SmartCompute: Hierarchical Cloud Analysis

Karan Kamatgi
Indiana University
Bloomington
Bloomington, Indiana
krkamatg@iu.edu

Karan Kotabagi
Indiana University
Bloomington
Bloomington, Indiana
kkotabag@iu.edu

Ramyashree DG
Indiana University
Bloomington
Bloomington, Indiana
rdasego@iu.edu

ABSTRACT

There has been a lot of buzz around cloud computing and big data in the last decade leading to the evolution of technologies in both the domains. The same holds good for evolution in mobile user equipment such as smart-phones, tablets which also lead to the evolution of newer and sophisticated mobile applications. The mobile application developers need to stay tuned with emerging technology and requirements of these mobile device users. Most of the applications that need high computation running on these mobile devices are constrained by the battery capacity and the energy consumption. Sensible and suitable state-of-art solution is to offload the applications that require huge processing to a centralized and more powerful cloud solution. This offloading solution nevertheless introduces significant execution delays both in terms of latency and as well as the computation time in the cloud in case of lower intense computations. These delays are very inconvenient for the smartphone users who expect their applications to be responsive and power efficient. Also, the offloading solution is not suitable for applications which are real-time and requires the response in quick time. To overcome this critical problem, a new concept has emerged called edge computing. Mobile edge computing brings storage and computation resources at the edge of the mobile network which enables demanding applications to be executed closer to the client device and also meeting strict time response requirements. In this paper, we designed a system called SmartCompute that offloads computation only when necessary. We implemented a Hierarchy of layers for the client-server architecture. The client is at the lowest layer in the hierarchy while Mobile which acts as a Mobile Edge Server in the middle layer and the actual cloud at the topmost layer. Our architecture is a 3 layered hierarchy and the major offloading logic is implemented in the Mobile Edge device. We then Analyze the performance of local computation versus the performance of offloaded computation. Our results indicate that smart offloading yields better results where the threshold parameter depends on the application. We developed three kinds of application each belonging to the small, medium and high level of computation standards. We conducted extensive experiments to validate the key hypothesis of the paper and conclude that high-end computations tasks must be offloaded for better latency and performance results while low and medium level computation is better served at the edge of the network saving lot of energy and radio resources with reduced latency. User study analysis was also completed. Finally, we list out the open research challenges that need to be addressed in our future work.

1. INTRODUCTION

The smartphones and other smart devices are evolving at a faster rate. This evolution has lead to increase in user's expectations. This is true especially in case of smartphone users where people have strict requirements in terms of data rates and Quality of Service (QoS). One of the key constraints in the recently developed applications is that they utilize more network resources and are power hungry. Also, some of the application's performance is degraded due to increased latency as the architecture used by the application is inappropriate. The evolution of smartphones, laptops, and tablets enables to emerge new high demanding services and applications. However, new mobile devices are more and more powerful in terms of memory and CPU but these may not be able to handle applications that require large processing in a very short time. In an attempt to reduce the latency few applications have compromised in terms of battery consumption, where this problem is a limiting factor for the smartphone users to fully enjoy highly demanding applications on their own devices. This provides us the motivation for the development of 'SmartCompute' that enables the features of cloud computing exposed to mobile users at the edge of the cellular network. Most of the big enterprises are familiar with cloud computing since it is the benchmark for industry standards.

2. PROBLEM STATEMENT AND SOLUTION OVERVIEW

Cloud computing is an integral part of computer infrastructure and is a software-based model that provides access to collective pools of configurable resources such as application, storage, computer networks, and

Technical Aspect	MCC	Edge Computing
Deployment	Centralized	Distributed
Distance to UE	High	Low
Latency	High	Low
Computational Power	Ample	Limited
Storage Capacity	Ample	Limited

