each request requires the execution of virtualized application allocated on the VM of each physical server. A cloud computing resource amount enables multiple virtualized applications may be increased when request increases and reduced when request reduces. This dynamic resource provisioning allows flexible response time in a cloud platform where peak workload is much greater than the normal steady state.

Our Framework provides a high-level dynamic resource provision architecture for cloud computing platform, which shows relationships between heterogeneous server resources pool and self-management function. Server pool contains physical resources and virtualized resources. A lot of VMs hold several Server Pool sharing the capacity of physical resources and can isolate multiple applications from the underlying hardware. VMs of a virtualized application may correspond to a physical machine.

Self-management function means mechanisms to automate the VMs of configuring and tuning the virtualized application so as to maintain the guaranteed response time for requirements of the diverse users. As previously stated, four main components more detail explanation are as follows:

① Runtime Monitor: Collects the runtime information, including resource usage, network load, and request arrival rate, such as the response time, the request arrival rate, the average service time, and the CPU utilization, etc.. All information is sampled periodically without affecting application performance significantly

② Modeler & Predictor : Use data from Runtime Monitor to calculate the objective values and predict the future state.

③ Runtime VM Configurator : decides when and how to reconfigure the VMs. To reduce the runtime reconfiguration costs

In conclusion, Fig. 1 is presented the dynamic resource provisioning method. Our research is a great help of on the improved design of resource scheduler for requested workload. The goal is to minimize the using of resources for request workload while satisfying different users for the guaranteed response time.

## 3.2 Proposed SAA provisioning Algorithms

In this section, we propose an auto-control algorithm denoted as SAA provisioning method (SLA Aware Adaptive) to dynamically provide an adequate amount of resources to virtualized application. To maintain acceptable response time and cost efficiency, it would find the configuration value which the Sum of cloud platform profits is maximized. Considering all of virtual machine system parameters observed by monitor, especially response time and usage cost, we compute the profit value of each VMs. Through equation (1), our method calculates the optimized next step setting value. Resource scheduler receives the modified configuration parameter. Then it reflects the value next schedule period.

Table.1 List of Notations

| Symbol | Definition |
|---|---|
| $r(R_A)$ | Rate of Arrival Request |
| $r(SLA_S)$ | Rate of SLA Satisfied |
| $r(VM_F)$ | Rate of VM Failed |
| $c(VM_A)$ | Active VM maintain Cost |
| $c(VM_I)$ | Idle VM maintain Cost |
| $\alpha$ | Created Value (per Application) |
| $\beta$ | Weight Value (per Application) |

$$\text{Profit}(P_i) = \alpha \times \{r(R_{Ai}) \times r(SLA_{Si}) - r(VM_{fi})\} - \beta \times \{c(VM_{Ai}) + c(VM_{Ii})\} \quad (1)$$

After each Server-Pool Profit(Pi) is calculated, Periodically it is updated and check SLA-requirements. After the specific point which variability is minimized, Our Scheme elect optimized parameter for Global Profit.

$$\max\{ Profit_{global} = \sum_{i=1}^{n} p_i \} \quad (2)$$

After all, our mechanism collect local profit and calculate Global profit as shown in (2). It would find the optimal value for certain period. Also it is adaptively perform in the course of time. Since it has sufficient information about virtualized application. For example, there are a little difference between current parameter and next-step parameter. It check prefixed threshold. If it is not exceed, retain the system current parameter. Finally, Our SAA provisioning algorithms would find SLA- guaranteed response time and low maintenance cost.

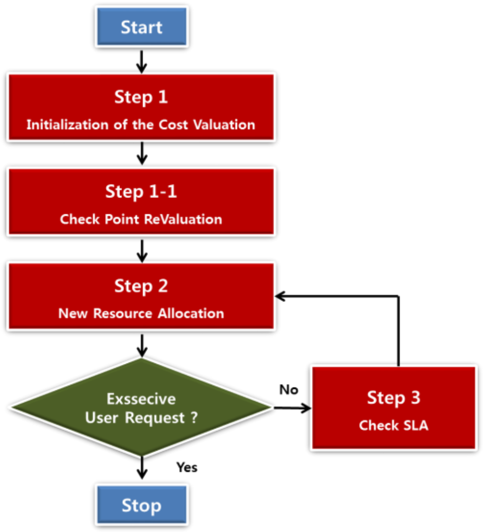


Fig. 2 Flowchart Calculate Optimized Profit

## 4 Performance Analysis

### 4.1 Experiment

In the following experiments, we evaluate our dynamic resource provisioning technique for virtualized applications. We establish a prototype system of cloud environment such that each of the server nodes was run on Intel Xeon 3.2GHz processors with 24GB RAM. Processing capacity of each VM server is equal in cloud platform.

We evaluate the effectiveness of our initial VM deployment method used in sandpiper[18]. The VM template capacities and application demands are distributed in the following sets: CPU-{0.25*2.4, 0.5*2.4, 1*2.4, 1.25*2.4, 1.5*2.4, 2.0*2.4, 3*2.4, 4*2.4}, memory-{0.5, 1, 1.5, 2.0, 3.0, 4.0, 6.0}, network I/O-{4, 6, 10, 15, 20}. For example, Physical server's total capacity is {4*3.4GHZ, 24GB, 100M}.

Table.2 Use of Servers

| ID | Use |
|---|---|
| $S_1$ | File system for VM migration |
| $S_2$ | Client workload generator for applications |
| $S_3$ | Request router, distributing client requests |
| $S_4$ ~ S7 | Hosting applications packaged into VM |

Table.3 VM Template

| ID | Name | Configuration |
|---|---|---|
| V1 | Common | 0.5*2.4GHZ, 1GB RAM, 10M I/O |
| V2 | High-CPU | 2*2.4GHZ, 1.5GB RAM, 20M I/O |
| V3 | High-Mem | 1*2.4GHZ, 3GB RAM, 20M I/O |
| V4 | High-I/O | 1*2.4GHZ, 1GB RAM, 30M I/O |

Table.4 Application Instances and Allocation

| App Type. | App. ID | VM Template | Instance Number | Server ID |
|---|---|---|---|---|
| CPU-Intensive | CI-1 | $V_2$ | 2 | $S_4$ ,$S_5$ |
| | CI-2 | $V_2$ | 2 | $S_4$ ,$S_5$ |
| Mem-Intensive | MI-1 | $V_3$ | 2 | $S_4$ ,$S_5$ |
| I/O-Intensive | NI-1 | $V_4$ | 2 | $S_4$ ,$S_5$ |

We employ three types of applications, CPU-intensive (CI), Memory-intensive (MI) and Network I/O-intensive (NI), with multiple instances. Multiple VM templates are provided and allocated to these applications. TABLE 2, 3, 4 describe the servers' uses, VM template configurations and application instances respectively.

### 4.2 Experiment result

Existing method, focus on maximizing resource utilization is approximately demonstrated 87% SLA-satisfied rate. We give consideration to improve SLA-satisfied ratio. Our mechanism indicate settlement for content better SLA-satisfied rate and diminish maintain cost.

Fig. 3 is our deployed VM template simulation result. Each type application requires unpredictable demand. So there are variable response time. We should stable cloud platform performance due to such fluctuation.

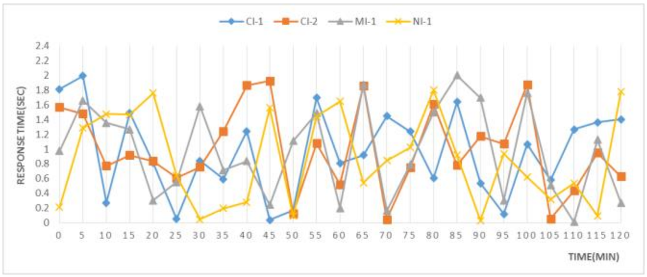


Fig. 3 Initial VM Deployment Result

Our method conducts initiation and removal of VMs before each interval while considering the utilization of the previous interval. It is noticeably cost-aware. And the results of response times and costs are shown in Fig. 4 and Fig. 5.

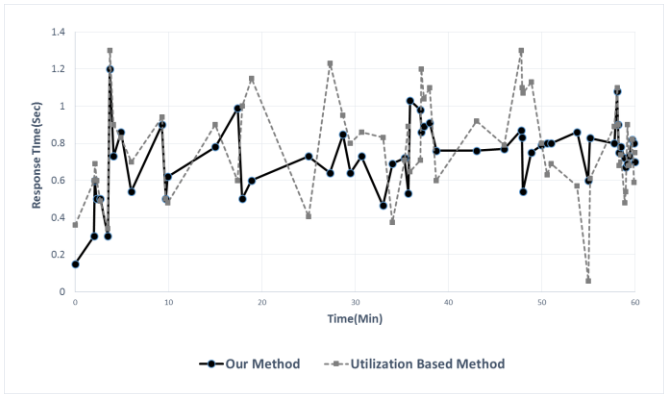


Fig. 4 Comparative results of Average response times between our method and utilization-based method

The benefit of our proposed method is appeared in Fig 4 and 5. Response time is faster 6.53% than existing method. And on average, maintain cost is reduced 6.92%. Through comparative results, we can notice that utilization-based methods can also handle the workload variations. However, it is not rapid enough to supply the proper number of VMs in order to meet the response based on time restricted. In respect of cost, this method occasionally uses fewer resources than our proposed method, sometimes at the cost of violating the SLA. In conclusion, the minimized number of VMs as well as the maximized CPU resource utilization can be achieved with our method by dynamic resource provisioning mechanism, and then we can keep the high global utility.

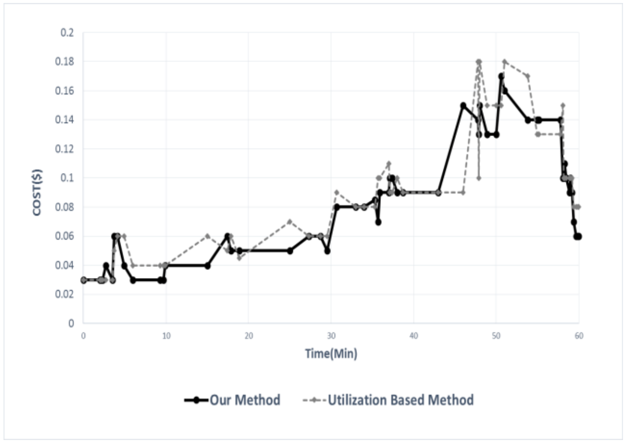


Fig. 5 Comparative results of Average costs between Our method and utilization based method

## 5 Conclusions

In this paper, it is argued that dynamic provisioning of virtualized applications environment raises new challenges not addressed by prior work on provisioning technique for cloud computing platform. We presented an optimal autonomic virtual machine provisioning architecture. We proposed a novel dynamic provisioning technique, which was algorithms for virtualized applications in cloud computing platform. Hence the efficiency and flexibility for resource provisioning were improved in cloud environment.

Currently many server applications adjust the amount of resources at runtime manually. So we address the problem of the VM deployment and reconfiguration The framework in this paper allows applications to automatically manage the amount of resources according to the system workload. It offers application providers the benefits of maintaining QoS-satisfied response time under time-varying workload at the minimum cost of resource usage. Also, we adopt Service Level Agreement (SLA) based negotiation of prioritized applications to determine the costs and penalties by the achieved performance level. If the entire request cannot be satisfied, some virtualized applications will be affected by their increased execution time, increased waiting time, or increased rejection rate.

However, there are still some limitations in our work, including: 1) the prediction techniques may have observational error which affect the runtime VM reconfiguration decisions; 2) the interferences between VMs on a hybrid server type are ignored. In future work, we will focus on these limitations by applying the much more accurate prediction techniques and consider hybrid server architecture.

### ACKNOWLEDGMENTS

This work was all-supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (No. 2012R1A1A2009558), and part - supported by Technology Innovation Development Program funded by Small & Medium Business Administration(No. S2057016)

## 6 References

[1] M. Armbrust, et al. “Above the clouds: A Berkeley view of cloud computing,” Tech. Rep. UCB/EECS-2009-28, EECS Department, U.C. Berkeley, 2009

[2] M. Armbrust, A. Fox, and R. Griffith, et al, “Above the clouds: A Berkeley view of cloud computing”, Technical Report No. UCB/EECS-2009-28, University of California Berkley, USA, Feb. 10, 2009.

[3] R. Buyya, C.S. Yeo, and S. Venugopal, et al, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility", Future generation computer systems, Elsevier science, Amsterdam, the Netherlands, 2009, 25(6), pp. 599- 616.

[4] S. Li, and D. Tirupati. Technology choice with stochastic demands and dynamic capacity allocation: A two-product analysis. *Journal of Operations Management*, Vol. 12, no 3-4, pp. 239-258, 1995.

[5] E. Kalyvianaki, T. Charalambous, and S. Hand, "Self-adaptive and self-configured CPU resource provisioning for virtualized servers using kalman filters", Proceedings of the 6th international conference on Autonomic computing, Barcelona, Spain, June 15-19, 2009.

[6] W. E. Walsh, G. Tesauro, and J. O. Kephart, "Utility functions in autonomic systems", Proceedings of the First IEEE International Conference on Autonomic Computing, New York, NY, USA, May 17-18, 2004

[7] E. H. Miller Lim, H., Babu, S., Chase, J., Parekh, S.: Automated Control in Cloud Computing: Challenges and Opportunities. In: 1st Workshop on Automated Control for Datacenters and Clouds, 2009.

[8] P. Barham, B. Dragovic, and K. Fraser, et al, "Xen and the art of virtualization", Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, 2003, pp. 164-177.

