Abstract – This research was done to evidently prove that the response time for a cloud-hosted website is faster when the data is cached to a proxy server physically located closer than the cloud to clients. A K-means-based database partitioning algorithm was developed to group data with similar access patterns together for caching. A bookstore web application was implemented in the Amazon Cloud to perform experiment. The results of the experiment show that data caching can significantly reduce response time for a cloud-hosted website.

Keywords: data-caching; cloud-computing; K-mean data-partitioning; cloud-hosted website.

1. Introduction

In simple words, Cloud Computing is the computing based on the internet. In a typical environment, programmers build some dynamic applications and deploy it in a physical computer on a server located near the business. But, with the advancement on Cloud Computing, working with dynamic applications has taken a new dimension. Cloud Computing allows people to access the same dynamic web application through the means of the Internet [16]. According to the Amazon Web Services (AWS), “Cloud Computing, by definition, refers to the on-demand delivery of IT resources and applications via the Internet with pay-as-you-go pricing.” Whether we are running applications that share photos to millions of mobile users or we are supporting the critical operations of our business, the “cloud” provides rapid access to flexible and low cost IT resources. By using the Cloud Computing technology, we have a simple way to access servers, storage, databases and a broad set of application services over the Internet. In today’s world, we see that more and more websites are moving towards the cloud. There are many advantages of using Cloud platform over website hosting in a server. Instead of having to invest heavily in data centers and servers before we know how we are going to use them, we can only pay when we consume computing resources, and only pay for how much we consume. Another benefit is that by using the Cloud Computing technology, we can achieve a lower variable cost than we can get on our own. Because usage from hundreds of thousands of customers are aggregated in the cloud, providers such as Amazon Web Services can achieve higher economies of scale which translates into lower pay as you go prices. In a cloud computing environment, new IT resources are only ever a click away, which means we reduce the time it takes to make those resources available to our developers from weeks to just minutes. This results in a dramatic increase in agility for the organization, since the cost and time it takes to experiment and develop is significantly lower [16].

Despite all these benefits and its flexibility, there is a major drawback for many businesses in using the cloud platform. Cloud is deployed in data-centers. It requires a lot of investment for businesses to get data centers up and running. There are a very few data centers in many locations around the world. This is because of the initial and the maintaining cost of data centers for many businesses. So, for businesses that are physically located far from the data center, the response time for the dynamic web application is usually more than those located near to the data centers. To improve the response time for the cloud hosted websites, there is a need of some cloud caching mechanism.

Web caching is a technology aimed at reducing the transmission of redundant network traffic. A lot of other related works have been done in this field of research. A user would opt for web caching to increase the bandwidth availability by curbing the transmission of redundant data [1]. Solutions including optimization of the Apache web server, introduction to caching technologies and widespread implementation of AJAX code are purposed as optimization techniques to improve performance [2]. In proxy based caching, two broad approaches exist in using proxies to cache dynamic pages, namely page-level caching and dynamic page assembly [4]. In these approaches, the dynamic contents are cached outside the site’s infrastructure. In page-level caching, the proxy caches full page outputs of dynamic sites. Whereas some solutions are deployed in forward proxy mode, in distributed caching architecture located at numerous points around the Internet [5, 6]. Page-level caching can improve website performance by reducing delays associated with page generation, as well as reducing bandwidth consumption. The server side caching solutions are based on the idea of caching dynamic content within the site architecture at various levels, to accelerate dynamically generated content [3]. Various types of database caching have been suggested, including caching the results of database queries [7, 8], caching Web Views [9], caching database tables [10] and caching database tables in main memory [11].