Table 1: High-level comparison of Mobile Cloud Computing versus Edge Computing

services. Cloud computing involves less effort in management and allows users to process and store data in either private or public cloud. Mobile Cloud Computing (MCC) brings several advantages 1. extended battery life due to offloading of energy demanding applications to the cloud 2. enables sophisticated and real-time applications to the mobile users 3. provides larger data storage capabilities to the smartphone users. All of the advantages of cloud computing comes with the cost of significant execution delay consisted in both the delivery of offloaded applications to cloud and return back also including the time of computation. This delay is significant especially in case of real-time applications that have strict time deadlines. MCC also imposes huge additional loads both on radio network as well as mobile networks. The network topology is too far away from the data creator also incurs additional latency. To overcome these problems of Mobile Cloud Computing, we focused on the more emerging concept of Mobile Edge computing leading to the development of our system 'SmartCompute'. Edge computing is an extension of cloud networks which is a collection of servers comprising a distributed network. The computing and storage resources are supposed to be in the proximity of the client(In network topological sense) which could be either IOT devices or a smartphone. Thus MEC has significantly lower latencies when compared to the conventional cloud. While MCC is fully centralized approach with farms of farms of computers usually placed at one or few locations while edge computing needs to be deployed in a distributed environment. Table 1 depicts the high-level comparison of Mobile Cloud Computing versus Edge Computing.

In this paper, we present our system 'SmartCompute' that uses edge computing whenever feasible and offloads computation if the input size is higher requiring larger processing power. The current state-of-art is to offload everything to the cloud and get the results back to the client taking a longer route in terms of network topological geography. We show that forcing offloading always will harm the application performance due to increased latency and higher energy consumed for the similar task at the edge of the network. We Configured a smartphone that acts as a 'Mobile Edge Server' by developing and deploying an Android HTTP server on to it. Any client request sent to this Edge server will receive the response only if the client and our Edge server are on the same network. Our code was deployed on the smartphone to convert it to a mobile edge server. We developed 3 simple applications to compare and contrast the performance and energy consumption of these application served at the edge server versus these application being served in the cloud. The three applications belong to 3 different range of computation such as low, medium and high. The first low range application is a simple Calculator application served at the edge server and is not offloaded as the computation requirements are very low while the second medium application we developed is a UnitConverter application that converts input values from one given unit to other desired unit. This was compared and contrasted with a cloud-based solution. Finally, the third and high-end application is Solving the NQueen's problem on 27*27 Chessboard. The NQueen's problem uses the smarCompute implementation that offloads computation only if the input size is too large to be handled at the edge server. Our current implementation uses Amazon web services T2 instances that use burstable CPU and provide a baseline of CPU performance with the ability to burst above the baseline. Our cloud instance runs on high-frequency Intel Xeon processors with burstable 1 vCPU and 2 GB RAM with Elastic Bean storage. The Java application program running on the Mobile edge server was also deployed on the cloud with help of Apache Tomcat that runs on the existing instance of the cloud. Application Resource Optimizer an open source tool developed by ATT was used to capture the trace for analysis of energy consumption, data transferred and response time. We evaluated our system by running the ARO tool for all the three types of applications and noting the energy consumed and data transferred for each of these applications. Our results indicate that for a High computational application like NQueen's, SmartCompute consumes 43 percent lesser energy than the current state-of-the-art systems. While for the medium and low compute applications the energy consumed is around 32 percent lesser for the Mobile Edge server compared to the cloud solution. Owing to the scaling requirements of the distributed nature of the application, we developed a mechanism to load balance multiple client requests by using a laptop as a load balancer and running 2 edge servers in parallel. This solution helps to scale up the usage of multiple clients concurrently since the edge servers are less busy and the request is balanced equally among the existing edge servers in the network. We strongly believe that more energy could be saved by running a sophisticated and powerful instance on the cloud with larger RAM and more vCPU.

The contributions of this paper can be summarized as follows:

- To the best of knowledge, our system 'SmartCompute' is the first system to offload based on the input size in a load balanced distributed setup that enables multiple clients to connect to multiple edge servers without overloading the traffic on a single edge server.

- We have used Application Resource Optimizer to collect data traces with VPN collector and analyze the energy consumption, data transferred for smart compute when served at the edge and when served in the cloud. 3. We conducted extensive experiments and evaluations with users and found that the smart compute method performs better reducing latency and energy consumption based on the input size of the incoming load.