[9] Y. Song, Y. Li, and H. Wang, et al, "A service-oriented priority based resource scheduling scheme for virtualized utility computing", Proceedings of the 9th IEEE International Symposium on Cluster Computing and the Grid, 2009, pp. 148-155.

[10] J. Zhang, M. Yousif, and R. Carpenter, et al, "Application resource demand phase analysis and prediction in support of dynamic resource provisioning", Proceedings of the 4th International Conference on Autonomic Computing, 2007.

[11] X.Y. Wang, Z.H. Du, and Y.N. Chen, et al, "Virtualization based autonomic resource management for multi-tier Web applications in shared data center", The Journal of Systems and Software, 2008, 81(9), pp. 1591-1608

[12] J. Zhang, M. Yousif, and R. Carpenter, et al, "Application resource demand phase analysis and prediction in support of dynamic resource provisioning", Proceedings of the 4th International Conference on Autonomic Computing, 2007, pp. 12-12.

[13] P. Padala, X. Y. Zhu, M. Uysal, et al, "Adaptive control of virtualized resources in utility computing environments", EuroSys, 2007, pp. 289-302.

[14] Ranjan, S., Rolia, J., Fu, H., Knightly, R.: QoS-Driven Server Migration for Internet Data Centers. In: The Tenth International Workshop on Quality of Service, Miami, FL, 2002

[15] Ranjan, S., Karrer, R., Knightly, E.: Wide Area Redirection of Dynamic Content in Internet Data Centers. In: The IEEE INFOCOM, HongKong, 2004

[16] Ranjan, S., Knightly, E.: High-Performance Resource Allocation and Request Redirection Algorithms for Web Clusters. IEEE Transactions on Parallel And Distributed Systems 19(9), 2008.

[17] Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., Agile, T.W.: Dynamic Provisioning of Multi-Tier Internet Applications. ACM Transactions on Autonomous and Adaptive Systems 3(1), 2008.

[18] S. Malkowski, M. Hedwig, D. Jayasinghe, C. Pu, and D. Neumann, "Cloudxplor: A tool for configuration planning in clouds based on empirical data," in Proc. of SAC, 2010.

[19] T. Wood, et al, "Black-Box and Gray-Box Strategies for Virtual Machine Migration," in Proceedings the 4th USENIX Conference on Networked Systems Design and Implementation, 2007.

# DiAF: A Dynamic virtual Appliance provision and management Framework for cloud Computing

**Rajendar Kandan**[1], **Mohammad Zakaria Alli**[2] **and Hong Ong**[2]
Advanced Computing Lab, MIMOS, Kuala Lumpur, Malaysia

**Abstract** - *Cloud computing has driven the power of aggregating computing resource across geographical location, enabling enterprises an easier way of establishing and managing resources virtually. Most of industries rely on cloud computing for reducing the ownership cost and maximizing the profit. However it requires an efficient solution to build an infrastructure and offer software deployment and maintenance in a much reduced time. Virtual appliance addresses this solution and provides an easier way of deploying software applications in shorter time, removing the burden towards installation and configuration of applications. It acts as a ready built solution to the cloud consumers, eliminating the manual intervention in configuring any software applications. Many open source cloud systems offer this facility for users to build and deploy virtual appliances. We have developed a provisional and management framework (DiAF) which differs from the traditional approach and provide much more efficient way of managing the appliances. DiAF manages the conversion of appliances over different formats and deploys them according to the user requirement. In this paper, we also shared our experience with multi-tier appliance deployment and its results.*

**Keywords:** Virtualization, Cloud computing and Virtual appliance

## 1 Introduction

Cloud technology has proven the concept of dynamic resource management. It realizes on-demand resource provisioning to users irrespective of the geographical locations. Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services [1]. Many definitions about cloud computing have been put forth by various organizations across world and one of the recent definition include NIST [2], where it describes cloud computing as a model for delivering various services. It also describes five essential characteristics of cloud, service and deployment models. Service model describes about the type of services being offered to the user which includes Infrastructure as Service, Platform as Service and Software as Service. Deployment models states about the type of cloud which includes public, private, hybrid and community.

In Cloud computing, user can request virtual machine instances from an image which is preconfigured with a basic operating system like Ubuntu, Centos, Fedora, etc. These images are stored in a repository and user has option to choose the virtual image according to their needs. However, if the user wants to develop an application, their requires a long process including selection of base operating system, installation of software required for specific application and finally setting up configuration. This process can be much reduced by the process of employing virtual appliances, pre-integrated, self-contained system which is a combination of operating system with the adequate software application [3]. Virtual appliances [9] are software appliances prepared to run on any virtualized environments and hence widely used in most of the virtualization industries.

Virtual appliances are more specific to each application eg. Database appliance [4] and are executed in the form of virtual machines, removing the manual installation and configuration of software application. The user can deploy virtual appliance as virtual machines and can be ready to use application with much reduced time.

Current cloud system provides users, a way to access virtual appliances which is already published in a marketplace [5, 6]. Each cloud providers manage different market place for their user community. User can also build virtual appliances and upload in marketplace for others to use. Various virtual appliance building software were available on the market. Our focus is not on the development of the virtual appliances but rather on the deployment of virtual appliance from the available marketplace and manages them more efficiently. Wide range of image format is being followed by different cloud providers and hence one of our main focuses is on providing a unique architecture for dynamically managing the appliances of varied formats. In this paper, we present architecture for dynamic deployment of appliance integrated with open source cloud software Mi-cloud, an initiative from MIMOS.

This paper is organized as follows: Section II discusses on related papers. Then section III offers general description of our proposed architecture and its components. Section IV deals with the life cycle of virtual appliances and detailed information about our use cases is made in Section V followed by a conclusion in Section VI.

## 2 Related Papers

There were only a few initiatives towards the deployment of multi-tier based applications using virtual appliances. Amazon web service, CloudFormation [15] used to create distributed applications. User can choose from the list of template or can create an own template while launching an instances. However, the limitation includes the usage within the AWS environment and only AMI images are used for deployment. Further, CloudFormation as of independent product doesn't describe more on the configuration management of multi-tier based application and VM contextualization. An Open source initiative, Context Broker [24] from Nimbus project allows the creation of large virtual cluster and introduced the concept of "one-click" cluster. It manages the configuration using context agent, a lightweight agent on each VM for performing the required action. However the hypervisor is limited to xen or kvm. Further, it is more focused on science cloud and lacks a management system for virtual appliances.

VMware studio [16] from VMware is a free tool for managing and updating virtual appliances. It provides the functionality of application authoring, management, compatibility and validation. The limitation of this product is that it could be worked only with VMware product platforms. Claudia [17] is another open source service management toolkit which allows service provider to control resource provisioning and scalability. As other software, it also lacks the repository management system for virtual appliances. It also lacks a graphical user interface for managing the admin operations over virtual appliances.

ViApps [18] is another initiative from opencloud solutions. It is an open source tool managing the automation of infrastructure services on cloud environment. It manages the network infrastructure services like firewall, DNS servers, HTTP proxy, SMTP gateways and IP Load balance. However this product limits to the management of only network appliances. Similar approach to our proposal is provided by OpenNebula [19] an open source project for building and managing the enterprise clouds. It offers OpenNebula Marketplace [20] for users to deploy virtual appliance on OpenNebula clouds. To deploy virtual appliances, the users select virtual appliances from Marketplace and then download to their local site, prepare the template and then instantiate the virtual appliance. However, this implementation lacks the feature of communicating with external marketplace and dynamic deployment of virtual appliance.

Hence, we propose architecture to resolve the problem which failed to address by these softwares and provide a unique feature of managing the complete lifecycle of deploying virtual appliance. The complete detail of our architecture is explained in the next section

## 3 Proposed Architecture

DiAF manages the complete life cycle of virtual appliance. This component interacts with Mi-Cloud component in managing the deployment of multi-tier applications. It has unique feature of communicating with public marketplace, download, convert and deploy virtual appliance. It also handles the image repository for managing the deployment of virtual appliances.

User interacts with Mi-cloud using Mi-cloud portal, a graphical interface for requesting their service. The user can view list of appliance available at local repository and also has provision to choose appliance from public marketplace, turnkey linux [10]. User first creates the template for launching single/multi-tier applications and can set the order by which virtual appliances can be booted. The user can also add the contextualization parameters to the template if required.

The detailed architecture of DiAF is described in fig 1.

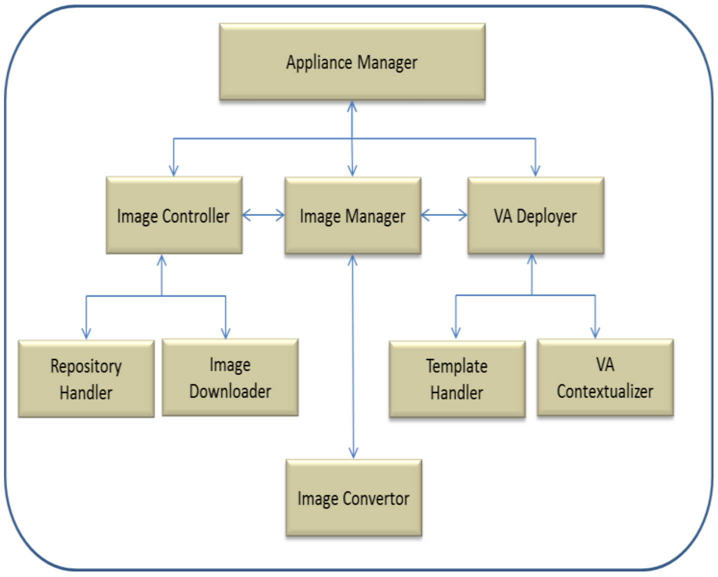


Figure 1. DiAF - Architecture

Appliance Manager handles the complete life cycle of virtual appliances. It interacts with Mi-cloud components and updates status of the virtual appliance periodically. The major components include Image Controller, Image Manager and VA Deployer.

### 3.1 Image Controller

Image Controller manages the download of images requested by the Appliance Manager. It periodically updates about the status of image which is being downloaded. It has two sub components namely Repository Handler and Image Downloader. Repository Handler connects to the external/local repository for handling requested image. It initiates the downloading process of image from the desired repository. Image Downloader verifies the image which is being downloaded from the external/local repository. It also updates the status of completion to the Image Controller.

### 3.2 Image Manager

Image Manager controls the image conversion process. It ensures that the image downloaded from that repository is converted to the desired format for the Virtual appliance deployment. It has sub component Image converter which converts the images according to the desired format.

### 3.3 VA Deployer

VA Deployer manages the virtual appliance deployment. It ensures the requested virtual appliance is deployed as VM and periodically updates the status to the Appliance Manager. It has two sub components Template handler and VA contextualizer. Template handler creates the image and virtual machine template for the deployment of virtual appliances. VA contextualizer configures the multi-tier based appliances and updates information to VA deployer about the status of configuration.

## 4 Life Cycle of Virtual appliance

The deployment of virtual appliances follows various stages as shown in Fig.2

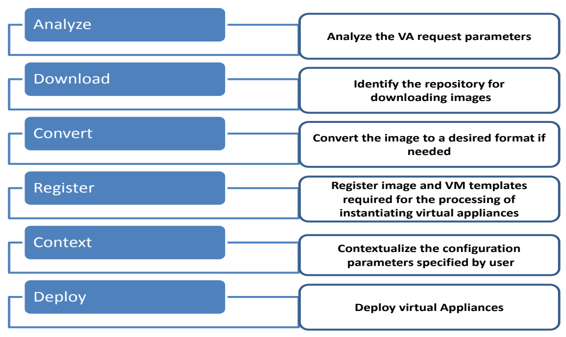

Analyze
Analyze the VA request parameters
Download
Identify the repository for downloading images
Convert
Convert the image to a desired format if needed
Register
Register image and VM templates required for the processing of instantiating virtual appliances
Context
Contextualize the configuration parameters specified by user
Deploy
Deploy virtual Appliances

Figure 2. Lifecycle of Virtual Appliance

First stage consists of analyzes, where the requested parameters were verified. This stage also decides whether the requested Virtual appliances can be processed by the cloud infrastructure. The second stage is the download process of the appropriate images. Next is the conversion phase where the downloaded images are converted if necessary. Once the images are converted and made ready, the next phase is to register the image. Next stage is the contextualization phase; here the configuration management takes place. Depending upon the user specification virtual machine templates were updated and finally boots virtual appliance in the last stage of deployment.

## 5 Implementation

The complete detail about our test bed is shown in Figure 3. Setup consists of Mi-Cloud frontend, managing the complete user and admin operations with two clusters, each consists of two Mi-Cloud nodes, which manages the life cycle of virtual machines. We use Mysql as database and NFS for sharing the file and directory across the nodes and Frontend.

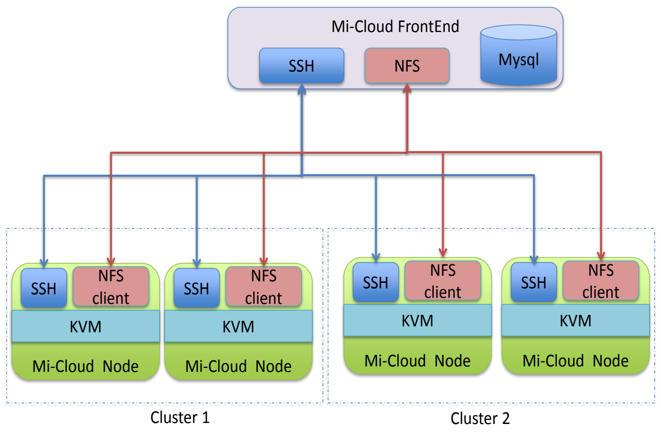


Figure 3. Cloud setup using Mi-cloud

Our experiments were carried out on a quad-core Intel Xeon CPU 5140 machine with 2.33 GHz. The machine has 32GB main memory and 250GB of local storage.