All these techniques used work very well for web caching technologies. However, as we mentioned that more and more websites are moving towards the cloud platform, one key issue remains in hand. There is a need for caching of the cloud-hosted website.
2. System Model

In the model shown in Figure 1, the AWS Cloud Platform consists of the bookstore web application and the database. The bookstore application is a Java-based web application that serves as a web service for the clients to interact with the database. The web application is wrapped by a proxy service which acts as a standard used for exchanging data between the application and the client. In our application, the client sends a standard HTTP protocol involving the HTTP request methods like GET, POST, PUT, DELETE, and PATCH etc. to access, retrieve, modify and delete information from the database. The web service in the cloud accepts the client request, processes it with the web service, handles the request, and responds it back to the client with the HTTP response object. The web service has specific methods to handle the client requests. For example, there exists a service method named getAllBooks() which serves as a service layer between the client and the database. When the getAllBooks() service call is made by the client, the web service will then communicate with the database and retrieve all the books. The information thus received from the database will then be sent to the client in a JSON object format. The database hosted in the AWS cloud platform is a relational database using the MYSQL database engine.

The database consists of a table named Book. The table consists of the id of the book, the title, the name of the author, the language of the book, the online-based user rating, the date that the book was published and the subject or the field of study of the book. The table consists of 5000 unique books. So, altogether the cloud platform consists of the web service along with the database with all the books available to the customer. When client requests for resources on the cloud, the response time is recorded. Now, on the other hand, a proxy web server is created which is located physically closer to the web client. This proxy server consists of the web application which is again a wrapped web service that lets a user request for resources. The database however consists of data retrieved after running the K-Means Data Clustering Algorithm on the main dataset. Since the proxy web server is just a proxy server and not the main server, only a fraction of the main dataset is stored in it. In order to retrieve only a fraction of the main dataset, we choose the retrieve the dataset that are mostly accessed by a client of that location. For example, there is a client who uses the bookstore application living in Shanghai, China, and the web application is hosted in Washington, United States. For this client, the fraction of the main dataset is stored in the proxy server located closer to the client. In this example, this fraction of data will most likely consist of many Chinese books. These books will then be stored in the proxy server. Now, when a client who is located in China makes a request to the web application, the client is served by the proxy server first. When the proxy server can handle the request of the client, the response time is recorded. When the proxy server is not able to handle the request, the original request of the client is redirected to the main server. The response time is recorded in this case as well. Based on this model, a various number of experimentations are performed to eventually prove that data caching can significantly reduce response time for a cloud hosted website.

3. K-Mean Data Partitioning Algorithm

K-means is a clustering method that aims to find the positions of the clusters that minimize the square of the distance from the data points to the cluster. The problem statement of this algorithm is that we are given a set of data points and we need to group these data points into a cluster so that points within each cluster are similar to each other and points from different clusters are dissimilar. Usually, the data points are in a high-dimensional space, and similarity is defined using a distance measure like Euclidean, Cosine, Jaccard, Edit Distance, etc. K-Mean data partitioning algorithm is commonly known as the K-means clustering algorithm. It is also referred to as Lloyd’s Algorithm. It is the most common algorithm which is as an iterative refinement technique. Clustering, in general, is the process of partitioning a group of data points into a small number of clusters. The goal is to assign a cluster to each data point. The K-mean Algorithm works as follows: First, initialize the center of the clusters. Then, attribute the closest cluster to each data point. Next, set
the position of each cluster to the mean of all data points belonging to that cluster and finally repeat steps until there is a full convergence.

The algorithm eventually converges to a point, although it is not necessarily the minimum of the sum of squares. That is because the problem is non-convex and the algorithm is based on a heuristic, converging it to a local minimum. The number of clusters should match the data. An incorrect choice of the number of clusters will invalidate the whole process.

Thus, for the K-means clustering method, if k is given, the algorithm can be executed in the following steps:

- Partition of objects into k non-empty subsets
- Identifying the cluster centroids (mean point) of the current partition
- Assigning each point to a specific cluster
- Compute the distances from each point and allot points to the cluster where the distance from the centroid is minimum
- After re-allocating the points, find the centroid of the new cluster formed

Mathematically, minimizing the within-cluster sum of squares is defined as

\[ \sum_{i=1}^{k} \sum_{j=1}^{n} ||x_{ij} - c_i||^2 \]

where, n number of objects or data items have been partitioned into k non-empty subsets S_i, i = 1, 2, ..., k and the term \( ||x_{ij} - c_i||^2 \) provides the distance between a data point and the centroid of the cluster.