- As part of future work, we plan to extend the offloading logic based on other network parameters such as network connectivity and signal strength. Also, we need to deploy our Java code on a powerful cloud instance which would relatively take less or no time in processing the high computation requests on the cloud. We also need to perform multiple user clients and multiple edge evaluation using our load balancing algorithm. We plan to implement more sophisticated load balancing algorithm such as least recently used, Shortest job first and also incorporate load balancing based on priority.

3. RELATED WORK

- WinSystems [1] performs analysis and differentiates the between cloud computing, edge computing and fog computing. They provide high performance embedded systems that could be utilized in industrial environments to enable solutions for edge computing requirements and gateways within fog platforms. Their system allow users to leverage particular IOT hardware and networks.

- Juniper Networks [4] provides Mobile Edge Computing(MEC) use cases and deployment options in their domain of networking systems. The lead deployment options would be to deploy MEC in a manner that is practical and transparent to the mobile core. They have used MEC to provide a multi-tenant hosting environment for 4G edge applications.

- Mobile Edge Computing [?] provides a survey for running computationally demand applications on mobile user equipments such as smartphones which are limited by battery capacity and their energy consumption. To tackle these challenges, computations of applications are offloaded to conventional cloud servers to handle the huge computation. The paper conveys that the cloud approach introduces latency which involves delay for making request to the cloud and getting back the result; which is undesirable for real time applications. Further, the paper explores how Mobile edge computing can help to solve this problem by processing request from user equipment at the edge of network. MEC offloads requests to cloud based on a certain offload logic.This research focuses on three key areas of mobile edge computing which are 1) Allocation of computation resources for MEC 2) Decision based on offloading logic and 3) Mobility Management. Additionally, paper discusses user oriented uses cases of mobile edge computing and finally draws conclusion that edge computing is still an immature field and smart decisions need to applied in the field of off loading logic.

4. SYSTEM DESIGN AND METHODOLOGY

The system design follows a multi-level hierarchical model where the web application is designed to run with the help of Android-HTTP libraries to study and contrast the differences between the edge and cloud server. Figure 1 demonstrates the architecture of the system on a high level. User running a web application initiates a HTTP request, this request is first intercepted by the load balancer. The load balancer implements round robin algorithm that shares the load equally to the interconnected edge servers in the network. The edge server then runs the HTTP server code which converts the existing smartphone to an actual edge server closer to the user and within the same network. Based on the input size the edge server decides whether to offload the computation to the cloud or to serve the request at the edge server itself.

4.1 System Design Overview

As compared to conventional control system, the distributed system architecture is gaining popularity in order to serve the end users by placing the control function nearer to the data center [2]. The key concept of this paper is surrounded around mobile edge computing where the mobile devices act as the edge servers and serve the end users with their request much faster, utilizing less energy as compared to the cloud device.

With the recent increase in the network traffic and its increasing tendency diverging towards the cloud endpoints on centralized server, most of network service providers are interconnecting the systems on their network to process the content closer to the users. The Figure 2 [5] gives the simple architecture of the Content distribution System.