We describe our experiments using following scenarios

*Scenario 1: Deploying an appliance from Marketplace*

We carried out the experiment by selecting "turnkey linux" as external repository.

| Appliance Name | Image Size [qcow2] (MB) | Image Registration (Seconds) | Appliance Deployment (Seconds) |
|---|---|---|---|
| | | Converting from VMDK to qcow2 format, Template creation and Image Registration | Boot time |
| Django | 786 | 45 | 60 |
| Joomla | 887 | 56 | 61 |
| OrangeHRM | 908 | 46 | 61 |
| Torrentserver | 860 | 56 | 65 |

Table I : Single appliance deployment

Table I describes our experiment results with different appliances of varied sizes. In this scenario, we employed the conversion of images from VMDK to qcow2 for managing across Mi-Cloud environment. This conversion mechanism is not limited only to these formats and can varied according to the hypervisors including xen and vmware. As seen from table I, booting time for all the VMs show an average of 60 seconds, and this refers to first time boot configuration employed and this could be optimized for a faster launching of appliances during the second time. We optimized the way in which if the user launches the same appliance for second time, the deployment time will be much reduced. We can see the comparison graph in Fig 4.

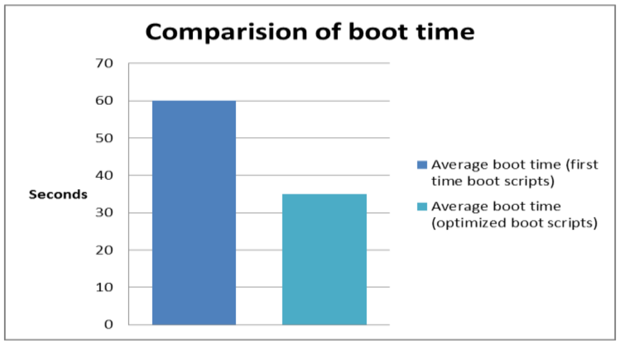


Figure 4. Comparision of booting time

*Scenario II: Deploying multi-tier appliance*

In our second experiment, we launch a multi-tier based appliances. We used “Nginx”, “MySyql” and “OwnCloud” appliances for loadbalancer, database and application respectively. Graphical user interface is designed in such a way that user can easily choose multiple appliances and specify the required parameters for setting up database and application configuration.

We customized these appliances and associated images were readily available in local site. We also employ the optimized boot and context scripts for the deployment of these appliances. The number of instances which we deployed using these appliances is represented below

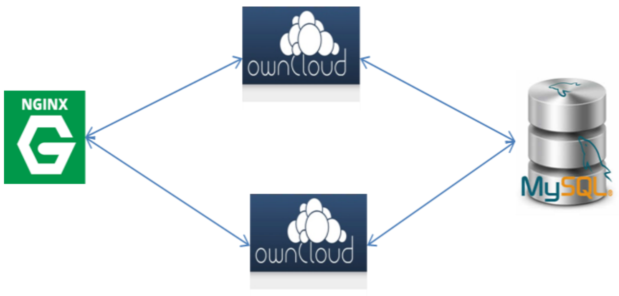


Figure 5. Instances of multi-tier appliances

Whenver the user launches multi-tier appliances, initialization of image starts parallely to provide faster launching. Contextualization script is managed to bind the relationship among the application, database and loadbalancer. The complete details about deployment time is represented in Fig 6.

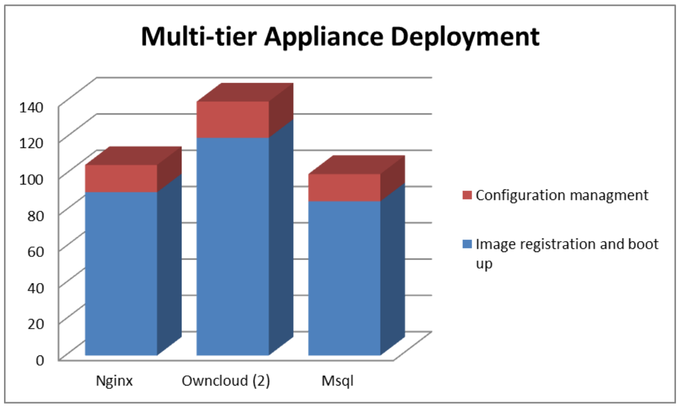


Figure 6. Multi-tier appliance deployment

As shown in fig 6, we can see the deployment time of each appliances. The total deployment time includes image registration, instance boot up and configuration for setting up application or database. We can see the total time for launching the entire multi-tier appliances is around 140 seconds. This provide users a easier approach for deploying any multi-tier appliances.

## 6 Conclusion

In this paper, we have presented architecture to provide dynamic deployment of appliances, which enables users, a simpler way to launch appliances across sites without any manual intervention. Furthermore, it also provides a basic approach over image management and fits for cloud environment which focus over interoperability, allowing users to launch different appliances of varied formats. For our future work, we would like to investigate more on VM image distribution, synchronizing and security aspects.

**Acknowledgment**

The authors sincerely thank MIMOS BERHAD, for financially supporting the Advanced Computing Laboratory, MIMOS BERHAD, Kuala Lumpur, Malaysia for carrying out research.

## References

[1] M. Armbrust, A. Fox, and R. Griffith, et al, “Above the clouds: A Berkeley view of cloud computing”, Technical Report No UCB/EECS-2009-28, University of California Berkley, USA, Feb 10, 2009
[2] http://csrc.nist.gov/publications/PubsSPS.html
[3] http://www.turnkeylinux.org/virtual-appliance
[4] A. Aboulnaga, K.Salem, A.A.Soror, U.F.Minhas, P.Kokosielis and S.Kamath, “Deploying database appliances in the cloud’, IEEE Data Eng. Bull..Vol.32,no 1 , pp. 13-20, 2009
[5] Public EC2 Amazon machine images, 2010. URL: http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=171

[6] VMWare public virtual appliances, 2010. URL: http://www.vmware.com/appliances/
[7] Eric Hammond, http://www.alestic.com
[8] Bitnami, http://www.bitnami.org
[9] rPath, http://www.rpath.com
[10] Turnkey Linux, http://www.turnkeylinux.org
[11] The Cloudmarket, http://www.thecloudmarket.com
[12] C. Sapuntzakis, D. Brumley, R.Chandra, N.Zeldovich, J.Chow, M.S.Lam and M. Rosenbulm, "Virtual appliances for deploying and maintaining software", in Proceedings of seventeenth large installation system administration conference (LISA 2003 ), October 2003
[13] Gabor Kecskemeti, Gabor Terstyanszky, Peter Kacsuk and Zsolt Nemeth, "An approach for virtual appliance distribution for service deployment", FGCS, 2011.
[14] Z. Cheng, Z. Du, Y. Chen, and X. Wang, "SOAVM: A Service- Oriented Virtualization Management System with Automated Configuration", IEEE Int'l Workshop on Service-Oriented System
[15] http://aws.amazon.com/cloudformation/
[16] http://www.vmware.com/products/studio
[17] http://stratuslab.eu/fp7/doku.php/claudia.html
[18] http://www.viapps.org/viapps/en/about.html
[19] OpenNebula Project, "OpenNebula: the open source solution for data center virtualization", http://opennebula.org
[20] http://marketplace.c12g.com/appliance
[21] Peter van Heusden, Long Yi and Alan Christoffels, "An OpenNebula-based cloud computing environment for bioinformatics"
[22] VMTorrent: Virtual Appliances On-Demand
[23] https://c370023.ssl.cf1.rackcdn.com/documents/Virtual_Appliance_Whitepaper.pdf
[24] http://www.nimbusproject.org/docs/2.6/clouds/clusters2.html

# A Framework for Selecting Cloud Service Providers Based on Service Level Agreement Assurance

**R.El-Awadi**[1]**, M. Esam**[2]**, M. Rizka**[3] **and A.Hegazy**[4]
[1]Information System, AAST, Cairo, Egypt
[2]Storage Solution Architect, EMC$^2$ ,Cairo, Egypt
[3]Centre of Excellence, AAST, Cairo, Egypt
[4]Information Technology, AAST, Cairo, Egypt

**Abstract** - *Cloud Computing has become a promising technology that offers a commoditized service to the software, the platform and the infrastructure where they are delivered as a service. It faces several challenges, one of which is responding to customers' requirements on-demand. This can be achieved only through creating an agreement which is referred to as the Service Level Agreement (SLA) that guarantees the customers' rights. In addition, more and more providers are currently emerging, thus it is difficult for customers to select the most reliable one. It is important to have a methodology capable of mapping customers' requirements, which are termed Service Level Objectives (SLO) in SLA, so as to determine the different criteria for selecting the best cloud providers. Thus in this work a framework is presented that acts as an index of providers and allows customers to evaluate Cloud service offerings and to rank them based on their abilities. This study intends to integrate the automated SLA negotiation among the four cloud agents with the measurement of Quality of Service (QoS), termed Service Measurement Index (SMI). Such an index should be guaranteed by the provider through the SLA which meets the specifications of the customers' requirements.*

**Keywords:** Cloud Computing, SLA, SLO, QoS, SMI

## 1 Introduction

Cloud Computing is a new trend in the IT field where the Computing resources are delivered as a service. These computing resources are offered as pay-as-you-go plans and hence have become attractive as they are more cost effective than traditional infrastructures. As customers delegate their tasks to more and more Cloud providers, it is important to have the SLA between customers and providers, which makes it a key performance indicator. Due to the dynamic nature of the Cloud, continuous monitoring of QoS is necessary to enforce SLAs.

There are various definitions of Cloud Computing proposed in the literature. One of these definitions sees Cloud Computing as a parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically managed to act as one or more unified computing resource based on SLA [1].

Cloud Computing is based on grid computing, Virtualization and service oriented computing paradigms in sharing common features [2] [3] [4]. With the increase of public Cloud providers, Cloud customers are facing various challenges which are divided into different categories, such as accountability, assurance, performance, agility, financial factors, security, and privacy. One of the accountability challenges is SLA creation and verification.

Because customers' demands are always different, it is not possible to fulfill all their expectations by the service provider; hence a balance needs to be made via a negotiation process. At the end of this negotiation process, both the provider and the customer commit to create a Service Level Agreement.

The SLA automatically facilitates the process of contract signing between the customer and the service provider that guarantees a specified QoS and enforces penalties in case of agreement violation. In addition, Service Level Management is intended to ensure that the defined service meets certain criteria that are established in the agreement. Failure to meet the terms of the agreement may lead to service level degradation [1].

The problem is concentrated in that Cloud service provider infrastructure is capable of satisfying the required needs in the customer's specifications. The objective of this paper is proposing a framework to evaluate Cloud providers through their offerings and to rank them according to their level of achieving customers' SLOs.

The paper also discusses the nature of Cloud Computing through its architecture and its distinctive characteristics in section II. It also introduces the related work in section Ш, which shows the previous frameworks that focus on the evaluation of the Cloud vendors and on finding appropriate solutions to establish confidence between the customers and the Cloud vendors' community; this is the dimension that this study's proposed framework attempts to improve by integrating the role of third party in the negotiation scenario of creating SLA, while monitoring its QoS parameters. In section IV the proposed framework design is presented together with a sequence diagram of its mechanism. Section V presents the proposed experimental study through testing the benchmark applications on platform as a service layer by using VMware hypervisor on the Cloud. Finally, a summary of the current work and the future recommended work is be discussed in section VI

## 2 Cloud Computing Characteristics and Architecture

Cloud Computing is “A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., servers, storage, networks, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [4]. This definition includes the Cloud characteristics, architecture, and deployment strategies [5] [6]. The NIST definition of Cloud Computing states that the Cloud infrastructure should possess five essential characteristics [4]. They are stated as Rapid Elasticity, On-Demand Self Service, Measured Service, Resource Pooling and Broad Network Access. A layered model of Cloud Computing describes the architectural, business and various operational models of Cloud Computing [7].

Cloud service offerings as shown in Fig.1 are classified primarily into three models: Infrastructure- as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Accordingly, there are different types of Cloud deployment models, each with its own benefits and drawbacks: [5] [7].

Cloud deployment models will be classified as Private, Public, Community, Hybrid and Virtual Clouds.

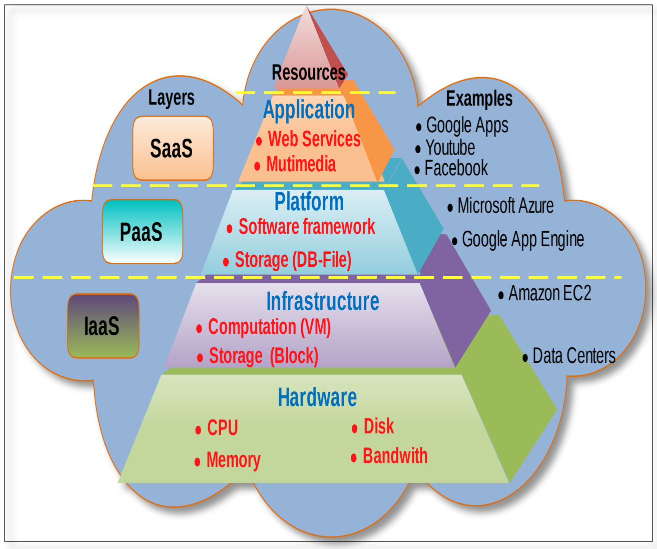


Fig. 1 Cloud Computing Architecture

## 2 Related Work

There is a wide-range of research work around the SLA for Cloud Computing. Some models of Cloud Computing are introduced to maintain the reliability between Cloud providers and consumers involved in the negotiation process. [8].

