International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 04 | Apr-2018 www.irjet.net p-ISSN: 2395-0072

ACCELERATING THE MOBILE CLOUD USING AMAZON MOBILE ANALYTICS AND K-MEANS CLUSTERING

Anmol Gupta[1], Nikhil Khanna[2], Yash Gurnani[3], Sufal Addya[4]

[1,2,3,4] *Student, Department of Computer Engineering, SIT Lonavala, Maharashtra, India*

Abstract - *Current patterns demonstrate a move far from work area registering and toward the ascent is popular on cell phones. However cell phones experience the ill effects of impediments in memory, stockpiling, computational power, and battery life. A significant number of these confinements can be tackled by offloading calculations and capacity to cloud-based stages. This could cause lost business or moderate client development in more removed areas. Utilizing a few Amazon Web Services (AWS), k-mean examination based information apportioning answer for this issue. The exchange of k-means database parceling portrays a preprocessing procedure for adjusting crude AWS Mobile Analytics log information for use in the k-implies calculation.*

Key Words: **Amazon Web Service Mobile Analytics, Mobile Cloud Computing Proxy Server Caching, K-Means Database Partitioning.**

1. INTRODUCTION

Into districts and the application engineer must pick the region(s) from which to buy registering assets. AWS, for example, characterizes districts, for example, US East (N. Virginia), US West (Oregon), EU (Ireland), Asia Pacific (Tokyo), and Any case, for a worldwide client base, benefit quality can shift generally. For example, expecting an application sent to the US East area of AWS, clients in the United States will encounter much preferred reaction times over clients in different parts of the world, just in light of the fact that solicitations and reactions must travel considerably shorter

Separations, when they begin in the U.S. than if they begin in different nations. Poor administration quality could prompt absence of or moderate development in removed locales.

One potential answer for this issue is to send the application to a more concentrated district. While this arrangement may diminish inertness for the most far off areas, it will expand the dormancy for different locales, and there will even now be a huge divergence accordingly times between clients close to the more brought together district and those in more far off areas.

Distributed computing stages don't dispose of the designer's requirement for registering equipment, yet rather they comprehend that need by offering the utilization of that equipment as an administration. In this way, distributed computing specialist organizations arrangement their own particular servers and server farms, which must be in some area, or set of locations. Often, specialist organizations will gather these assets. Be that as it may, in the two cases, the exploration concentrated on web applications got to through a work area web program, and the server side of the framework depended on a conventional SQL database. This paper expands on this examination by adjusting the virtual intermediary arrangement proposed in past research to incorporate the utilization of a portable application on the customer side and a cloud-based, NoSQL database on the server side of the framework.

2. SYSTEM MODEL

In this exploration, we examine and test an edge processing framework, in which information will be conveyed to a NoSQL database at a focal area on a distributed computing stage, and the application rationale and a subset of the information will be imitated to intermediary servers physically nearer to far off clients. Customer asks for originating from a portable application, will initially be steered to the intermediary server, and if the demand is a hit, the information will be come back to the customer. In the event that it is a miss, the demand will be sent to the focal NoSQL database, which will restore the information.

Cloud assets were obtained through AWS, since Amazon is the main cloud stage supplier, and the intermediary server was conveyed to a Ubuntu virtual machine on a VMware server. The customer versatile application was produced for Android since it is the main portable OS as far as piece of the overall industry. See Figure 1 for a full framework outline. Dashed connectors speak to an average, non-intermediary, customer server engineering, while the strong connectors speak to the intermediary arrangement. Notice that, notwithstanding the intermediary server association, the versatile customers are additionally associated straightforwardly to the Amazon Mobile Analytics benefit. This AWS benefit is intended to gauge portable application utilization and income and track enter inclines in maintenance and new client procurement. Additionally, it enables the designer to track custom in-application practices to help settle on information driven choices identified with assist improvement. Notwithstanding furnishing a support with diagrams and charts of this information, Amazon Mobile Analytics enables the engineer to store application occasion information (i.e. logs) to Amazon S3 or Redshift, both long haul stockpiling administrations gave by AWS. In this arrangement, the portable customers log client action utilizing the Amazon Mobile Analytics benefit, and the logs are then put away in Amazon S3.

