MTC: Multi-Tiered Cloud Architecture

Wei Xie\textsuperscript{1,2}, Chongwu Dong\textsuperscript{1}, Wushao Wen\textsuperscript{1,2,*}, Zhe Xuanyuan\textsuperscript{1}, Yin Jia\textsuperscript{1}

\textsuperscript{1}School of Data and Computer Science, Sun Yat-sen University, Guangzhou, Guangdong, China
\textsuperscript{2}Joint Institute of Engineering at Carnegie Mellon University, Guangzhou, Guangdong, China

Abstract—A cloud has both finite bandwidth resources and service capability. Specifically speaking, one mega data center belonging to a cloud is only able to provide services for a certain number of connections that do not exceed a performance threshold. When saturated by bandwidth-consuming connections from a huge number of users, the cloud could not sustain such load, resulting in a low level of service quality, such as high latency and limited bandwidth. In order to solve this problem, we present a new cloud computing architecture named Multi-Tiered Cloud (MTC) architecture. MTC has high scalability and customizability. It supports hybrid pull-push model especially focusing on pushing multimedia resources with high concurrency. We abstract a concept, User Groups capable of Autonomous Programming (UGoAP), to help explain how our architecture is both inward and outward customizable. A case study is also shown to better illustrate the architecture.

Keywords: Cloud computing, Multi-tiered architecture, Inward and outward customizability

1. Introduction

Cloud computing is a specialized distributed computing paradigm where computing power and resources are provided as service to be accessed remotely via the Internet [7]. It has successfully enhanced the efficiency of resource management, supporting pay-as-you-go fashion, and lowers the up-front investment and operating cost of enterprises. Companies and developers no longer need to throw large capital outlays in hardware to deploy new services, thus also save vast expenses related to maintaining hardware [1]. By purchasing the services with elasticity, startups can even handle huge transient data flow. It seems like that cloud computing provides the tenants with an illusion of infinite computing resources available on demand [6].

However, in most cases cloud providers adopt a centralized architecture that has inherent problems such as unreliable latency and bandwidth bottleneck [8]. There are four categories of typical scenarios in cloud computing: 1) big data processing. (e.g., with MapReduce or Spark model) 2) video streaming. 3) cloud storage. 4) interactive services. (e.g., video chat, cloud gaming). In big data processing scenarios, a centralized architecture is appropriate, benefiting from discount over standard power rates [4]. However, in video streaming scenarios, which are bandwidth-consuming, a centralized architecture will cause the mega data center to be under a heavy bandwidth load. In cloud storage scenarios, the interplay between the server side and user side is usually pull-based (i.e., end users are always the initiator of a connection). Since end users lack the global view of the whole system, it is difficult to achieve the overall optimization.

In interactive services scenarios, latency is the essential part that influences the user experiences. Unfortunately, to pursue lower energy and construction costs, cloud service providers prefer to locate mega data centers in a desolate area far away from end users, leading to a higher latency [2].

In general, the current centralized cloud computing architecture is not efficient and scalable in bandwidth consumption and could not meet the stringent latency requirements of evolving applications. Moreover, as applications become more data-intensive and the need for interactiveness continue to grow [6][11], new cloud computing architectures are necessary to bring both bandwidth, storage and computing power closer to the users.

We are motivated by the fact that though content delivery network (CDN) has been the most widely used solution in the industry, it only provides the data and content delivery. CDN does not provide computing service delivery. Therefore, we are inspired to design an architecture that transforms the delivery model from simple ‘delivery of data’ to ‘delivery of services’. Besides, we leverage the feature of locality of CDN to achieve high bandwidth utilization, high end-to-end throughput and low service delay.