K-means clustering has a lot of applications in day-to-day life. So examples include clustering customers based on their purchase histories, clustering products based on the sets of customers who purchased them, clustering documents based on similar words or shingles, clustering DNA sequence based on edit distance.

In case of our bookstore web application, The K-Mean Data Partitioning Algorithm was used to cluster the books into k = 3 groups or subsets. The clusters where chosen based on the languages of the books accessed by the clients. This information was retrieved from the apache access log. The access matrix was created of dimension n*m where ‘n’ were the number of data rows and ‘m’ were the number of data columns. To run the K means Algorithm, the Lloyd method was chosen which categorized a data point into a cluster based on the Euclidean distance from the point to the centroid of the cluster. There were a total of 5000 data points and each data point consisted of 10000 dimensions. So, instead of the two dimensional Euclidean distance, the ‘n’th Euclidean Distance is calculated based on the ‘n’th Euclidean distance formula. Mathematically, if ‘p’ and ‘q’ were two data points, then, in n-dimensional space, the distance is calculated as

\[ d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2 + \ldots + (p_n - q_n)^2}. \]

After all the necessary data was collected, a Cluster Analysis programming language named ‘R’ was used. R Language is a programming language and environment for statistical computing and graphics. It is an integrated suite of software facilities for data manipulation, calculation and graphical display. Before we perform the clustering, we need the numeric matrix of data. This is the usage matrix of a tuple of database rows with the queries extracted from the apache log file.

Usage matrix is nothing more than just a multidimensional matrix. For example, in a two dimensional numerical matrix, if there are ‘n’ number of data rows and ‘m’ number of data columns, then the usage matrix can be represented by a numerical matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns). For example, in a usage matrix of one thousand rows by one thousand columns, the item at row five and the column ten represents the data value for that specific row and column.

In case of our bookstore application, let us take a subset of the usage matrix of the tuple of database rows with the queries extracted from the apache log file.

```
<table>
<thead>
<tr>
<th>Tuples/Queries</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>...</th>
<th>Qn</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Tn</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Figure 2: Tuples-Queries Matrix

In figure 2, the tuples (T_1, T_2, ..., T_n) represent the rows in the database whereas the queries (Q_1, Q_2, ..., Q_n) represent the queries extracted from the apache access log file. Each data in the matrix represents if that query (column value) accesses the given tuple (database row). Inside the matrix, 1 represents the access hit while 0 represents the access miss of the database row using the query.

Now, to perform the k-means clustering on a data matrix, the following command was used in the R language.

\[ \text{kmeans(x,centers, item.max = 100000, nstart = 1, algorithm = c(“Lloyd”))} \]

where,

- ‘x’ is the numeric matrix of data. This is the usage matrix of a tuple of database rows with the queries from the apache log file. ‘centers’ is the number of clusters (say k).
- ‘iter.max’ is the maximum number of iterations allowed. This was chosen to be 100000 so that algorithm does not run for a very long period of time. Algorithm is chosen to be ‘Lloyd Algorithm’ to calculate the cluster based on the Euclidean
distance from the data point to the chosen centroids. This Algorithm is the most common algorithm used for an iterative refinement technique. This algorithm is often presented as assigning objects to the nearest cluster by distance. This distance is the Euclidean distance between the cluster and the data point. When running the K-Means Algorithm, the centroids of the clusters are chosen at random selecting K points from the dataset. In our case, we choose 3 centroids at random from the dataset to serve as centroids. But, the centroids are adjusted in the following iterations in the algorithm.

Figure 3 represents a simple diagram of how the clustering looks after the K-Mean Data Partitioning Algorithm is run. As we can see, the data are clustered into 3 distinct and different clusters named 1, 2 and 3.

![Figure 3. K-Means Data Clustering and the Cluster Membership](image)

After the K-Mean Data Clustering Algorithm is run on the usage matrix, we retrieve the cluster membership. In the figure 3, we can see that there are 3 distinct clusters. Each data point defines its own cluster membership. For example, the first data-point belongs to Cluster 1, the second data-point belongs to Cluster 2, the third data-point belongs to Cluster 3, the forth data-point belongs to the Cluster 1, the fifth data-point belongs to cluster 3 and so on. This finally let all the data-points belong to one of the three clusters.

4. Experimental Study

Using the model described in the System Model, experimental study was done to prove that data caching can significantly reduce response time for a cloud-hosted website. In order to effectively manage a web server, it is necessary to get feedback about the activity and performance of the server as well as any problems that may be occurring. The Apache HTTP Server provides very comprehensive and flexible logging capabilities. As the client makes HTTP calls to the web service in the cloud, there are call traces created. The call traces are nothing but the apache access logs. Apache provides the access log which is the server access log records of all the requests processed by the server. Then, the entire access log was analyzed in detail. A subset of the data access log was taken into the experimentation. The subset consisted of ten thousand lines of access log data. The format for the access log is exactly the same as the Common Log Format, with the addition of two more fields. Each of the additional fields uses the percent-directive %{header}i, where header can be any HTTP request header. The access log under this format will look like:

```
```

This server IP address, the GET request were then taken into account for the experimentation process. This was used to create the usage matrix. The usage matrix was nothing but an information matrix of the access log to the database row item as described in Section 2. The access hit or miss is determined from the access log.

Figure 2 represents a subset of the main usage matrix. In figure 2, the rows represent the database rows. As there are 5000 rows in the database hosted on the Amazon RDS instance, there are 5000 number of rows represented by T1, T2, T3,..., T5000. The columns represent the access hit or miss of each apache access log information. For instance, let us take the row number 10 from the database. As we analyze the tenth row with the column number 2, we see a value of ‘1’. We can interpret that as row number 10 is one of the data items in the database that is accessed/hit by the query in column number 2. Similarly, if we take row number 1 and column number 1, we see a value of ‘0’. We can interpret that as first database row is not accessed (or missed) by the query in column number 2 and so on.

This usage matrix is then fed into the K-Mean Data Clustering Algorithm. The K-Mean Algorithm then computes the distances from each point and allot points to the cluster where the distance from the centroid is minimum. For convenience, we set the number of clusters (k) to 3. After the K-Mean Algorithm is run, we have data nicely grouped into 3 clusters. We then cache one of the clusters to a proxy server physically located closer than the web server to clients. Then, we conduct the final set of experimentation. We created a pool of 25, 50, 75, 100, 125, 150, 175 and 200 concurrent users to test this scenario. On one hand, we record the response time for the clients who access the cloud application and on the other hand, we record the response time for the clients who access the application from the cached server.
The bookstore application used in the experimental study was created using the Spring Framework. The bookstore application was nothing but a web service with an HTML view engine for the client. When a client makes a request to the web application, an underlying service in the application is called to do a specific task. This service is nothing but a web service call. When the web application gets the request from the client, a specific web service method is called and the response is returned back to the client. This response is returned in the JSON format. Since this is not an ideal or a pretty form of viewing the resources returned from the server back to the user, the JSON objects are parsed and presented in a nicely formatted table or output to the user [12]. HTML view engine helps present the response of the web service is a user-friendly view back to the client. The Spring Framework used to create the bookstore application uses the Spring Boot technique. Spring Boot makes it easy to create stand-alone, production-grade Spring-based Applications that you can just “run”. Most Spring Boot applications need very little Spring configuration. Spring Boot applications feature of stand-alone Spring applications; embedded tomcat (no need to deploy WAR files); provides opinionated 'starter' Gradle file to simplify the Gradle projects; provides production-ready features such as health checks and externalized configuration and automatically configures Spring whenever it is possible. This dynamic web application as well as the database is hosted in the Amazon Cloud platform. The web application is hosted in the AWS Elastic Beanstalk platform. AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Go, Ruby, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS [14]. The bookstore web application and services was developed in Java with the Apache server. The web server environment was created as a single instance, auto-scaling Apache Tomcat 8 environment. The WAR file generated from the build of the web application was used as the source for the web server. The processing power of the web server in the environment used was of instance type t1.micro. With all these configurations, the environment was launched.

Similarly, the Database was hosted in the Amazon Relational Database Service (Amazon RDS). Amazon RDS is an Amazon Cloud Platform application which makes it easy to set up, operate, and scale a relational database in the Cloud. MySQL was used as the database engine. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing us up to focus on our applications and business needs. The database engine hosted is easy to administer and is highly scalable. It is also highly available and durable, fast, secure and inexpensive [13]. The database engine used was My-SQL. The instance class was used as db.t2.micro with the storage capacity of 5 GB. The default endpoint is created which connects to the dynamic web application with the means of My-SQL driver.