Other works focus on the revenue and Quality of Services (QoS), and some mechanisms are introduced to maximize the Cloud consumers or providers' revenues [12]. Monitoring in Cloud is also a hot topic in Cloud Computing research. Some architecture is proposed to improve the capacity of the Cloud monitor [11].

I.al, I.br, and E.sc present SLA Validation Models in layered Cloud Infrastructures [13]

M. Radi represents the Parameters for Service Level Agreements Generation in Cloud Computing [12]. Sh. Mahbub, S. Ries and M. Muh provide an overview of the important aspects that need to be considered when integrating trust and reputation concepts in Cloud Computing [14].

According to SLA Architectures, M. Al. Hamad proposes a Conceptual platform [8] which is considered the third party between the Cloud providers and the consumers. Cloud providers can advertise their services in the platform and Cloud customers can search and select the services in the service list. When the customers find the services which meet their needs, they can negotiate with the providers through the platform. From our point of view this platform does not observe the role of the negotiation process by the third party and neglects the role of SLA parameter measurement and the calculation process before monitoring, where it stores any violation found in the monitoring process.

A .Al Falasi and M. A. Serhani framework [15] enables Cloud clients to easily search through a repository of Cloud service providers, and to specify their required QoS measures. It addresses the issue of SLAs reliability and the conformity of web services. In addition, QoS measures through the use of a dedicated third-party broker for real-time testing and composition of web services on the Cloud. It is also based on formally specifying the service level objectives (SLOs) required by the client, as well as formally specifying the services' performance capabilities of the Cloud provider SLAs. Unfortunately, this framework observes the role of broker and creation of the SLA to the Web Services but it misses the measuring and monitoring of the QoS parameters.

## 3 The Proposed Framework for Service Level Agreement Selection

With the expansion of Cloud Computing; there is a high emergence of Cloud providers and Cloud resources, therefore it is troublesome for Cloud customers to pick out the most reliable providers and resources. It is vital to possess a technique that maps the customer requirements and their SLOs outlined within the SLA.

In this section, a proposed framework is presented that indexes and ranks the Cloud providers according to their offerings and integrates the automatic SLA negotiation with the measure of service quality parameters to make sure that they are secured by the Cloud service provider through the SLA that is accountable for mapping the customers' SLOs and their specifications.

This framework handles and reflects the dynamic nature of the Cloud and achieves the QoS assessment through the automated negotiation scenario between the four agents (Cloud Customer, Provider, Broker, and Carrier or SLA Generator) as shown in Fig.2.

This Framework consists of four Cloud agents and shows their roles in the negotiation process.

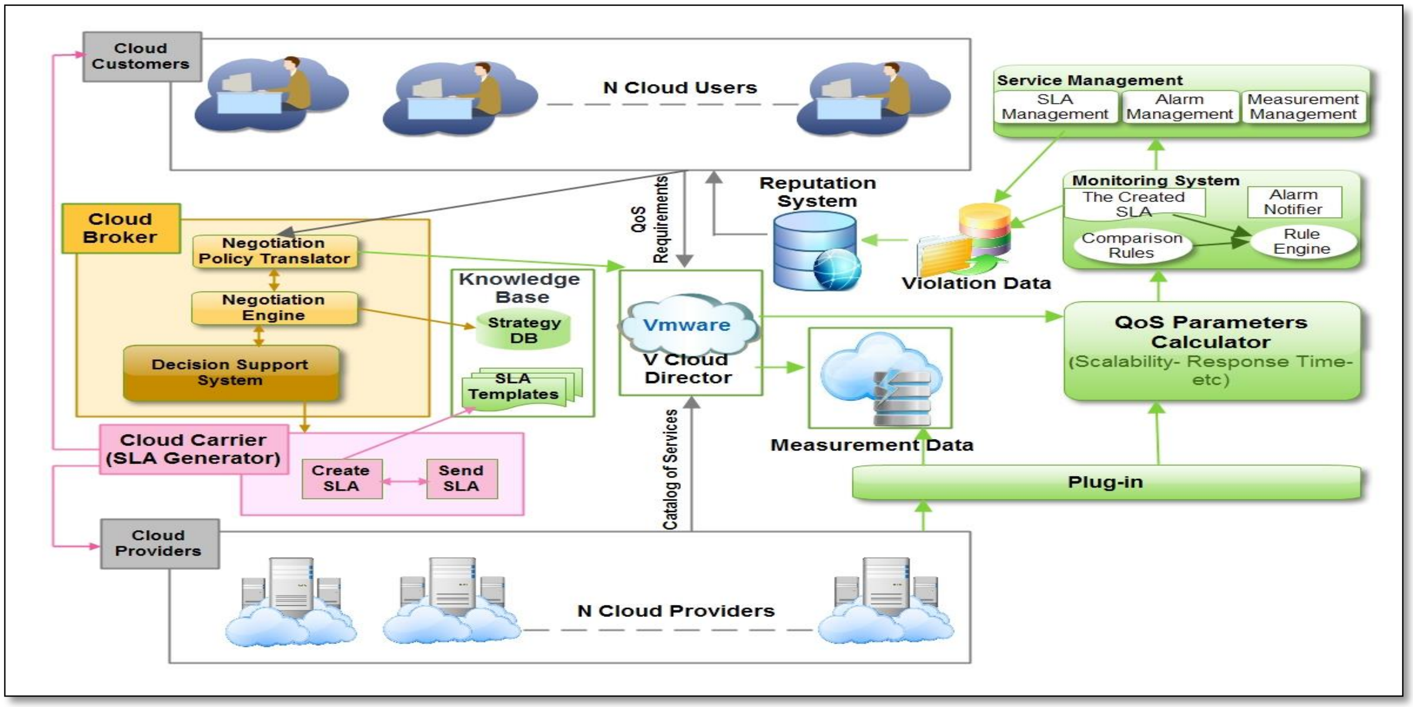


Fig. 2 Proposed Framework Components for selecting Cloud providers through SLA Assurance

It also observes the measurement of QoS parameters that should be guaranteed by the Cloud service provider through the SLA which meets the specifications from the customer-centric view into one system.

After the measurement, the role of the monitoring layer shows whether there are any violations. Such a layer is managed by the actions agreed upon in the SLA document.

This framework provides a complete view regarding the quality information of Cloud services for users and service providers in real time.

## 4.1 Framework Components

The main components in the proposed negotiation framework are; the Customer Agent (CA), the Broker Coordinator Agent (BCA), the Provider Agent (PA), the Service Provider Agent, the SLA Generator (Carrier Agent), the Directory of Cloud services, the QoS Database, the Knowledge Base (KB), the Plug in, the QoS Calculating layer, the Monitoring System and the Service Management [17].

**Customer Agent:** Represents the customer, submits requests for software services and registers their QoS requirements into QoS Data.

**Broker Agent:** Represents the third party who receives customers' requests and negotiates with the providers to achieve the required business objectives.

**Negotiation Policy Translator:** Maps customers' QoS requirements according to cloud provider service catalogs.

**Negotiation Engine**: Includes workflows which use negotiation strategies during the negotiation process.

**Decision Making System:** Uses decision making heuristics to update the negotiation status.

**Provider Agent:** Represents the provider and can include the third party monitoring system used to update the providers' dynamic information.

**The SLA Generator (Carrier Agent):** When the negotiation has been successfully completed, the SLA Generator creates a SLA between the customer and the provider using templates retrieved from the KB. The template includes specified Service Level Objectives (SLOs) according to the QoS.

**The Cloud Directory:** The repository stores the Cloud Provider services' catalogs and also the Customer QoS requirements.

**The Knowledge Base:** It represents the repository that stores negotiation strategies and SLA templates.

**The plug-in:** This is mainly used to collect service measurement information. [16]

**QoS Calculating Layer:** It is responsible for assessing the quality of service using is used to generate the QoS report.

**Monitoring Layer** It is used mainly for monitoring the violations of service quality. ***Comparison Rules*** are the conditions used to trigger an alarm. ***Rule Engine*** compares the QoS data calculated by the QoS Calculating Layer with appropriate SLA contract and Comparison Rules, to detect whether the QoS is in conformity with that in the SLA. If there is a service violation, the ***Alarm Notifier*** will alarm. The

Alarm Notifier includes various types of alarm modules, which are used for different purposes [16].

**Service Management** is a set of methods used to check whether there are any violations in any parameter. If any violation is detected, it will lead to some punitive action which is agreed upon in the documented SLA and it will be stored in the violation database.

## 4.2 Framework Mechanism.

First, the Cloud customers define their requirements to be stored in the Cloud Directory (QoS Data) Repository and the Providers provide their registered service information which is stored in the Cloud Service Catalog repository of the same QoS Data Repository.

Second, the Role Negotiation Policy Translator comes to map the Customers' QoS Parameters. The broker provides comments about the users' requirements and providers' services that are mapped to the Reputation System then sends the major parameters to both the customer and the provider according to their SLOs accepted in the negotiation.

Third, thereafter, the Negotiation Strategies are extracted from the Knowledge Base and provided to the Negotiation Engine so as to help the Decision Support System to start its role in mapping the Service Level Objectives (SLO), which defines objectively the measurable conditions of the service. Examples include the parameters of throughput, data streaming frequency and timing, as well as availability percentages for VMs through the Previous Service Level Agreements that are stored in the Service level Agreement Templates Pool which provide it to the measurement Data Repository related to the Provider Cloud Service Catalog.

Fourth, the SLA Generator or Cloud Carrier comes to create and send the agreement with defined SLOs and parameters to both the customer and the provider, which in turn will be monitored in the monitoring layer.

Fifth, the Plug In then collects service measurement information to be calculated by the QoS Parameters Calculator which extracts the QoS Parameters from the QoS Data Base. The results are then provided to the Monitoring System which checks whether the parameters are equal, under, or over the threshold defined in the SLA and whether any violations exist. If there are any violations from the providers' side related to performance in the transaction, it will be converted to the SLA Violation Data then to the Reputation System, which will finally evaluate the providers' performance and send the results to the Reputation System which will send a recommendation to the customers about the ranked providers.

The following Fig.3 shows this mechanism as a Sequence Diagram which includes both; the Cloud Customer and the Cloud Provider as an actor and the various objects related to its events.

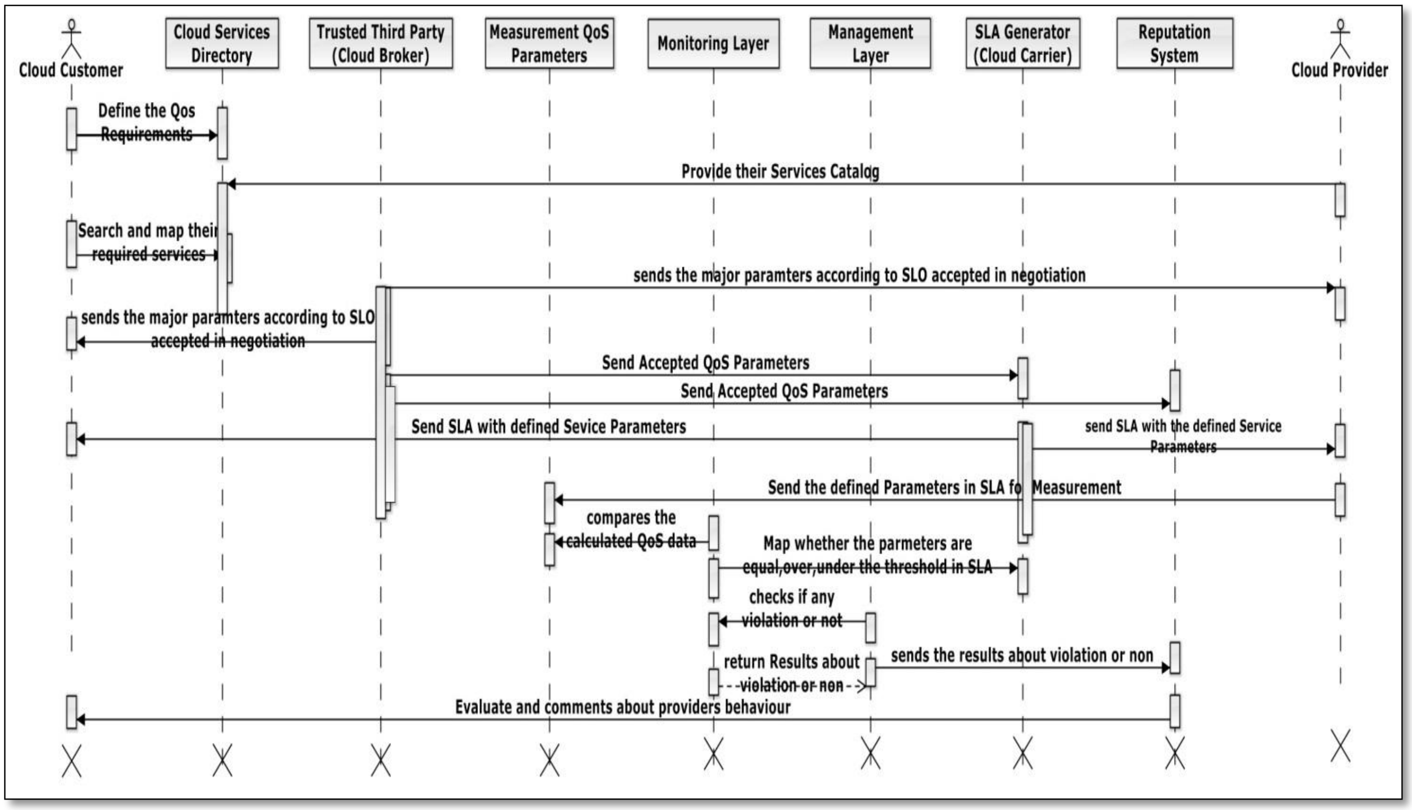

Cloud Customer
Cloud Services Directory
Trusted Third Party (Cloud Broker)
Measurement QoS Parameters
Monitoring Layer
Management Layer
SLA Generator (Cloud Carrier)
Reputation System
Cloud Provider
Define the Qos Requirements
Provide their Services Catalog
Search and map their required services
sends the major paramters according to SLO accepted in negotiation
sends the major paramters according to SLO accepted in negotiation
Send Accepted QoS Parameters
Send Accepted QoS Parameters
Send SLA with defined Sevice Parameters
send SLA with the defined Service Parameters
Send the defined Parameters in SLA for Measurement
compares the calculated QoS data
Map whether the parmeters are equal,over,under the threshold in SLA
checks if any violation or not
return Results about violation or non
sends the results about violation or non
Evaluate and comments about providers behaviour

Fig. 3 Sequence Diagram for the Proposed Framework

## 4 Experimental Study

This study is based on the proposed framework. Fig. 4 shows the testing environment that is implemented in EMC$^2$ Co.[18] through a testing lab which consists of two physical Dell servers that are connected through a Storage Area Network “SAN” with the Fibre Channel Switch to EMC VNX Storage Array; additionally the VSphere Clients work as windows programs that are used to configure the host and to operate its virtual machines. The VSphere Clients are connected via Ethernet Switch to access the hosts.