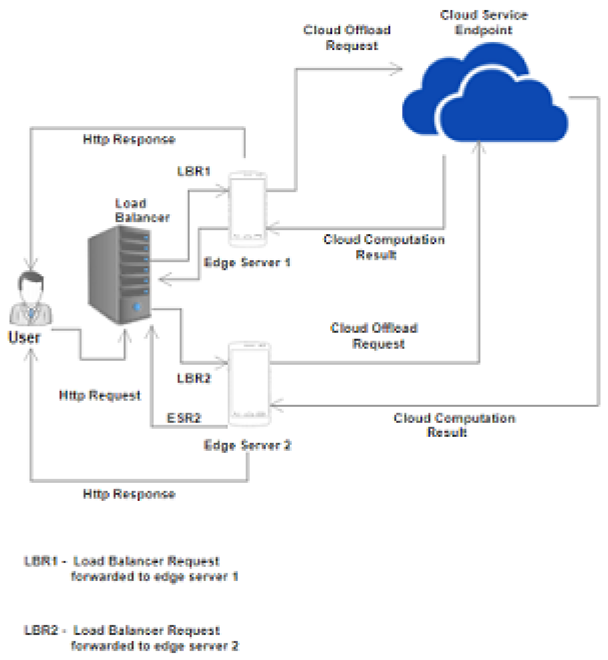


Figure 1: Architecture of the System

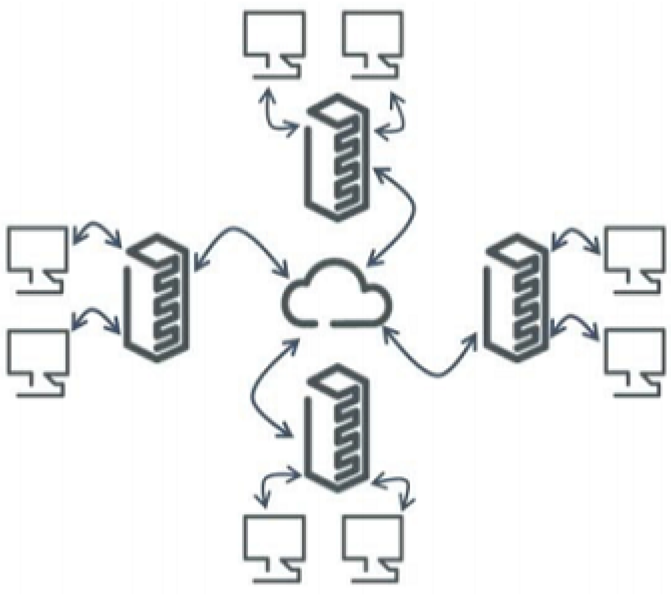


Figure 2: Content Distribution System

4.2 Motivation and System Design Highlights

With the recent increase in the traffic of cloud services, most of the applications incur network transfer delay while transferring the data and require more processing power for the client device in order to access the cloud services.

Consider a scenario in a critical environment where the medical applications are utilized from a client device in a hospital. This client device works in order to receive the heart rate prediction of a user in an ICU from the remote cloud service. It is critical for a doctor to get the updates with minimum delay over the network and receive computational results as fast as possible. If multiple edge servers are set-up using smartphones in the hospital as described in this system. Then, the client request to receive the computation results from the cloud service will greatly benefit from the Edge computing perks.

Besides the numerous advantages of the edge computing such as reduction of delay in the transfer of the data over network, reduction in the network traffic, increase in the application performance and decrease in the network latency, the system described in this paper has effective support for load-balancing between the multiple edge servers. In case of failure of one edge compute server, the other edge serves to fulfill the request for the end user.

Traditional systems have been implemented with the edge servers that are dedicated to perform the basic computation at the edge. These edge servers serve the various requests and achieve the benefits of edge computing only, but the system implemented in this paper provides end users with the more sophisticated model that achieves edge computing over the mobile devices that are effectively load balanced in a distributed environment.

4.3 System Components

The system design is a simple distributed system consisting of multiple edges and clients serving multiple requests from end users. The system design consists of the three layers which forms the primary basis of construct for the system. The three layers that primarily describe our system are Load balancing layer, Edge layer and Cloud Endpoint layer. The following sections describes the each layer in more detail.

4.3.1 Load Balancing Layer

One of the major concern in the distributed computation environment is to implement the efficient dynamic load balancing between the servers that enhances the overall performance of a system. For the purpose of efficient enhancement in performance, this layer serves to provide the load sharing between the multiple edges.

This is the layer exists between the end users and the group of edge devices serving as the edge servers. The number of the edge devices can range from one, two or more depending on the local requirement of the system. The load balancing layer acts as the intermediary between the end user and routes the request to the subsequent servers by following a specific algorithm. We will see more details related to the implementation of the Load balancing layer in the implementation section.

4.3.2 Edge Layer

This is the primary layer that serves requests and fulfills the purpose of the edge compute platform. The layer provides low-power computation compatibility to the end users to fulfill the requests from the client.

In our system called âĂIJSmartComputeâĂİ, end-users request web pages on their respective devices, which is served at the edge servers if the input size is comparably less, but if the input size is comparatively high leading to higher computation then the request is served at the cloud. Two devices are implemented as edge servers and one device operates with the Android 7.0 operating system and other device with Android 7.1 operating system.