The java based web application, along with one of the clusters obtained from the K-Means Data Partitioning Algorithm is cached to a proxy server physically located closer than the web server to clients. Only one of the clusters is picked and cached to the proxy server. The web service also needs to be cached meaning that the web service is wrapped in the proxy server so serve the same functionality as the main server. The web service hosted in the cloud is modified in the proxy server so that when a request comes in from the client, the web service always looks for the data in the proxy server first. So, for instance, a client sends a request to the bookstore web application. This request is first sent to the proxy web server that is physically located closer to the client. Since this data is a cached version of the main web service, the client’s request may or may not be handled by the proxy server. If the proxy server is able to handle the request, then the response is returned back to the client. But, if the proxy server cannot handle the request, then the original request from the client is sent to the main server so that the request is handled appropriately. Figure 4 represents the graph of plotting the response time for 25, 50, 75, 100, 125, 150, 175, and 200 concurrent users for both the main cloud server and the cached server. For the cached server, the response time for clients are significantly less than that of a server hosted in the cloud. The response time for the cloud server is 191 milliseconds for 25 concurrent users, 195 milliseconds for 50 concurrent users, 188 milliseconds for 75 concurrent users, 226 milliseconds for 100 concurrent users, 239 milliseconds for 125 concurrent users, 255 milliseconds for 150 concurrent users, 340 milliseconds for 175 concurrent users and 318 milliseconds for 200 concurrent users. Similarly, for the cached server which is located physically closer to the client, the response time is 14 milliseconds for concurrent 25 users, 12 milliseconds for 50 concurrent users, 11 milliseconds for 75 concurrent users, 14 milliseconds for 100 concurrent users, 14 milliseconds for concurrent 125 users, 18 milliseconds for 150 concurrent users, 19 milliseconds for 175 concurrent users and 26 milliseconds for 200 concurrent users.
After the experimentation, we can see that the response time for the cached server is significantly lower than the main server. Thus, the experiment's results show that data caching can significantly reduce response time for a cloud-hosted website.

5. Conclusion and Future Work

Thus, with the means of various applications and different software platforms, we were successfully able to carry out the research of the data caching for a cloud hosted website. After the experimentation was carried out, we were evidently able to conclude that the data caching for a cloud-hosted website can significantly reduce the response time for the clients.

In this research, all the HTTP requests by the clients are GET requests. In other words, both the cached server and the main server only handle the reads from the database. As part of an on-going research, the future work remains. The future work will consist of experimentation of the research along with the write operations. After a client buys a book, the number of books in the bookstore application should reduce from the total number of available books to be purchased. While this seems like a feasible task, we need to be careful on how we handle the synchronization issues. If a client buys a book from the main server, then all of the cached servers with the total number of books have to be updated. So, to make it practical, all the cached servers and the main cloud server have to be in sync. Determining and enforcing per-application resource quotas in the resulting cache hierarchy, on the fly, poses a complex resource allocation problem spanning the database server and the storage server tiers. Modern enterprise systems consist of multiple software layers including web/application server front-ends, database servers running on top of the operating system, and storage servers at the lowest level. Thus, there is a high need for better management of shared multi-tier caches. Thus, much of the work on this part is needed.

References:


[4] Suresha, Dr. of Philosophy in the Faculty of Engineering.