Based on the study’s proposed framework and its components, there is a Cloud Directory that contains the Cloud Provider services which are mapped to Customer requirements through the Negotiation Policy Translator. In our test this Cloud Directory is represented by “VCloud Director” which is a VMware Solution that provides a self-service portal and catalog that enables policy-based infrastructure, application provisioning and automated operations management [9].

VCloud Director improves catalog sharing, the ability to publish catalogs externally, automated versioning of catalog content, and support for storing additional file types to the catalog. It has the ability to edit virtual hardware and the ability to import VApps directly into the virtual datacenter without having to first upload them to the catalog.

Because standard a benchmarking does not exist yet to test this framework, SAP, SQL, and Oracle are considered the benchmark for testing the quality of service in EMC$^2$ as a Cloud provider of IaaS and PaaS.

It is considered that this benchmark represents the customers’ applications they own and they request infrastructure and platform as a service with the specifications in table 1.

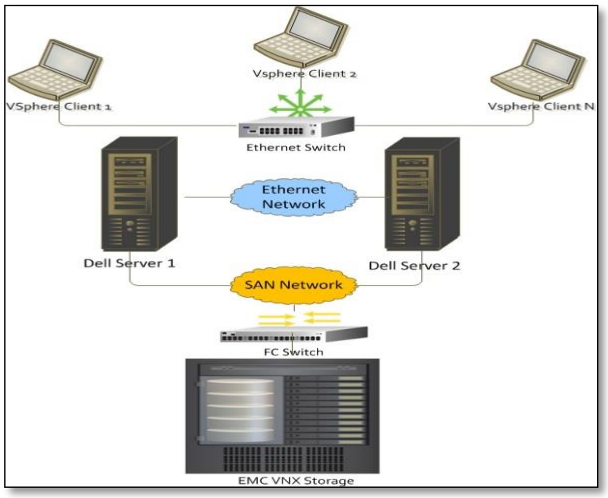


Fig. 4 Testing Lab Environment

| OS | CPU | Memory | Network |
|---|---|---|---|
| Windows 7 64 bit | 1 VCPU= 3GHZ | 2GB RAM | 1 Ethernet Adaptor with 1000 MB Base T |

Table1. Specifications of Customer’s Requirements

The basic requirements of infrastructures are represented in CPU, RAM, and Network. In addition, the windows OS is required for the platform. The three applications I/O Characteristics were tested by using I/O meter App applied on the testing Cloud. The testing was run for 6 hours and the following table.2 shows the input I/O characteristics of the benchmark applications testing.

| | SAP | SQL | Oracle |
|---|---|---|---|
| Max. IOPs | 11200 | 11200 | 11200 |
| Read Ratio % | 88 | 88 | 88 |
| IO Size (Byte) | 20500 | 8192 | 16000 |
| Testing Duration (hours) | 6 | 6 | 6 |
| IO Alignment | 1MB | | |
| Burst length | 1 IO | | |

Table2. Benchmark Testing Input Data

| | SQL | SAP | Oracle |
|---|---|---|---|
| Memory Active Average (KB) | 129060.1382 | 226166.9767 | 112972.4108 |
| Memory Consumed Average(KB) | 2497314.715 | 2986035.283 | 2602874.099 |
| Memory Active Percentage | 6.154066953 | 10.78448184 | 5.386944334 |
| Memory consumed Percentage | 119.0812452 | 142.3852579 | 124.1147089 |

Table 3. Average Percentage Result of Memory

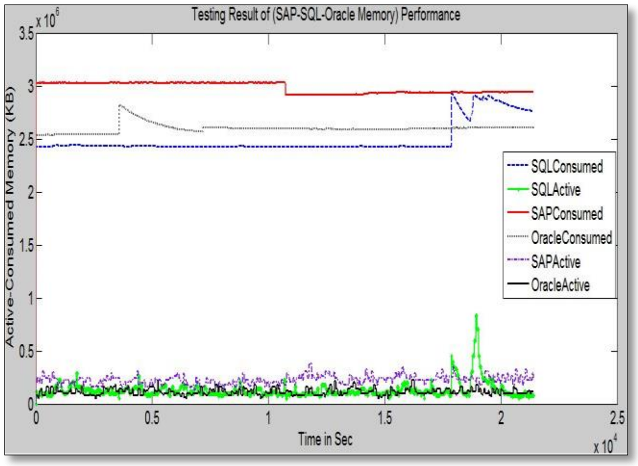


Fig.5Memory Performance of SAP, SQL, and Oracle

Fig.5 shows the performance result of memory testing data as it is one of the user required specifications. The result is represented in Consumed and Active. The consumed parameter represents the number of RAMs that the VM reserved to meet the user requirements. The Active Parameter represents the number of RAMs that have already been used by the user.

Table 3 shows the average percentage results for all active and consumed memory according to the requested size from the user's specifications.

It is concluded from the previous graph and table results that users' applications used memory less than they requested according to the percentage of active memory, which is proof that the provider reserves their request according to the consumed percentage.

This Cloud provider thus satisfies the user requirements represented in memory specifications in this case and the service was available. So, this Cloud provider meets the requested parameters defined in the SLA.

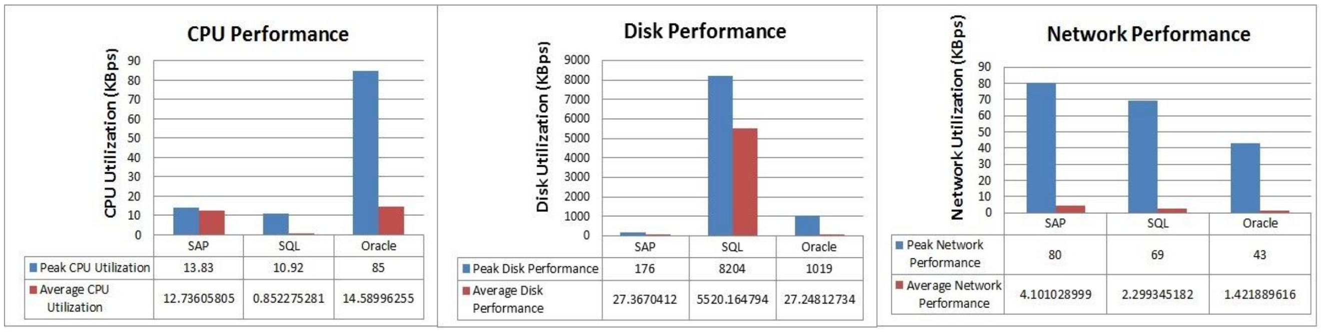
The results show that all customers' specifications of virtual CPU, Memory, Network, and Disk that are required to run their applications SAP, SQL, and Oracle are met with the defined criteria in the SLA but in different utilization performance averages according to each application, as shown in Fig. 6

The efficiency and performance monitoring of cloud services are measured by Resource Utilization. Utilization parameters of physical servers/infrastructure are an important factor in cloud monitoring. We need to collect the resource utilization data from the Virtual machines.[10] This provides a picture of how much of the VM is being utilized and the data helps in analyzing the resource utilization by applications and to decide on the scaling requirements. This can be computed as:

$$RU = \frac{amount\ of\ allocated\ resources}{amount\ of\ pre{-}defined\ resources} \qquad (1)$$

Where the denominator is defined in the SLA. The numerator is a literal amount of assigned resources from the amount of pre-defined resources for invoking the web service. The range is 0 ... 1 and higher value shows that the Web service has higher resource shares. Fig. 6 shows the peak and average utilization performance for the different benchmark applications. According to Eq.1, the amount of allocated resources is extracted from the previous testing results shown in Fig. 6 and the pre-defined resources defined Table.1, and by using the resource utilization equation for the resources (CPU, Memory and Network) the results are shown in Fig. 7

The CPU resource utilization percentage number in the different apps is measured in MHz, but in memory and network it is measured in KBps. They show how the benchmarks are utilized on the testing cloud.



| | SAP | SQL | Oracle |
|---|---|---|---|
| 🟦 Peak CPU Utilization | 13.83 | 10.92 | 85 |
| 🟥 Average CPU Utilization | 12.73605805 | 0.852275281 | 14.58996255 |

| | SAP | SQL | Oracle |
|---|---|---|---|
| 🟦 Peak Disk Performance | 176 | 8204 | 1019 |
| 🟥 Average Disk Performance | 27.3670412 | 5520.164794 | 27.24812734 |

| | SAP | SQL | Oracle |
|---|---|---|---|
| 🟦 Peak Network Performance | 80 | 69 | 43 |
| 🟥 Average Network Performance | 4.101028999 | 2.299345182 | 1.421889616 |

Fig.6 Utilization performance of SAP, SQL, and Oracle

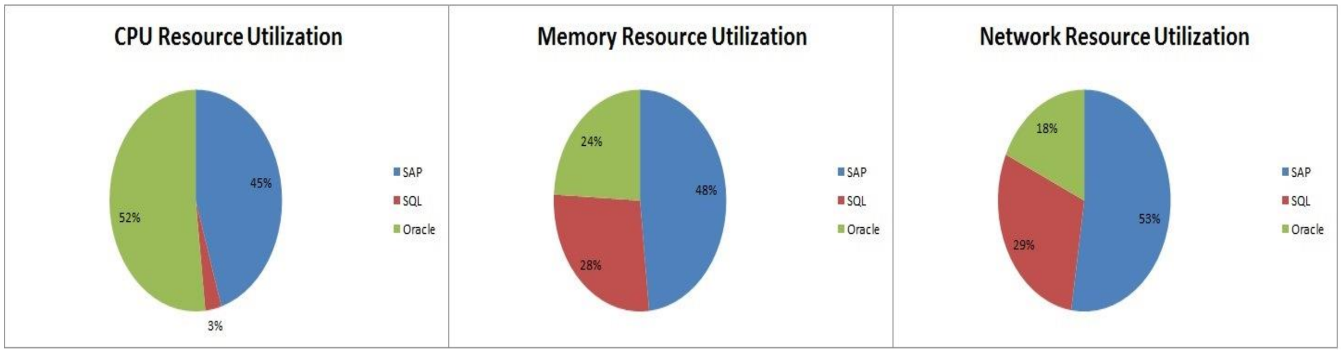


Fig.7 Resource Utilization Performance Results

## 5 Conclusion and Future Work

Cloud Computing has become a promising paradigm in the IT industries where all technologies are offered as a commoditized service. Currently, there are many Cloud providers who offer different Cloud services. It is important to have a flexible methodology that can handle and manage the SLAs in the context of Cloud Computing.
As indicated, the study’s proposed framework focuses on establishing confidence between customers and Cloud providers through an evaluation of their service offerings. The study shows the customers’ specifications through their required infrastructure, and platform as a service is defined and mapped to the SLA by using the VCloud director. Additionally, the experimental testing of the customers’ applications on this cloud test assures having the QoS parameters that are defined in the SLA.
Future studies will aim to extend the ranking methodology to include more than one Cloud provider in order to select the best one from the customers’ perspectives.

## Acknowledgments

This work is achieved in cooperation with EMC Egypt COE. Thanks for the support provided by the Academic Alliance Team.

## 6 References

[1] Hammadi, Adil M., and Omar Hussain. "A framework for SLA assurance in cloud computing." *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*. IEEE, 2012.

[2] Zhang, Shuai, et al. "The comparison between cloud computing and grid computing." Computer Application and System Modeling (ICCASM), 2010 International Conference on. Vol. 11. IEEE, 2010.

[3] Wei, Yi, and M. Brian Blake. "Service-Oriented Computing and Cloud Computing: Challenges and Opportunities." *IEEE Internet Computing* 14.6 (2010).

[4] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." *National Institute of Standards and Technology* 53.6 (2009): 50.

[5] Dillon, Tharam, Chen Wu, and Elizabeth Chang. "Cloud computing: issues and challenges." *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. Ieee, 2010.

[6] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of internet services and applications* 1.1 (2010): 7-18.

[7] Borges, Hélder Pereira, et al. "Automatic generation of platforms in cloud computing." *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012.

[8] Alhamad, Mohammed, Tharam Dillon, and Elizabeth Chang. "Conceptual SLA framework for cloud computing." *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*. IEEE, 2010.

[9] Krieger, Orran, Phil McGachey, and Arkady Kanevsky. "Enabling a marketplace of clouds: VMware's vCloud director." *ACM SIGOPS Operating Systems Review* 44.4 (2010): 103-114.