4.3.3 Cloud Endpoint Layer

This layer has the capability to provide the high compute resources at the event of time when edge server runs out of compute resources required to fulfill the client request from system. The cloud has multiple instances of database, application and other major compute resources required to run the high computation based applications. A prototype model and demo use-case has been implemented in this system that gives the option for the end user to select a valid choice of either to carry the computation over the edge device or the cloud end-point. The AWS (Amazon Web Services) cloud is used in the backend to offload the data and get the get the computation results back from the cloud service.

5. IMPLEMENTATION

5.1 Edge Server Implementation

The edge server runs a HTTP based web application that implements the functionality like Simple Calculator, Unit Converter and computation of the Nqueens problem. The web application is hosted in the edge server that is developed with the help of android programming language. To effectively analyse the data transfer delay the applications are implemented with the continuous interaction between the client and server model. The Java Servlets have been utilized in the backend to run the web application modules. The different modules in the web application are Login, Simple Calculator, Unit Converter and Nqueens Computation. The

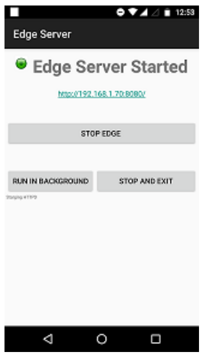


Figure 3: Edge Server Online

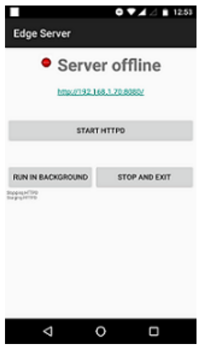


Figure 4: Edge Server Offline

Figure 3 indicates the edge server UI running from an android application when the server is online and the Figure 4 indicates the edge server application when the server is offline.

The web application running on the web server provides the choice for the users to run the application from the cloud or the edge device. The Nqueens computation is implemented with SmartCompute logic where the computation is offloaded if the input is greater than 8, otherwise the Nqueens computation happens at the edge of the server. The following figure 5 shows the Unit Converter, Calculator and the NQueens module that is being used from the client side.

5.2 Cloud Service Implementation

The cloud service is implemented using the Amazon Web Services. The Amazon Web Service provides the high compute and high reliable platform to run the applications in the cloud. A web service is deployed with the same functionalities as the Unit Converter and the NQueens problem in the AWS platform with the help of Apache Tomcat server and the same HTTP modules are served over the cloud service. The Java Servlets are utilized to implement the code deployed in the AWS Beanstalk instance. The war file is generated my compiling the code on maven and this war file is deployed using Apache Tomcat on the AWS instance. We used

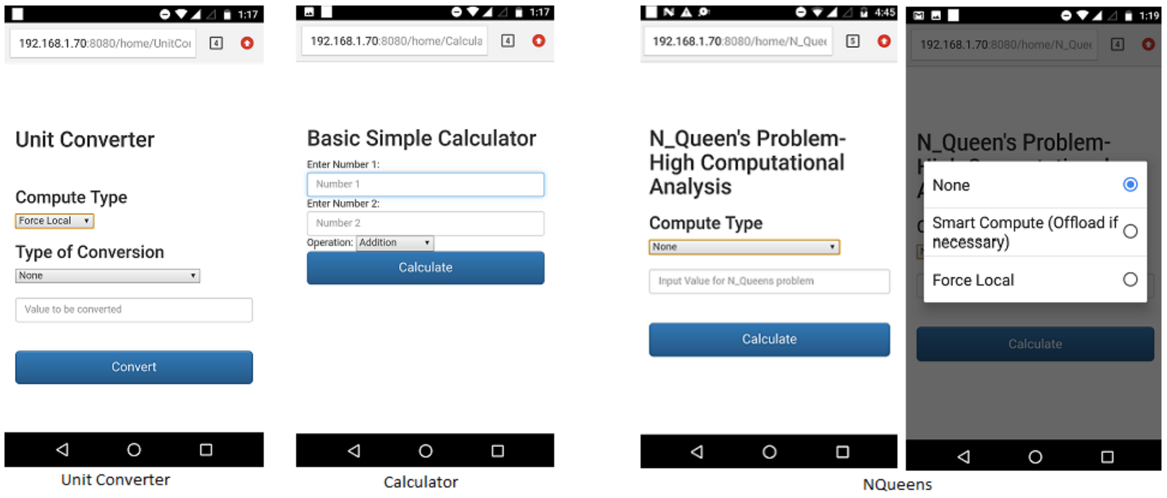


Figure 5: Unit Converter, Calculator and NQueen's Module

T2 instance of the cloud which has 1 vCPU and 2 GB of memory.