In this article, we present a multi-tiered cloud (MTC) architecture. A novel feature of MTC is to decouple the cloud into control plane, data plane, and service plane. Based on such decoupling, the bottom tier of our architecture can be built not only by cloud providers, but also a group of users working in the same company, living in the same residential area, or learning in the same campus. The goals of this architecture are highlighted as follows:

- High scalability: It is convenient to add new nodes to an existed MTC architecture.
- High bandwidth efficiency: MTC has an obviously better bandwidth utilization than a centralized architecture.
- High customizability and encouraging innovations: MTC is both inward and outward customizable and exposes open interface for various content providers.
(CPs) or ISPs to deploy their services.
- Hybrid Pull-Push Model: MTC provides an efficient mechanism and interfaces to support both pull and push services.
- Co-operative construct: Cloud providers and users can construct MTC in co-operative efforts. Users can build their own local cloud subordinated to the MTC model and interfaces. Then the local cloud can access into MTC core framework.

The remainder of the article is organized as follows. We first present the multi-tiered cloud architecture. Then we make a comparison between MTC and traditional solutions such as CDN, hybrid cloud. After that, we provide a case study of MTC followed by conclusion.

2. Related Work

There are several works in the literature that address the bottleneck of the current cloud models. For example, CDN focuses on the network and content placement optimization problems for delivering data contents based on locality and caching techniques. Hybrid cloud and cloud federation, which are created by interconnecting multiple independent clouds either among public clouds and private clouds, or among public clouds built by different vendors [9][12], also try to solve scalability and security issues of a centralized cloud.

2.1 Solutions based on locality

CDN is a distributed system that provides fast web content and streaming media delivery based on the geographic closeness to the users. As a distributed approach lacks a global view of the whole system, some research introduced a central control plane designed around centralized optimization to provide great benefit [10]. However, CDN cannot be customized by end users.

2.2 Hybrid cloud

A Hybrid cloud is a cloud composed of two or more cloud types, such as private cloud and public cloud. It hosts resources with a shared resource pool for all the members in the cloud and a private resource pool solely for the private members in the cloud [14]. Fog computing was proposed to address latency problem by providing elastic resources and services to end users at the edge of the network [13]. However, it is expensive to build many micro data centers at the edge or use high-end devices such as IOx [5].

2.3 Cloud federation

In this research track, there are many researches on Intercloud. The concept ‘Intercloud’ was proposed to support scaling of application across multiple vendor clouds [3]. If a cloud cannot meet the demand of users, other clouds can serve them immediately. Intercloud requires different vendors obeying to a common standard and protocol, and most importantly sometimes vendors do not provide the same cloud service functionality. Such fact causes a big obstacle if Intercloud will apply in the real IT industry.


The MTC architecture consists three tiers as depicted in Figure 1. We refer to the top tier as the Top Cloud, the second layer as the Trunk Cloud, the bottom layer as the Service Cloud. This architecture has one Top Cloud, several Trunk Clouds and many Service Clouds, each belonging to its corresponding Trunk Cloud. Service Clouds are located in end users vicinity, ensuring QoS to end users. A Trunk Cloud can be designed as a distributed data center, which provides basic cloud services like storage capacity and computation capacity. Different Trunk Clouds connect to each other over Internet. Data can be transferred from a Trunk Cloud to another. The Top Cloud serves as a control center which owns a global view of the whole architecture.

3.1 Decouple the control, data and service plane

In MTC architecture, we decouple the control plane, data plane and service plane as depicted in Figure 2. According to the geographical distance to end users and the difference of storage capacity, we divide the data plane into three tiers. According to the range of known information, we divide the control plane into three tiers to establish a top-down control model. By defining interfaces between each tier and each plane, we hide the implementation detail of them and expose
the abstraction of their functionality, making this architecture more agile and scalable.

The control plane is responsible for supervising and directing the whole system. In this top-down control setting, the Service Cloud responsible for managing end users (e.g., user access management) and Trunk Clouds are responsible for managing Service Clouds belonging to it. The Trunk Cloud hands over the control information (e.g., link state, bandwidth load) to the Top Cloud for a global view and optimization (e.g., centralized optimization and load balancing).