[10] Lee, Jae Yoo, Jung Woo Lee, and Soo Dong Kim. "A quality model for evaluating software-as-a-service in cloud computing." *Software Engineering Research, Management and Applications, 2009. SERA'09. 7th ACIS International Conference on*. IEEE, 2009.

[11] Comuzzi, Marco, et al. "Establishing and monitoring SLAs in complex service based systems." *Web Services, 2009. ICWS 2009. IEEE International Conference on*. IEEE, 2009.

[12] Vision, A. Client-Centric. "Parameters for Service Level Agreements Generation in Cloud Computing." *Advances in Conceptual Modeling* (2012): 13.

[13] Haq, Irfan Ul, Ivona Brandic, and Erich Schikuta. "Sla validation in layered cloud infrastructures." *Economics of Grids, Clouds, Systems, and Services*. Springer Berlin Heidelberg, 2010. 153-164.

[14] Habib, Sheikh Mahbub, Sebastian Ries, and Max Muhlhauser. "Cloud computing landscape and research challenges regarding trust and reputation." *Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2010 7th International Conference on*. IEEE, 2010.

[15] Al Falasi, Asma, and Mohamed Adel Serhani. "A framework for sla-based cloud services verification and composition." *Innovations in Information Technology (IIT), 2011 International Conference on*. IEEE, 2011.

[16] Bao, Dongmei, et al. "A method and framework for quality of cloud services measurement." *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*. Vol. 5. IEEE, 2010.

[17] Liu, Fang, et al. "NIST cloud computing reference architecture." *NIST Special Publication* 500 (2011): 292.

[18] www.emc.com

## SESSION

## POSTERS

### Chair(s)

**TBA**

# Research and Implementation of a New Cloud Server

**Hua Nie[1], Xiaojun Yang[2], Chaoqun Sha[3], Yanping Gao[4], and Keping Long[5]**
[1, 3, 5]School of Computer and Communication Engineering, University of Science and Technology Beijing. Beijing, China
[2]Dawning Information Industry Co., Ltd. Beijing, China
[4]Institute of Computing Technology, Chinese Academy of Sciences. Beijing, China
[4]Loongson Technology Co., Ltd. Beijing, China

**Abstract -** *Instead of all using TOR schemes, an approach building a cloud server on top of a high-performance fabric is presented in this paper. The advantage is to provide both high performance/cost and performance/Watt compared with the existing method. A FPGA-based system controller integrated with shared networking, shared storage, and interconnect fabric controller is designed and implemented to interconnect a set of lightweight server processors for building a high-density server. All the processors can share the networking and storage resources through an inter-system interconnect fabric. For the 64-processor prototyping system, the evaluating results show the cloud server not only keeps some traditional cluster advantages such as OS compatibility, but also achieves the better scalability, high performance/cost and high performance/Watt for workloads.*

**Keywords:** Cloud Computing; Cloud Sever; Shared Storage; Shared Networking; Inter-System Interconnect Fabric

## 1 Introduction

Cloud computing represents 11% of the market in 2013, expected to grow to 17% by 2014. The growth in cloud leads to growth in servers in data center and new requirements for servers. Cloud is redefining traditional servers. We are paying attention to the research and implementation of a new server to be assembled in data center in future, which can better meet the requirements coming from data center[1]. The targeted server is named Sugon[2] cloud server here to distinguish from others. The remainder of the paper introduces the concept and scheme of Sugon cloud server. A resulting prototyping system is discussed together with the evaluation.

## 2 Concept of Sugon cloud server

The traditional server system is a cluster of server nodes, the dedicated local storage and connected over an Ethernet network. These server nodes use their directed-attached-storage as scratch/swap space and use a storage server on the Ethernet network for primary storage. In the cloud era, on one hand, optimized TCO, compute efficiency, and fastest growing server segment will grow to dominate the server trends. On the other hand, cloud deployment models, big data analytics, and data center virtualization are driving highly evolving parallelized workloads. The servers in large-scale data centers require high density, high performance/cost and high performance/Watt [1]. Furthermore, the rapid growth in dense compute shows dense compute clusters are the future of volume servers for cloud computing [2]. According to the above mentioned requirements for cloud server, the concept of Sugon cloud server targeted to maximum efficiency in Figure 1 can be concluded as the following aspects.

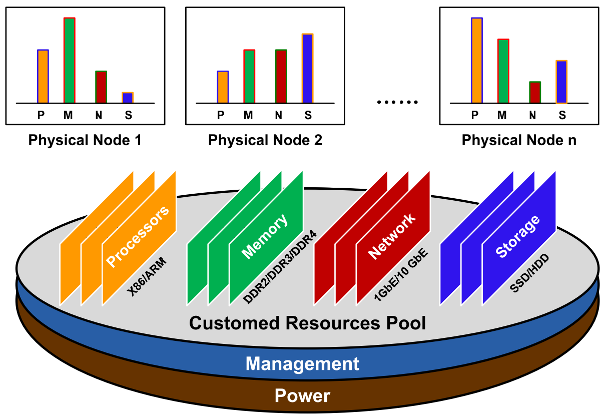


Fig. 1 Concept of Sugon Cloud Server

- *Sharing System Resources:* Processor, memory, storage, network, power, cooling, infrastructure and management can be shared by all the computing units.
- *Very High Processor Densities:* Compared with the traditional server chassis, more processors can be integrated into the chassis.
- *Very Low Processor Power:* The lightweight or single-chip processors will be adopted to better meet the demands of different workloads.
- *Highly Configurable to Computing:* The architecture can be reconfiguration according to workloads flexibly.

## 3 Architecture of Sugon cloud server

The goal of Sugon cloud server is maximum efficiency under minimal standed capacity. So the traditional cloud

[1] The work is supported in part by the National High-Tech Research and Development Plan of China under grant numbered 2013AA01A209.
[2] Sugon is a logo of Dawning Information Industry Co., Ltd.

server architecture based on TOR is unsuitable. To achieve the goal and keep to the above mentioned concept of Sugon cloud server, a new cloud server architecture is innovated as Figuire 2 shown.

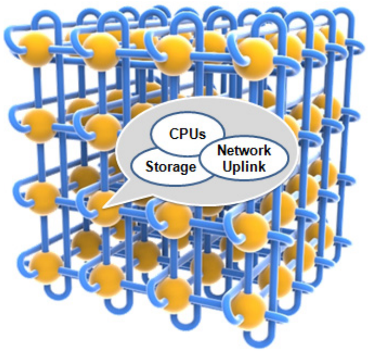


Fig. 2 Architecture of Sugon Cloud Server

Different from Seamicro microserver, Sugon cloud server achieves a distributed architecture as follows. On one hand, a high performance direct network is built to strongly support the share mechanism of cloud servers, and to improve the remote I/O access performance. On the other hand, the share mechanism of Sugon cloud server consists of two levels. Processors can share their local I/O symmetrically, and also share the remote I/O through a 3D torus interconnect fabric. A FPGA-based system controller integrated with shared networking, shared storage, and interconnect fabric controller is designed and implemented to interconnect a set of lightweight server processors for building a high-density server. All the processors can share their local and remote I/O.

## 4 Implementation and evaluation

A 64-processor prototyping system is implemented to evaluate the concept and architecture of Sugon cloud server, and validate some key technologies such as the FPGA-based system controller and the inter-system interconnect fabric.

As Figure 3 shown, the inter-system interconnect fabric is a 4x4 2D torus. Each point in the fabric is a compute module, which consists of four processors, one HDD/SSD, one Ethernet uplink, one BMC, and one FPGA used as the system controller. For example, uplinks except that of $P_{00}$ are all disabled. All nodes can share the $P_{00}$ uplink. For shared storage, it is as the same as that of shared networking.

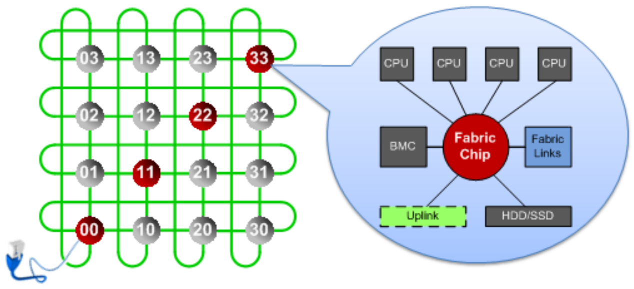


Fig. 3 Prototyping System Fabric

The prototyping system has been implemented in hardware as Figure 4 shown. It consists of *SDCompute*, *SDControl*, and *SDConnect* modules.

- *SDCompute:* A compute blade integrates with 4 Intel G2100T processors and 32GB ECC Reg memory. There are 4 *SDComputes* in the prototyping system. *SDCompute* can be up to 16. Each *SDCompute* can be designed to support up to 1 to 8 X86 or ARM processors and various memory configs.
- *SDControl:* A system controller board integrates with a FPGA-based system controller. The system controller contains shared storage, shared networking, and interconnect fabric controller. On one side, *SDControl* connects with *SDCompute* through Ethernet links and SATA links. On the other side, it interfaces local HDD/SSD and Ethernet uplinks.
- *SDConnec*t*:* It is a baseboard used to connect with all *SDControls*. S*DControl* connects with *SDCompute* directly by the board-to-board way.

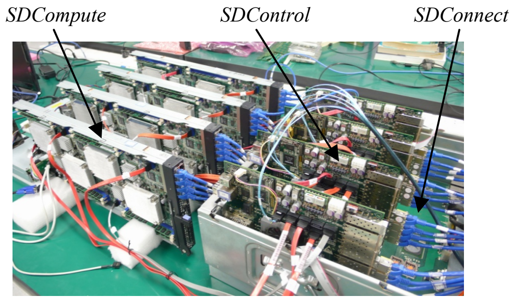


Fig. 4 A Photo of a 16-Processor Sugon Cloud Server

The evaluating results show Sugon cloud server is a dense, high-performance cluster with shared storage and networking features. All processors can share a physical HDD/SSD, and share one or more Ethernet uplinks through the Ethernet over the direct interconnect fabric. The software successfully installed and run includes CentOS 6.2, Linux-2.6.32-220.e16.x86_64, Clusconf-1.5.4, Hadoop-1.2.1, JDK-1.8.0, OpenMPI-1.6.5, Linpack HPL-2.0, Gridview2.65 cluster management tools, and Cloudview (one of cloud OS).

## 5 Conclusion and future work

The 64 processors scalability for the first generation Sugon cloud server has been implemented. It will be up to 320 processors scalability in a 44U rack during the next generation in this year. Companioning the developing trends of processors and workloads, we will scheme the cloud server system on top of the validated architecture and key technologies in this generation,

## 6 References

[1] M.A. Raza, S. Azeemuddin. "Multiprocessing on FPGA using light weight processor". CONECCT.2014, p:1-6, 2014.

[2] Rahul Kulkarni, "Microservers: Target Workloads and Architecture Trends". Technical Report, Intel, April 2013. [Online]. Available: http://www.intel.com/go/idfsessionsBJ.

# Comparison of system monitoring tools for large cluster system

**Sung-Jun Kim**[1], **Joon Woo**[1]
[1]Supercomputing Center, Korea Institute of Science and Technology Information

**Abstract** – *For system management, System administrators always check system log and service status. To reduce these efforts, there is various open-source system monitoring tools. As system sizes are getting larger, the performance of these tools is getting important. In this paper, we compare open-source monitoring tool's performance that most popular in the world – nagios and icinga .*

**Keywords:** nagios, icinga, system monitoring, cluster

## 1 Introduction

To stable manage large cluster systems; administrator should be recognized service status and failure as soon as possible. Normally, administrator are watching log periodically or use various monitoring tools to detect failures.

Nagios has been used at KISTI supercomputing center during the last 7 years; it offers quite a lot of features. But, new systems were installed in accordance with a steadily. Recently, nagios master server that collects monitoring data from remote hosts was not work properly. So, we need a more reliable and scalable monitoring tools and open source.

In this paper, we were evaluated performance of some tools before change it. We were compare two open source monitoring tools; Nagios and Icinga. Nagios is world famous and Icinga is a fork of nagios and backward compatible.

## 2 Backgrounds

### 2.1 Nagios

Nagios is an open source computer system monitoring, network monitoring and infrastructure monitoring software application. Nagios offers monitoring and alerting service for severs, switches, applications and services. It alerts the users when things go wrong and alerts them a second time when problem has been resolved [1].

Figure 1 shows Nagios architecture. Nagios core is the monitoring and alerting engine that serves as the primary application around which hundreds of Nagios projects are built. It serves as the basic event scheduler, event processor and alert manager for elements that are monitored.

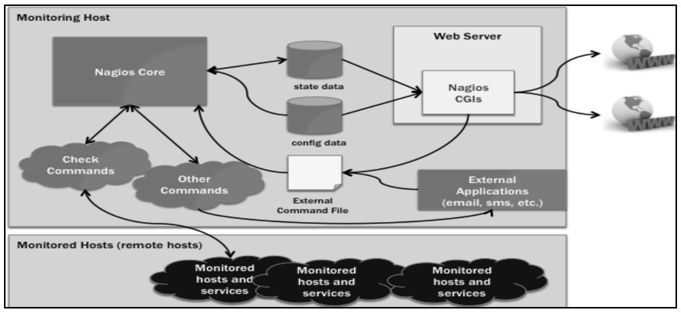

Monitoring Host
Nagios Core
state data
config data
Web Server
Nagios CGIs
Check Commands
Other Commands
External Command File
External Applications (email, sms, etc.)
Monitored Hosts (remote hosts)
Monitored hosts and services
Monitored hosts and services
Monitored hosts and services

Figure 1 Nagios architecture

Plugins is used to verify services and devices. All Nagios host and service checks are performed by external plugins. A plugin command will be invoked by Nagios core as required, with arguments as specified in the command definition that was used.