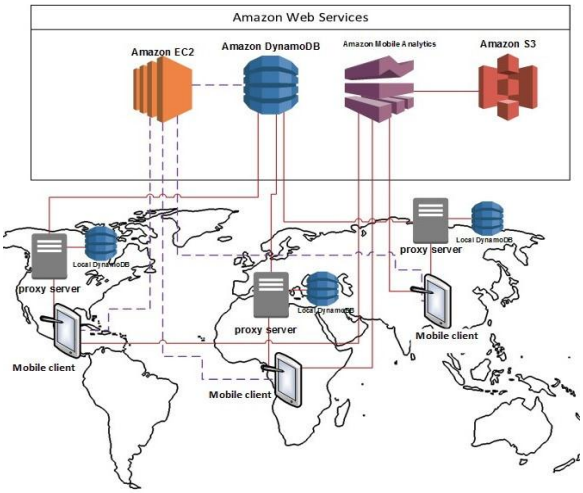

Figure 1. System diagram

There are a few inquiries to address before actualizing this arrangement. Two of them will be tended to here, and the others will be left to additionally investigate. The primary inquiry concerns information replication to the intermediary servers. One could reproduce the whole database to the intermediary areas, yet that would be a misuse of assets, since expansive parts of the database likely have a low likelihood of being asked for from certain geological areas. For example, consider a versatile internet business application for an online book shop. The segment of the database speaking to Chinese dialect books has a low probability of being asked for by clients in the United States. Moreover, duplicating every one of the information deters the requirement for a focal database occurrence. In spite of the fact that it isn't considered in this paper, repeating every one of the information likewise can possibly entangle, significantly further, information consistency issues related with compose solicitations to the database. Answers for this issue will be left to future research.

2.1 Data Partitioning Methodology

To begin addressing the first question, a data partitioning methodology will be described. The methodology relies on gathering mobile client log data, pre-processing it, and analyzing it for patterns using the k-means clustering algorithm. Variations on the algorithm are discussed and compared.

2.2 Log Generation

A basic mobile ecommerce app was developed for a fictional bookstore with a global customer base. A user of the app could browse a list of books, search for books, view detail pages, add selected books to the cart, and checkout. The mobile app logged user activity using the Amazon Mobile Analytics service, which then stored those logs using Amazon S3. Log entries tracked start and end timestamps for user sessions, user location, and the genre, language, and price of books the user browsed. An example log entry can be seen below. Field names have been bolded, and irrelevant fields have been omitted, for easier reading.

{ "event type": "Browse Event",

...

"attributes": {

"Longitude": "89.59",

"Latitude": "22.71",

"Language": "Hindi",

"EndTimeStamp": "2016-07-18 00:00:53.922",

"UserRegion": "India",

"StartTimeStamp": "2016-07-18 00:00:00.922",

"Genre": "Graphic Novel", "SessionID": "304a4f65-6c96-4453-a8cfc85f22e6787e" },

"metrics": { } }

The initial 2/3 of the fields in the log section (discarded) are consequently included as a major aspect of the Amazon Mobile Analytics Service, however the sorts of "event_type" are client characterized. The "traits" and "measurements" areas incorporate client characterized fields which were indicated automatically. "Longitude", "Scope", and "UserRegion" record the geographic area of the client. "Dialect", "Sort", and "Value" record qualities of the book which the client was seeing. "StartTimeStamp" records the minute when the client asked for the book. "EndTimeStamp" records when the client explored far from the book or shut the application. "SessionID" is an interesting id for every client session. Log passages speaking to one of a kind peruse occasions for one client's shopping session have the same "SessionID".To analyze usage data for the possible existence of clustering, a large log file was required. Since obtaining real user traffic data was impossible, it was simulated.

2.3 K-Means Mathematical Model

After creating the data matrix, the log data matrix was imported and analyzed using the R programming language, which is specifically designed for statistical analysis applications. The log data matrix was stored in a variable called “matrix”. This secondary pattern was added to the log data to “challenge” the k-means algorithm variations to see if they would misidentify the appropriate clusters.

The ideal result would be to find 5 clusters, with each cluster only containing tuples of a single language, represented as such, by the bar plots in Figure 2

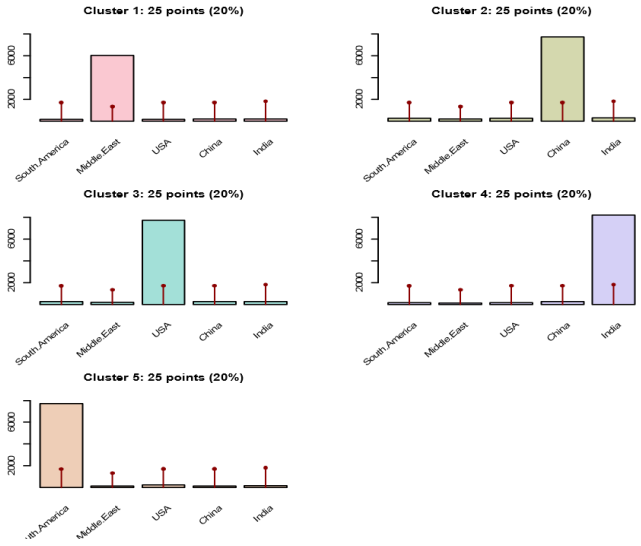

Figure 2. An ideal clustering

2.3.1 K-Means Algorithm Results

The "kmeans" and "angle" families produced similar results. With the Forgy method for initial centroid selection, 3 out of 10 runs produced the ideal clustering represented by Figure 2. For the other 7 Forgy runs of each, as well as all 10 runs of the "kmedians" family, partially ideal clusterings were produced. See Figure 3 for an example of a typical non-ideal clustering result.