The data plane is mainly constituted by Trunk Clouds. A Trunk Cloud can be implemented by a distributed data center, which can provide storage and computation capacity. The Service Cloud has the mechanism to buffer data that is frequently being used by end users, relieving the bandwidth load between the Trunk Cloud and the Service Cloud. Data can be transferred between Trunk Clouds. The Top Cloud stores the data index to identify which Trunk Cloud contains the users’ desired data and controls the data transfer from a Trunk Cloud to another.

The service plane eventually provides services to end users. It takes advantage of the geographic closeness to end users, ensuring a good QoS to them. The Service Cloud can run various applications in separated application containers, thus supporting multi-tenants and enabling a PaaS service with very low latency. Compared with the Trunk Cloud, the Service Cloud has a limited storage capacity and computation capacity. So encountering a heavy data or a heavy computation tasks, the Service Cloud delivers these tasks and data to the Trunk Cloud. The Service Cloud continually synchronizes user data with the Trunk Cloud periodically and just maintain the frequently used data for users.

### 3.2 Design of MTC architecture

The design of MTC architecture is depicted as Figure 3. The Top Cloud consists several servers located in different areas, responsible for controlling the whole system. As the Top Cloud server and the Trunk Cloud are scalable, this architecture supports convenient appending of new server and new Trunk Cloud, which is controlled by the authentication module. The Top Cloud, Trunk Cloud and Service Clouds have the information module, and the information, containing performance states like storage load and available bandwidth, is shared between adjacent clouds. The Top Cloud leverage the control module to process data and task flows. The cloud service module is used to provide storage and computation services while the transfer module is used to transfer data from a Trunk Cloud to a Service Cloud or between Trunk Clouds. The container management module manages the application containers to run applications and provide PaaS services. User tasks are scheduled by the task management module.

The Top Cloud interact with the Trunk Cloud though the north API. The Top Cloud can direct the transfer of data between Trunk Clouds. For example, when data is stored in unbalance between Trunk Clouds, the Top Cloud can recognize this unbalance by checking its information module. To adjust this unbalance, the Top Cloud can give a command to transfer data from a high-load Trunk Cloud to a lower one when idle bandwidth is sufficient. In some cases, when a Trunk Cloud needs some data located in other Trunk Cloud, such data transfer will also happen. Once this transfer occurs, a Trunk Cloud transfers its data to another through the horizon API. The north API also support the Top Cloud in pushing data to end users directly.

The horizon API is the interface between different Trunk Clouds. Each Trunk Cloud has several storage nodes. A file with considerable size will split into a series of slices. These slices and their duplicated slices are arranged to be stored on different nodes. When receiving the command from the Top Cloud to transfer files from a Trunk Cloud to another, the transferring initiator invokes the horizon API to manipulate the nodes to transmit data to nodes of other Trunk Cloud. Such multi-nodes to multi-nodes data transfer can reach high throughput as depicted in Figure 4.

Service Clouds can provide an environment for applications to run in the application containers. Applications in these containers can use storage and computation capacity...
provided by the Trunk Cloud. Service Clouds invoke the south API to leverage these cloud services. Service Clouds intrinsically provide services to end users. Service Clouds have a standard interface for new services to be deployed, so that Trunk Clouds can deploy new services on Service Clouds easily and a group of people can construct or customize their own Service Cloud conveniently.

3.3 UGoAP based customization and hybrid pull-push model

We abstract a concept, user groups capable of autonomous programming (UGoAP), to help in explaining the customization feature of MTC architecture. UGoAP refers to a group of users that have the ability to program based on a standard API. For example:

- A user group in a campus which consists of students and professors.
- A user group in a company with IT department.
- A user group in a government organization which has outsourced their IT function.
- A user group in an intelligent residential area of which the property management company is capable of programming.

These groups have the ability to achieve the fine granularity customization. Once one architecture provides the standard API (e.g., REST or HTTP API), they can deploy all kinds of services for their specific demands, thus bringing the service clouds high customizability.