5.3 Load-Balancer implementation

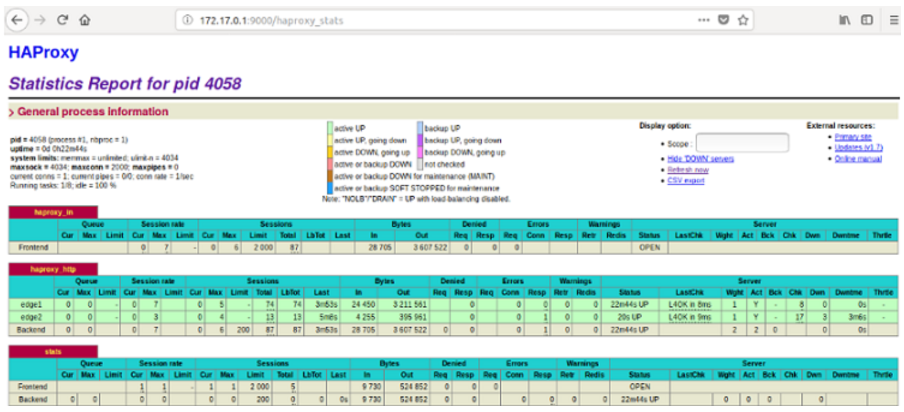
HAProxy [3] is used to implement the load balancing between the edge servers. HAproxy is the open-source load-balancer that can be easily configured in the linux system. It provides fast and reliable solution to achieve high availability solution for the TCP and HTTP based web applications. The configuration of the HAProxy is defined with the two nodes namely front-end and back-end nodes. The front-end node is the one through which the HAProxy listens for connections. The back-end nodes are the one to which HAProxy will forward the request from the user to the load-balanced servers. In SmartCompute the HAproxy is bound with the two of the edge servers and are subsequently load balanced between each other. The sample statistics of the load balancing system is described in detail with screenshots. The Figure 6 shows the statistics of the two load-balanced edge servers. The Figure 7 shows the monitoring and the statistics when the Edge server 1 server is up and Edge 2 Server is down.

6. EVALUATION

In this section we are benchmarking performance of edge server over cloud server for certain devices, in our case, it is a smartphone. We are evaluating performance in terms of energy consumed, data transferred and time taken to make the request from the device being evaluated to the edge server and cloud server. We have used three different applications of having different computational complexities. These programs are both deployed on edge server and cloud server. Series of tests are run by making requests and executing the application on both servers and result are compared to conclude.

6.1 Experimental Setup

We have used Application Resource Optimizer as a diagnostic tool for analyzing mobile web application performance by connecting it to the Android smartphone through Android Debug Bridge (ADB). This will help to get metrics like energy consumed, data used and time taken when a request is made from the device to either edge server or cloud. The edge server application is running on a different device on the same network which is accessed through the browser of the device. The edge server is an HTTP server implemented as an Android application and deployed on Android smartphone. The cloud server is deployed on small-scale Amazon AWS instance. The applications deployed on both the servers are implemented as Java Enterprise applications. Series of requests were made through the device to access and execute operations on different programs. We have used calculator and login system as small-scale applications and are only deployed on edge server since they need are not computationally intensive. As expected, results were faster by executing them on edge serve. Next, we have used unit converter which is medium scale application. This is deployed on both edge and cloud server. The application provides options to either offload the computation to cloud based on the input size or serve locally at the edge of the network. We have set threshold on input size based on which the request is passed to edge or cloud. The last application is Nqueen's program The n-queens puzzle is the problem of placing n queens on an nÃŮn chessboard such that no two queens attack each other. Given an integer input n, return all distinct solutions to the n-queens puzzle. This is computationally intensive and the same offloading logic as the unit converter is used here. The experiment is focused on comparing energy and data consumption's for executing different operations of an application on both