### 2.2 Icinga

Icinga is an enterprise grade open source monitoring system which keeps watch over networks and any conceivable network resource, notifies the user of errors and recoveries and generates performance data for reporting [2].

Icinga is a fork of Nagios and is backward compatible. So, Nagios configuration, plugins and addons can all be used with Icinga. Though Icinga retains all the existing features of its predecessor, it builds on them to add many long awaited patches and features request by the user community.

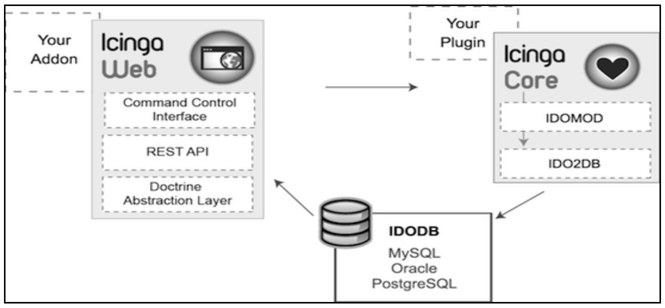


Figure 2 Icinga architecture

Figure 2 shows Icinga architecture. Like Nagios core, Icinga core does not check any services and hosts status. It is scheduling and processing of events and handle with alerts. Icinga used modern it techniques like Web 2.0 for web interfaces, mobile UI and supports Oracle and PostgreSQL.

## 3 Comparison

### 3.1 Test environments

We were built test environments using KVM to increase number of hosts dramatically. Our test servers are 17 nodes; Intel Xeon Quad Core 2.66 GHz, 4 GB memory. Test servers are consisting of three parts.

- Measure server: It is Ganglia server which checking performance of master nodes cpu, memory, I/O rate, etc.
- Master server: It is master server which collecting client server's status information.
- Client server: It is remote hosts that send their service status data to monitoring master.

Master and client servers have VMs using KVM. Master server has five VMs for combination of Nagios, Icinga and Mysql. Client servers have 9 VMs for NRPE (Nagios Remote Plugin Executor) to check their service status and report to master server. Finally, one physical node has 10 client nodes (1 domain server + 9 guest server). As a result, we built 150 virtual client servers using 15 physical servers. Each client server check 25 service status and master server is collecting about 4,000 services status check result from clients. Figure x. show architecture of testbed.

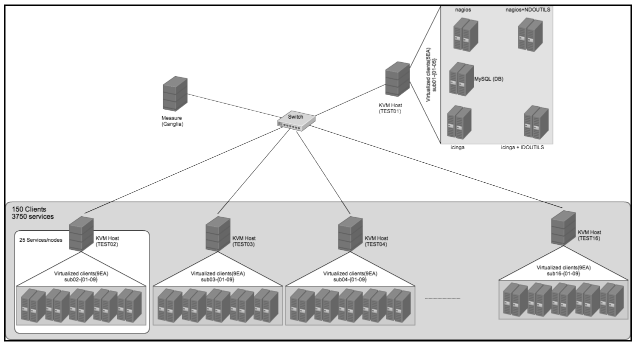


Figure 3 Architecture of Monitoring tools comaprison Testbed

### 3.2 Performance evaluations

As mentioned pervious section, we were using Ganglia to measure server side overload for each monitoring tools. We were tested Nagios and Icinga, with/without using database broker that stored status data to database.

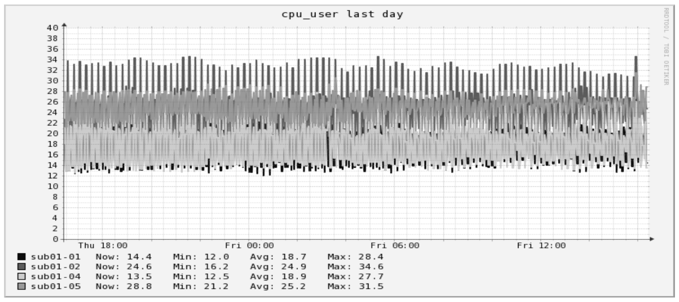
For each case, testing was progressed during the week. During a test, the other monitoring servers were halt to avoid effect between test servers. We only used default setting to compare under same condition.

Table 1 Test server information

| hostname | contents | hostname | contents |
|---|---|---|---|
| sub01-01 | Nagios only | sub01-04 | Nagios/DB |
| sub01-02 | Icinga only | sub01-05 | Icinga/DB |

Table 1 is describing installed monitoring tools on each test servers.

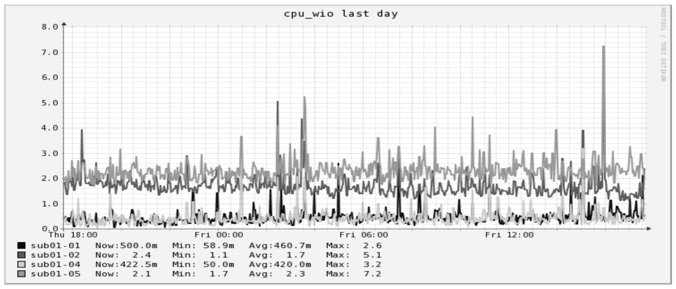
Figure 4 shows cpu_user metric from Ganglia. It means CPU utilization used by user processor.



```
■ sub01-01   Now: 14.4   Min: 12.0   Avg: 18.7   Max: 28.4
■ sub01-02   Now: 24.6   Min: 16.2   Avg: 24.9   Max: 34.6
□ sub01-04   Now: 13.5   Min: 12.5   Avg: 18.9   Max: 27.7
■ sub01-05   Now: 28.8   Min: 21.2   Avg: 25.2   Max: 31.5
```

Figure 4 cpu_user metrics in Ganglia

Figure 5 shows cpu_wio metric from ganglia. It means the time that processor wait for I/O.



```
■ sub01-01   Now:500.0m   Min: 58.9m   Avg:460.7m   Max:  2.6
■ sub01-02   Now:  2.4    Min:  1.1    Avg:  1.7    Max:  5.1
□ sub01-04   Now:422.5m   Min: 50.0m   Avg:420.0m   Max:  3.2
■ sub01-05   Now:  2.1    Min:  1.7    Avg:  2.3    Max:  7.2
```

Figure 5 CPU_WIO metrics in Ganglia

Above Figures, Nagios has used fewer CPU resource and shorter I/O wait than icinga generally. In case of using database broker, usage of CPU resource was lower and I/O wait time was longer.

## 4 Conclusions

When we were deciding to compare to tools, we had expected that performance of icinga is better than Nagios. We have planned to migrate Nagios to icinga. However, the results were entirely opposite.

So, we are planning to change configuration Nagios for large installation tweaks instead of migrate Nagios to icinga. In the future, we will evaluate convenience of SQL queries and response time to get some data on GUI interfaces. And we are evaluating new version of these tools continually.

## 5 References

[1] Nagios Offical website, "http://www.nagios.org"

[2] Icinga Offical website, "http://www.icinga.org"

## SESSION

## LATE PAPER: CLOUD AND TRUST

### Chair(s)

**TBA**

# Surveying Trust in the Cloud

**R. Goel, PhD[1], J. Nies[2], and M. Garuba, PhD[2]**
[1]Information Systems and Supply Chain Management, Howard University, Washington, DC USA
[2]Systems and Computer Science, Howard University, Washington, DC USA

**Abstract -** *Cloud computing has become a force multiplier for organizations who realize the benefits of shared computing platforms and services because of their convenience, dynamism, elasticity, and scalability to meet the growing demands of organizations. Control and visibility are central tenets to building trust in cloud computing. One critical factor to cloud success is how the infrastructure and policies for control and visibility are managed and presented to the customer. A well-developed trust architecture based upon a strong trust model establishes trust in the cloud. The intent of this paper is to extensively review exiting literature in order to conduct a holistic study of the challenges of establishing trust in cloud computing. Additionally, analysis of the current trust architectures and trust models will form the basis for a comprehensive trust architecture that addresses the concerns with control and visibility.*

**Keywords:** Cloud, Trust model, control, Visibility

## 1 Introduction

Over the last five years, cloud computing has went from a niche idea to a mainstream force enabler for organizations at all levels. It is doubtful that cloud computing may fully replace the typical enterprise information network since it, in itself, is a very attractive paradigm because of the associated economic and operational benefits [18]. This growth and continued interest in cloud computing warrants a greater interest in defining and establishing trust within the cloud between all the interacting entities.

Cloud computing facilitates on-demand access to a shared pool of computing resources that can be scaled to meet the customers computing needs and requirements. This scalable structure promotes greater collaboration and communication within the organization due in part to the richness of the offered web services and capabilities. It is not uncommon for applications from different users to simultaneously share cloud resources as a result of the scalability and agility of the cloud computing architecture [1]. Moreover, we are seeing more and more organizations literally moving large chunks of their business to an external cloud provider. A great example of this trending towards the cloud is illustrated by Washington, D.C.'s city government moving "all 38,000 city government employees have unlimited access to Google documents and services such as Gmail." [7] But, a paramount concern among cloud computing users involves placing trust in the cloud computing platform. In particular, [12] emphasizes that enterprises consigning their data to cloud computing actually creates two folds of a complex trust relationship.

Control and visibility are central tenets to building trust in cloud computing, especially how they are managed and presented to the customer. The customer needs to be assured that they have control and ownership over their data; no matter where it is physically located. This assurance extends to security and prevention measures that facilitate the confidentiality, integrity and availability of the customer's data. Building trust in the cloud requires measuring how to achieve greater control over and visibility into the cloud's infrastructure, identities, and information." [6] Although a well designed and developed trust architecture based upon a strong trust model is the key to establishing trust in the cloud, the obstacle is customer's buy-in. As such, the intent of this paper is to analyze current trust architectures and models noting their strengths and weaknesses that will in turn be used to formulate a trust architecture that addresses concerns over control and visibility within the cloud.

## 2 Background: Trust Relation

Trust in the context of cloud computing is a complex process which requires all participants to disclose volumes of information about themselves. The authors of [13] indicate how participation in cloud computing is "universally required to accept the underlying premise of trust. Vast types of entities interact and share with each other within the cloud, yet forgo assurances that these entities can be trusted. The collaborative nature in which these entities interact with each other "is only productive if all participants operate in an honest manner." [1]

Adding to this complexity is the association of attributes to an entity that can quickly multiply also need to be taken into account [8]. Moreover, the authors of [16] suggest entities will transition through various states while interacting with the cloud or vice versa. The combination of the entity transitions with entity states to the trust negotiation process present further challenges to the assurance of trust within the cloud. These entities comprise virtualization instances, cloud provider's infrastructure, and the various identities associated with users, devices, applications, and systems.

Kramer in [13] note that there are three distinct aspects that determine the concept of trust in a system:

- Trust relations: The interaction between two entities in which they believe or know that the other entity is operating in an honest manner.
- Trust domains: A community of mutually trusting entities in which there is a universal belief and a sharing of knowledge takes place among all entities within the community.
- Trust management: The organization of trust relations into trust domains and ensures the flow of trust negotiation between all participating entities.

Apparent from these three distinct aspects is that trust is very much based on the establishment of building a reputation amongst all participating entities. Trust relations can be further broken down to incorporate potential trustees, since each trust relation will begin with the two entities not knowing each other.

As such, reputation management has an important role in establishing the cooperative trust relations between entities [2]. Reputation schemes basically reach out to other peers of a particular entity to infer trust towards this potential trustee. More than likely what occurs is an aggregation of the inferences, which are in turn used to build a trusted entity. This newly trusted entity would then become part of a larger community of trusted entities forming a trusted domain. As these trusted domains interact with each other they effectively build a web of trust. However, organizations are cognizant of the fact that the ability to find trustworthy partners is critical to an agent's success; untrustworthy agents may deliver an inferior service [11]. Furthermore, [11] effectively demonstrates attacks on vulnerabilities by successfully using them against a number of existing Trust and Reputation System (TRS) proposals. Thus we conclude that trust relations, in effect, are fragile by nature. The next section will briefly analyze several trust models that will be used to develop a more sound trust architecture for the cloud computing environment.

## 3 Trust Models for the Cloud

The previous section illustrated some of the challenges of trying to implement trust architecture in the cloud computing environment. What is important, however, a trust model must take into account all entities that make up the cloud computing environment to support the chain of trust which in turn creates a web of trust. More importantly, because establishing trust is reputation based there needs to be processes that prevent tampering with the reputation information that is shared between peers. Some experts suggest the incorporation of policies and credentialing into the trust model while others suggest adding time-stamp hashing capabilities. The result, according to many of the experts, is a tamper-proof trust negotiation environment that will lead to greater assurances of the cloud computing environment. The following section will briefly analyze several trust models that will be used to formulate a trust architecture that addresses concerns over control and visibility within the cloud.