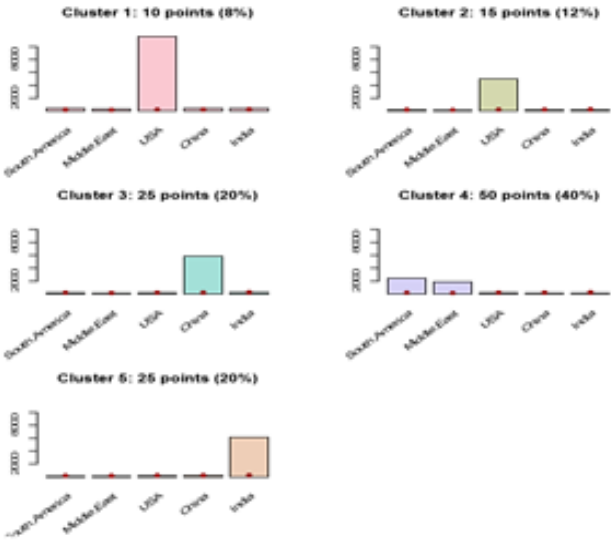

Figure 3. A non- ideal clustering

Both the Forgy method and hand-picked centroids runs of the "ejaccard" family produced results where the languages were not separated into distinct clusters. Instead the languages were roughly evenly distributed between clusters, but the results were slightly better than the "jaccard" family results

2.3.2 Conclusions from K-Means Analysis

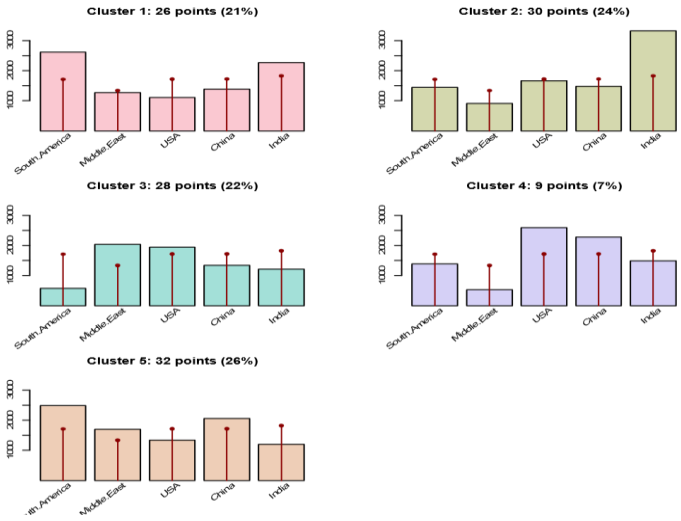

Figure 4. K-means clustering example

Finally, all tests of the Kmeans, Kmedians, and Angle variations, where the initial centroids were hand-picked, based on an intuitive sense of the data, correctly identified the 5 language clusters. This points to the important effect on the results of these algorithms of the choice of the initial centroids. Further research into the best alternative strategies for choosing initial centroids would likely be fruitful. For this project, the correctly identified language clusters were used as the guide for partitioning data to the proxy servers.

3. PERFORMANCE TESTING

Finally, a Java command line app was developed to test system performance from a MacBook Air laptop, connected to the Internet via Wi-Fi at a home, located in Newport, KY, USA. So, the client app was located roughly 6 miles from the proxy server, which was located roughly 6,550 miles from the central database. The command line app ran two tests: one to measure response time, and one to measure throughput.

3.1 Results of Performance Tests

The results of the performance tests can be seen below. All results were rounded to the nearest hundredth.