172.17.0.1:9000/haproxy_stats

HAProxy

Statistics Report for pid 4058

> General process information

pid = 4058 (process #1, nbproc = 1)
uptime = 0d 0h22m44s
system limits: memmax = unlimited; ulimit-n = 4034
maxsock = 4034; **maxconn =** 2000; **maxpipes =** 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/8; idle = 100 %

🟩 active UP	🟦 backup UP
🟨 active UP, going down	🟪 backup UP, going down
🟧 active DOWN, going up	🟪 backup DOWN, going up
🟥 active or backup DOWN	⬜ not checked
🟫 active or backup DOWN for maintenance (MAINT)	
🟦 active or backup SOFT STOPPED for maintenance	

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.7)
- Online manual

haproxy_in

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	7	-	0	6	2 000	87			28 705	3 607 522	0	0	0					OPEN								

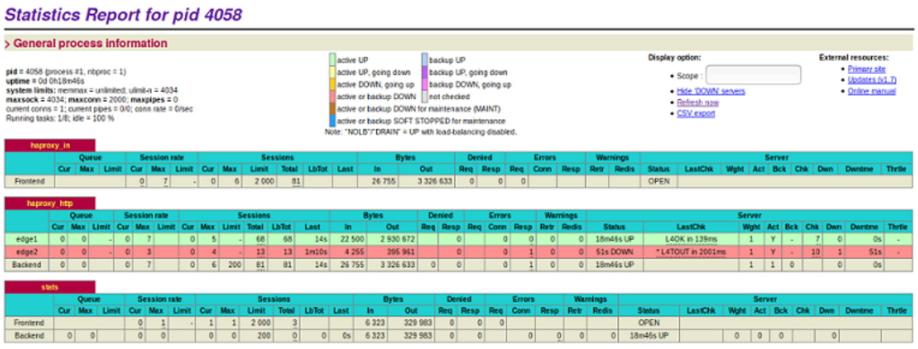
haproxy_http

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
edge1	0	0	-	0	7		0	5	-	74	74	3m53s	24 450	3 211 561		0		0	0	0	0	22m44s UP	L4OK in 8ms	1	Y	-	8	0	0s	-
edge2	0	0	-	0	3		0	4	-	13	13	5m6s	4 255	395 961		0		0	1	0	0	20s UP	L4OK in 9ms	1	Y	-	17	3	3m6s	-
Backend	0	0		0	7		0	6	200	87	87	3m53s	28 705	3 607 522	0	0		0	1	0	0	22m44s UP		2	2	0		0	0s	

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	1	-	1	1	2 000	5			9 730	524 852	0	0	0					OPEN								
Backend	0	0		0	0		0	0	200	0	0	0s	9 730	524 852	0	0		0	0	0	0	22m44s UP		0	0	0		0		

Figure 6: Both the edge servers are up and running



Statistics Report for pid 4058

> General process information

pid = 4058 (process #1, nbproc = 1)
uptime = 0d 0h18m46s
system limits: memmax = unlimited; ulimit-n = 4034
maxsock = 4034; **maxconn =** 2000; **maxpipes =** 0
current conns = 1; current pipes = 0/0; conn rate = 0/sec
Running tasks: 1/8; idle = 100 %

🟩 active UP	🟦 backup UP
🟨 active UP, going down	🟪 backup UP, going down
🟧 active DOWN, going up	🟪 backup DOWN, going up
🟥 active or backup DOWN	⬜ not checked
🟫 active or backup DOWN for maintenance (MAINT)	
🟦 active or backup SOFT STOPPED for maintenance	

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.7)
- Online manual

haproxy_in

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	7	-	0	6	2 000	81			26 755	3 326 633	0	0	0					OPEN								

haproxy_http

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
edge1	0	0	-	0	7		0	5	-	68	68	14s	22 500	2 930 672		0		0	0	0	0	18m46s UP	L4OK in 139ms	1	Y	-	7	0	0s	-
edge2	0	0	-	0	3		0	4	-	13	13	1m10s	4 255	395 961		0		0	1	0	0	51s DOWN	* L4TOUT in 2001ms	1	Y	-	10	1	51s	-
Backend	0	0		0	7		0	6	200	81	81	14s	26 755	3 326 633	0	0		0	1	0	0	18m46s UP		1	1	0		0	0s	