The Service Cloud provides both inward and outward customizability compared with a traditional CDN solution or a hybrid cloud. Regarding CDN, CDN is usually deployed by cloud providers or a third-party CDN company. So the end users do not have the privilege to operate the detail of the CDN server. As for hybrid cloud, Companies usually put their sensitive and private data in a private cloud and put their large size files and computing data on a public cloud. The private cloud can connect to the public cloud, but it does not expose API to outside. The outside usually do not know the presence of the private cloud of a company. The content providers have little chance of distributing their data and application on the private cloud. The Service Cloud can provide inward customizability. The services cloud can provide IaaS, PaaS, and SaaS to the inner users. Users can deploy their application on services cloud, which we referred as inward customizability. Moreover, the Service Cloud also exposes open API to outside, which supports different ISPs to deploy their services and different content providers to deliver their data.

In a pull-based model, the users usually serve as the initiator. Different initiator does not know the state, information of connection and status of links. In our architecture, the Top Cloud has the global view of the whole system, which can:

- For the large-scale foreseeable video streaming distribution (e.g., The World Cup), it can provide predistribution when bandwidth is idle.
- Achieve better load balancing based on the transferring content, status of link and the idle states of Service Clouds.
- Support more business form (e.g., auto upgrade of applications and information notification).

By introducing the hybrid pull-push model, our architecture can support the variety of demands of application and increase the bandwidth utilization.

3.4 Comparison with existing architecture and model

As shown in Table 1, under these four typical scenarios, we contrast three traditional approaches to our architecture. In the big data processing scenario, mega data center which adopts centralized power supply and cooling is fit, beneficial from discount over standard power rates. So it is cost-efficient, while other approaches do not have such advantages. In the video streaming scenario, a centralized architecture has low bandwidth utilization. The mega data center needs to establish different connections to different users. Many approaches use the locality designs to provide fast content distribution. The locality designs can be implemented by CDN and hybrid cloud. In the cloud storage scenario, traditional architectures and models are usually pull based and are activated by end users. Our architecture supports hybrid pull-push based model. In the interactive scenario, our architecture provides both inward and outward customizability. Content providers can deploy their data, information and even application on the services cloud which is close to the end users. That can help to achieve a low-latency QoS.

MTC can be beneficial to all the roles related to cloud computing, such as end users, content providers, cloud providers and ISPs. From the perspective of end users, they can experience a higher QoS (e.g., more fluently video streaming and lower latency). From the perspective of content providers, they can reduce some cost of building the CDN infrastructure. From the perspective of cloud providers, the expense also can be cut since MTC can be constructed
Table 1: Comparison with existing architecture and model.

<table>
<thead>
<tr>
<th>Sample applications or models</th>
<th>Big Data Processing</th>
<th>Video Streaming</th>
<th>Cloud Storage</th>
<th>Interactive services</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapReduce and Spark model</td>
<td>YouTube; Twitch</td>
<td>Dropbox; OneDrive</td>
<td>Cloud gaming; Video chat</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>The scale of computing cluster</td>
</tr>
<tr>
<td>High bandwidth ensurance</td>
</tr>
<tr>
<td>Content security and privacy ensurance</td>
</tr>
<tr>
<td>Latency and response time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mega DC</td>
</tr>
<tr>
<td>Cost efficient. Beneficial from discount over standard power rates</td>
</tr>
<tr>
<td>Bandwidth-consuming applications cause bottleneck</td>
</tr>
<tr>
<td>Pull based</td>
</tr>
<tr>
<td>Relatively high latency</td>
</tr>
</tbody>
</table>

| CDN |
| None |
| Use locality to provide fast streaming delivery |
| Pull based |
| Do not support customization from users |

| Hybrid Cloud |
| Depend on public cloud |
| Use locality to provide fast streaming delivery |
| Pull based |
| Do not support customization from ISP and content provider |

| MTC |
| Depend on the Trunk Cloud |
| Use locality to provide fast streaming delivery; Centralized optimization |
| Hybrid pull push based |
| Support high customization |

In co-operative efforts. From the perspective of ISPs, the bandwidth efficiency of MTC can release the congestion on the core network.