### 3.1 HiTrust Trust Negotiation Service

HiTrust is a trust negotiation model that is based upon a hybrid tree model. Within this model policies and credentials are embedded into the tree nodes that give greater assurance on trusting a particular node. According to [14], the result "can be a gradual evolution of trust relations through the interactive disclosure of credentials and security policies." An entity would begin the trust negotiation process by requesting a service from another entity, which in this case would be a service within the cloud. This in turn would launch a HiTrust agent that builds a Hybrid Tree and policy stack that are essential to the negotiation process. It is important to note that the HiTrust agent facilitates the disclosure of entities credentials which are tied to policies and are based on a minimum credential set. The minimum credential set determines the success of a session request as defined, "taking attribute value in credential and context information of the negotiation session as an input and making the security policy always true." [14]

The HiTrust model is comprised of five key modules: Message agent, negotiation session manager, negotiation strategy controller, credential/policy parser, and credential chain constructor. The message agent implements message encapsulation that contains session identification, credential, policy, and meta-information that will be used by the other modules to establish trust relations between two entities. The negotiation session manager is "responsible for the negotiation state maintaining," because each session will have different negotiation states controlled by the Hybrid Tree and policy stack [14]. The negotiation strategy controller maintains the control of credential and policy disclosure. What is important to note is the disclosure of these attributes is based on various negotiation strategies for a particular session. HiTrust relies on the X.509 v3 credential and the XACML format for security policies. Credentials and policies are maintained and verified by the credential/policy parser. Finally, the credential chain constructor serves two purposes: verify the credential chain and construct credential chains to satisfy specific security policies. [14]

HiTrust's tying of credentials to security policies give the assurance of a tamper-proof session. However, [14] also shows the time to execute trust negotiation in the HiTrust model "increases linearly with the number of concurrent requests. This linear growth in time, is a bit troublesome, is the result of the growth of the number request. This outcome does not make for a very efficient trust model for cloud computing even though the authors argue that HiTrust would scale to meet the trust negotiation requirements of cloud computing.

## 3.2 Trust Model in Hybrid Computing Environment

Jemal Abawajy in [2] points to a common issue with cloud computing, “users and computational agents and services often interact with each other without having sufficient assurances.” To counter this issue, Abawajy believes a reputation system that can efficiently integrate various attribute information about an entity can influence the trust negotiation process. But as was the case with HiTrust, the integrity of the trust negotiation process is important if not vital because of the influence of “false recommendation can result in committing a transaction with untrustworthy peers,” and the incorporation of reputation feedback management will mitigate the dishonesty of entities, while isolating negative behaviors in the cloud computing environment [2].

An integral process of Abewajy's trust model is the reliance on peering arrangements that are established between entities. Peering arrangements are defined in Service Level Agreements (SLA) which describes security policy attributes that must be met before an agreement is achieved. The cloud resource manager and inter cloud broker establish and manage these peering agreements, as well as provisioning of resources for the requested session. The core of the model, however, is the trust manager. This trust manager is entrusted with the task of collecting and maintaining reputation rating, honesty rating and personal experience rating about the peering arrangement. The personal experience rating is based upon first hand information and is shared among peers within the peering arrangement.

The reputation rating, however, is a confidence based rating and affects the trust update process of the trust manager. Unlike the personal experience rating, the reputation rating remains private but is factored into the negotiation process. Both ratings are used in the form of a personal feedback rating that can be queried by all participating entities. The feedback rating also includes fading factor that decreases the trust confidence level over time. A final element of the feedback management is the incorporation of a filtering process that establishes a threshold for separating trusted entities and untrusted entities.

The important take away from Abewajy's trust model is the use of a reputation-based system that incorporates a feedback filtering process to mitigate the propagation of dishonest ratings. More importantly, however, is users and cloud providers can be assured of guaranteed trust as the model defends against malicious information. Yet, the sharing of the reputation information as a result of the peering agreement lacks trusted communication path in the form of certificates, which could lead to data being manipulated in transit.

## 3.3 Trusted Platform-as-a-Service

Brown and Chase in [4] note that users of cloud services have no assurance of trust beyond the assurance of the service provider. In their model the cloud provider is considered a neutral Trusted Third Party (TTP). Trust in the provider is established using a reputation scheme similar to that offered by Abewajy. Each TTP would employ a combined trusted platform with trust management that attests to the identity of software being run by the cloud provider. The trusted platform in effect issues a digitally singed assertion that the software instance identity can be trusted. The authors add another level of assurance through “the concept of of instance 'sealing',” which prevents the launched instance from being modified by any user [4]. What the authors have done is to mitigate a number of attack vectors that have been shown to be exploited in *Top Threats to Cloud Computing*. Finally, users can combine assertions from multiple sources to add more control and visibility to assigning trust to a running instance.

## 3.4 A Novel trust management system architecture

The trust management model proposed in [8] reflects the multi-faceted nature of trust assessment by considering multiple attributes, sources, and roots of trust. These multiple attributes form the basis for making a reliable decision on whether to trust an entity or not. But the authors contend that the source of the attribute is an important factor to consider because of the quantitative and qualitative information that can be factored into the trust establishment process. [8] Thus, their model bases trust on the subjective probability of attributes in the form of a trust metric. This is accomplished through an opinionated expression that an entity believes the service delivered meets a certain quality. A TTP can be included, whose opinion, to add validity to the trust negotiation.

Habib et al. trust management model is comprised of five components. The Registration Manager is a registry for service level provisions promised by the provider. This registry information is forwarded to the Consensus Assessments Initiative Questionnaire Engine and is included with the competencies of different attributes that is provided by the cloud service provider. The Trust Manager formulates a trust score based on input from the Trust Semantic Engine, Trust Computation Engine, and user requirement and opinion information. A strong point of the Trust Manager is its support for trust customization and evolution empowering the user to specify preferences as they relate to their business model. The Trust Semantics Engine and Trust Computation Engine work together with the Trust Manager to integrate the formal framework of the Habib et al. model. Essentially, these three components work through the logics of the trust negotiation process. Finally, the Trust Engine Update filters opinions to assist users in validating the trustworthiness of an entity.

The Habib et al. model does not ignore the fact that cloud computing, in general, has multiple entities that interact with one another. And as such, multiple attributes are associated with these interactions. Empowering the user with the capability to customize their trust assertions on the merits of "subjective interests and requirements," is an important step [8]. However, as authors of [18] indicate, this model does not consider the providers also need to have some level of trust on the users to whom to release their services.

## 4 Comprehensive trust architecture

The previous section illustrated important factors to consider when dealing with trust management in the cloud. The statement that trust in the cloud is a complex and difficult process does not fall on deaf ears. The on-demand, elastic, and scalable architecture requires a trust model that incorporates every aspect of the cloud computing model and the multiple interactions. These very interactions between the entities can also be used to determine if the trust is direct or indirect [19]. The trust architecture to be proposed will incorporate many of the elements described in the models covered in the previous section. The explanation, however, will be a top-level explanation of the components of the trust architecture.

### 4.1 Trust management framework

**Reputation Manager:** The Reputation Manager will manage the reputation metrics, which are based on the opinions of an entity towards another entity.

**Policy Manager.** The Policy Manager manages the security policy established by the cloud provider. Additionally, the policy manager manages each users SLA.

**Trust Engine:** The Trust Engine compares the reputation metric against minimum reputation metric that determines if the reputation should be considered as weak or strong. The Trust Engine also compares attributes from the policy manager that determines if the trust negotiation between two entities should proceed. Finally, it receives trust session state information input from the Global Trust Manager. These attributes will be used by the Local Trust Manager to establish or kill a trusted session.

**Trusted platform module:** A Trusted Platform Module (TPM) will be used for remote attestation of all trust negotiation sessions and signs the sessions with a unique endorsement key. An additional use of the TPM is to attach a hashed value to the attributes exchanged between the peering entities. [15] points to a small issue with the use of a TPM, "the latency," to complete an attestation is "unacceptable when attestation is performed regularly."

**Trusted Third Party:** The idea behind the use of a Trusted Third Party is to mitigate the latency effect of remote attestations. As such, the TTP will be used to add a timestamp to the hashed value as demonstrated in *Improving the scalability of platform attestation.* Yet, the real purpose of using a TTP is to reduce the number high costing TPM operations by relinquishing "the server from integrating every nonce of each client into the costly TPM operations." [17] The timestamp can then be used for entity synchronization, which gives the entities greater control and visibility into the negotiated sessions.

**Local Trust Manager.** The Local Trust Manager manages all trust negotiations at the local level. It is comprised of the Reputation Manager, Policy Manager and the Trust Engine. Additionally it takes input from the TPM and TTP to determine if the requesting entity is a trusted user or a potential malicious user. The trusted session will be granted if the user is trusted, whereas the session will be halted if the user is determined to be malicious. A local trust negotiation process would be a user requesting a service hosted by a cloud provider.

**Global Trust Manager:** The Global Trust Manager, in a nutshell, manages all the Local Trust Managers and the relevancy of their trusted sessions. Take for example a server that can run hundreds of virtual machines which would overwhelm the Local Trust Manager. As such, the Global Trust Manager incorporates capabilities to aggregate the TPM attestations of sessions. But it also incorporates capabilities to monitor the local trust negotiations to identify changes in the state of the negotiation disseminating this information to trust engine.

## 5 Conclusions

In this paper we have presented a top-level trust architecture that addresses the inherent complexities of placing trust in cloud computing. Cloud computing users (organizations and individuals) demand the assurance that they have control and visibility over their data and interactions with the cloud provider. To do so required the inclusion of a Trusted Platform Module and Trusted Third Party. More importantly was the need for a Trust Engine that verifies the trustworthiness of the interacting entities. However with that said, there is much room for improvement of this architecture. For starters the logics behind the trust negotiation process needs to be expanded and formally verified. This would include the hashing of attestations and the addition of a timestamp to the hashed attestations. Additionally, the algorithms to be used to determine the trustworthiness of an entity would need to be established. Finally, a small scale implementation would establish if the presented model is a workable solution.

## 6 References

[1] Abawajy, J. (2009). Determining service trustworthiness in intercloud computing environments. *2009 10th International Symposium on Pervasive Systems, Algorithms,*

*and Networks.* doi:10.1109/I-SPAN.2009.155.

[2] Abawajy, J. (2011). Establishing trust in hybrid cloud computing environments. *2011 International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11.* doi: 10.1109/TrustCom.2011.18.

[3] Ahmed, M., Xiang, Y., & Ali, S. (2010). Above the trust and security in cloud computing: a notion towards innovation. *2010 IEEE/IFIP International conference on embedded and ubiquitous computing.* doi:10.1109/EUC.2010.114.

[4] Brown, A. & Chase, J. S. (2011). Trusted platform-as-a-service: A foundation for trustworthy cloud-hosted applications. *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop.* doi:10.1145/2046660.2046665.

[5] Cloud Security Alliance. (2010). Top threats to cloud computing v 1.0. Retrieved March 10, 2012, from https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf .

[6] C oviello, A. W., Elias, H. D., Gelsinger, P. & Mcaniff, R. (2011). Proof, not promises: Creating the trusted cloud. *EMC Corporation.* Retrieved March 10, 2012, from http://www.emc.com/collateral/emc-perspective/11319-tvision-wp-0211-ep.pdf.

[7] Craig, R., Frazier, J., Jacknis, N., Murphy, S., Purcell, C., Spencer, P., & Stanley, J. (2009). Cloud computing in the public sector: Public manager's guide to evaluating and adopting cloud computing. *Cisco Internet Business Solutions Group (IBSG)*. Retrieved March 12, 2012, from http://www.cisco.com/web/about/ac79/docs/sp/Cloud_Computing.pdf.

[8] Habib, M. H., Reis, S., & Muhlhauser, M. (2011). Towards a trust management system for cloud computing. *International joint conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11.* doi:10.1109/TrustCon.2011.129.

[9] Hazard, C. J. & Singh, M. P. (2010). An architectural approach to combining trust and reputation. Retrieved from http://www.csc.ncsu.edu/faculty/mpsingh/papers/mas/aamas-trust-10-architecture.pdf.

[10] Hwang, K. & Li, D. (2010). Trusted cloud computing with secure resources and data coloring. *IEEE Internet Computing*, *14*(5). doi:10.1109/MIC.2010.86.

[11] Kerr, R. & Cohen, R. (2008). Smart cheaters do prosper: Defeating trust and reputation systems. *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009).* Retrieved March 10, 2012, from www.cs.uwaterloo.ca/~rckerr/KerrCohen-aamas2009draft.pdf.

[12] Khan, K. M. & Malluhi, Q. (2010). Establishing trust in cloud computing. *IT Professional 12*(5). doi:10.1109/MITP.2010.128.

[13] Kramer, S., Gore, R., & Okamoto, E. (2010). Formal definitions and complexity results for trust relations and trust domains. Retrieved March 10, 2012, from http://www1.spms.ntu.edu.sg/~ccrg/documents/trust.pdf.

[14] Li, J., Li, B., Meng, L., & Sun, D. (2010). HiTrust: A hybrid tree based trust negotiation service. *2010 IEEE 24th International Conference on Advanced Information Networking and Application Workshops.* doi:10.1109/WAINA.2010.149.

[15] Ruan, A. & Martin, A. (2011). RepCloud: Achieving fine-grained cloud TCD attestation with reputation systems. *Proceedings of the sixth ACM Workshop on Scalable Trusted Computing.* doi:10.1145/2046582.2046586.

[16] Skogsrud, H. & Benatallah, B. (2003). Model-driven trust negotiation for web services. *IEEE Internet Computing,7*(6). doi:10.1109/MIC.2003.1250583.

[17] Stumpf, F., Fuchs, A., Katzenbeisser, S. & Eckert, C. (2008). Improving the scalability of platform attestation. *Proceedings of the 3rd ACM workshop on scalable trusted computing.* doi:10.1145/1456455.1456457.

[18] Takabi, H., Joshi, J. B. D., & Ahn, G. J. (2010). SecureCloud: Towards a comprehensive security framework for cloud computing environments. *2010 24th Annual IEEE Computer Software and Applications Conference workshops.* doi:10.1109/COMPSACW.2010.74.

[19] Zhou, Z. X., X. H., & Wang, S. P. (2011). A novel weighted trust model based on cloud. *Advances in Information Sciences and Service Sciences 3*(3). Retrieved March 10, 2012, from www.aicitglobal.org/aiss/ppl/Binder1-15.pdf.