Test 1 - Response Time (Sample Size = 1000)

Average Response Time of Proxy System (Hit rate = 90.0%): 21.13ms

Average Response Time of Non-Proxy System: 185.22ms

Average Improvement: 164.09ms | 88.59% speed up

Test 2 - Throughput (Sample Size = 1000)

Throughput of Proxy System (Hit rate = 90.0%): 43.86 reads/second

Throughput of Non-Proxy System: 5.4 reads/second

Throughput improved by 712.19%

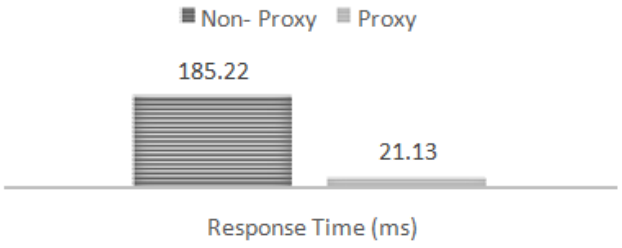

Figure 5. Response Time Improvement

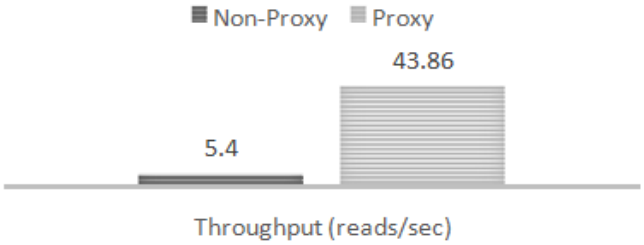

Figure 6. Throughput improvement

3.2 Analysis of Test Results

The intermediary framework offers a critical change over the non-intermediary framework as far as both reaction time and throughput. The intermediary framework accomplished a 88.59% change accordingly time, which would likely be recognizable to the client and offer a superior ordeal. The intermediary framework likewise accomplished a 712.19% change in general throughput, which implies a use of the intermediary framework would be better ready to scale thinking of it as could serve numerous more customer demands than the non-intermediary framework could serve in a similar measure of time. Moreover, despite the fact that the tests were performed with partitioned sets of HTTP asks for, they stick with each other, loaning legitimacy to the outcomes. Separating 1000ms by the normal intermediary reaction time (21.13ms) gives you 47.32 peruses/second, which is near the autonomously found intermediary throughput of Test 2

4. CONCLUSION

This domain is clearly a possible solution to handle a large set of data on the cloud. The available security measures are good to handle the data on the cloud in a secured manner. The proposed framework as a whole will be able to provide a potential result and will escalate the analytics process.

ACKNOWLEDMENT

It gives us great pleasure in presenting the project on 'Accelerating the Mobile Cloud:Using Amazon Mobile Analytics and K-Mean Clustering'. We would like to take this opportunity to thank my internal guide Prof. M.S.Chaudhary for giving us all the help and guidance we needed. We are really grateful to them for their kind support. Their valuable suggestions were very helpful. We are also grateful to Dr. S.D.Babar, Head of Computer Engineering Department, Singhad Institute of Technology, Lonavla for his indispensable support, suggestions. In the end our special thanks to all the teaching and non-teaching staff for providing various resources such as laboratory with all needed software platforms, continuous Internet connection, for Our Project.

REFERENCES

1. "Amazon web services (AWS) - cloud computing services," Amazon Web Services, 2016. [Online]. Available: https://aws.amazon.com/. Accessed: Dec. 3, 2016.

2. "Microsoft azure: Cloud computing platform & services," Microsoft, 2016. [Online]. Available: https://azure.microsoft.com/en-us/. Accessed: Dec. 3, 2016.

3. "Google cloud computing, hosting services & APIs | Google cloud platform," Google. [Online]. Available: https://cloud.google.com/. Accessed: Dec. 3, 2016.

4. W. Hao, J.Fu, I. Yen, and Z. Xia, "Achieving High Performance Web Applications by Service and Database Replications at Edge Servers," in 28th IEEE International Performance Computing and Communications Conference, Phoenix, Arizona, 2009.

5. W. Hao, J. Fu, C. Trenkamp, and P. Prapatanant, "Achieving Better Cloud Access Experience with Server Virtualization," Journal of Computational Methods in Science and Engineering, vol. 12, no 3, sup. 1, pp. 29 – 37, 2012.

6. B. Sauer and W. Hao, "Horizontal Cloud Database Partitioning With Data Mining Techniques," in 12th IEEE Consumer Communications and Networking Conference (CCNC 2015), pp. 796 - 801, Las Vegas, NV, 2015.

7. W. Hao, J. Walden, and C. Trenkamp, "Accelerating ecommerce sites in the cloud," in 2013 IEEE 10th Consumer Communications and Networking Conference (CCNC),

8. Institute of Electrical and Electronics Engineers (IEEE), 2013.

9. J. Panettieri, "Cloud market share 2016: AWS, Microsoft, IBM, Google," in ChannelE2E, ChannelE2E, 2016. https://www.channele2e.com/2016/02/04/cloud market-share-2016-aws-microsoft-ibm-google/. Accessed: Dec. 3, 2016.