### 4. Case Study

In this section, we elaborate an example case in a video surveillance system. The case is that the government is constructing a video surveillance system as shown in Figure 5. The government instructs a hypermarket to connect into the system. During the process, technical operations are outsourced to a video processing company. We show how MTC is superior in this case. The reality requirements are: 1) the hypermarket does not allow all videos to be uploaded, since machine rooms and storage rooms involve privacy or confidentiality. Therefore, the hypermarket requires configuring their privacy policies as a filter. 2) The government only requires a subset of the videos. For example, the government just wants videos from 6:00 AM to 11:00 PM (i.e., duration policies as a filter). 3) The video processing company may want to pre-process (such as video compression and video content indexing) videos in the hypermarket locally before uploading, to reduce the bandwidth consuming. In MTC, the Service Cloud provides policy customization and the environment for applications running. For inward customizability, the hypermarket as the end user can provide privacy polices as a filter. For outward customizability, the government can define other filters and the company can deploy their proprietary applications. After passing through these filters and applications, the videos become smaller in size and volume.

In the above paragraph, we observe the case from the perspective of the Service Cloud. Now we observe it from the perspective of a global view as shown in Figure 6. The company have deployed a Deployment Management System (DMS) in the Trunk Cloud to achieve the automatic deployment from the Trunk Cloud to the Service Cloud. The DMS makes the deployment convenient from a Trunk Cloud to multiple Service Clouds by a top-down control, a key feature of MTC. When videos transfer back to the Trunk Cloud after filtering and processing, the company transfers videos to the government’s storage system in other Trunk Cloud by the horizon API. The government uses their storage...
Fig. 5: Video surveillance system in the Service Cloud. The input is 20-TB surveillance videos. After the process of the Service Cloud, the output is reduced to 1-TB videos that need to be transferred. We have saved the broadband resources.

Fig. 6: Video surveillance system overview in MTC. Every Service Cloud can set their own filters and these filters can be auto deployed by a management system running on the Trunk Cloud. All the video data can flow across Trunk Clouds. To back up these videos, which can be retrieved for taking evidence. After all above procedure, the government gathers the statistical information and indexes and transfer them to the Top Cloud for universal control and scheduling.

Traditional CDN solutions and hybrid cloud cannot achieve this since the case requires the local components to have both inward and outward customizability. First, the case requires videos to be processed close to end users (i.e., the locality feature). CDN can support the feature but it is transparent to the users. The users cannot customize CDN (CDN has no inward customizability). Second, hybrid cloud is not suitable since the case requires the policy configuration from the government and the application deployment from the company (hybrid cloud has no outward customizability).

MTC shows three superiorities in this case that makes it outperform CDN solutions and hybrid cloud. 1) From the perspective of resource utilization, MTC makes a trade-off between the computation capacity and the bandwidth consuming. Since the videos pass through the filter and the pre-processing, we leverage the computation capacity of the Service Cloud to reduce the bandwidth consuming between the Service Cloud and the Trunk Cloud. 2) From the perspective of service demand, MTC can flexibly fulfill various requirements such as privacy requirements and duration requirements. 3) From the perspective of the whole architecture, the Top Cloud has a global view and it has a high control over the whole system. When the number of the Service Clouds increases or massive data flows occur, the Top Cloud can achieve some centralized optimizations such as link optimization and load balancing in MTC.

5. Conclusions

In this article, we have proposed a novel cloud computing architecture. The key part of this architecture to decouple the control plane, data plane and service plane. In addition, MTC architecture supports hybrid pull-push services and focusing on pushing multimedia resources with high concurrency. It is a high scalable architecture and it is easy to be customized by users. MTC supports both inward and outward customizability. We believe a highly customizable multi-tiered architecture will be prevalent in the near future as applications continue to become more data-intensive and require more stringent latency.

References