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	1	-	1	1	2 000	3			6 323	329 983	0	0	0					OPEN								
Backend	0	0		0	0		0	0	200	0	0	0s	6 323	329 983	0	0		0	0	0	0	18m46s UP		0	0	0		0		

Figure 7: Edge Server 2 is down

Server	Operation	Input	Energy
Edge	login	Credentials	65.74
Edge	Addition	567, 789	31.837
Edge	Subtraction	789, 567	51.335
Edge	Multiplication	123, 345	49.65
Edge	Division	789, 123	58.8

Table 2: Parameter analysis for small level computation of calculator application

Server	Calculation Type	Input	Energy	Time(s)
Cloud	fahrenheit to celsius	77	40.49	24
Cloud	Liter to gallon	77	53.4	33
Cloud	kilometer to mile	77	68.65	43
Edge	fahrenheit to celsius	77	33.35	19
Edge	liter to gallon	77	42.235	28
Edge	kilometer to mile	77	48.445	29

Table 3: Network parameter analysis for Medium Computation level application: Unit Converter

edge and cloud servers.

6.2 Evaluation Results

Energy cost: Using Application Resource Optimizer we have calculated energy consumption for different operations on different applications deployed in cloud and edge. Following are results of series of operations and respective energy consumption result.

6.2.1 *Parameter analysis for small level computation of calculator application*

Table 1 shows the results obtained from the ARO with respect to the Parameter analysis for small level computation of calculator application.

6.2.2 *Network parameter analysis for Medium Computation level application: Unit Converter*

Table 2 shows the results obtained from the ARO with respect to the network parameter analysis for medium computation level application: Unit Converter.

6.2.3 *Network parameter analysis for High computation: Nqueen's application*

Table 3 shows the results obtained from the ARO with respect to the Network parameter analysis for High computation: Nqueen's application.

6.3 Findings

We have used Unit-Converter as medium level computation and Nqueen's as high-level computation applications for comparing edge and cloud performance analysis.

6.3.1 *Unit-Converter*

Server	No. of Queens	Energy Consumed (Joules)	Time (S
Cloud	5	35.43	18
Cloud	8	40.44	30
Cloud	9	48.12	30
Cloud	11	56.42	37
Cloud	13	90.98	61
Edge	5	31.075	19
Edge	8	35.055	33
Edge	9	69.915	27
Edge	11	65.2	34
Edge	13	158.39	67

Table 4: Network parameter analysis for High computation: Nqueen's application

We can see for different operations made on unit converter, the energy consumption is high for requests made to the cloud than edge server. The computation time is negligible in this case as it is a simple calculation operation. The request response time for cloud is more compared to edge since the device and edge server are in same network and the request-response happens over internet for cloud server. Hence, total energy consumption is more for the request made to cloud server. From results, we can see that all three instances of unit converter has high energy consumption for cloud than edge server, Therefore, for non-computation intensive operations edge server is the right option.

6.3.2 *Nqueen's*

From the obtained results, we can see that for Nqueens application, as the input increases, the computation time increases and edge server takes more time to process the result than the powerful cloud server after a certain threshold. The request with input greater than 8, time taken to make request and fetch response can be ignored as computation time is large. For input 13 energy consumption for edge server is 158.39 which is 42 percent more than energy consumed for cloud server. To take advantage of edge server the Smart offload is used to send the request to the cloud when the input greater than the threshold value.

6.4 User Study

To validate the conclusions derived from our system, we conducted a detailed user study of the SmartCompute prototype with 6 users under various scenarios. We performed rigorous testing on different mobile phones acting as a client, by making request to all three levels of computations. All the tests were performed one after the other and not simultaneously. For every user, we asked them to provide inputs for the three computation applications and their responses were noted. Users were satisfied with the fast response from the edge server for the low intense computations and also the offloading to cloud requirement was justified with respect to high-level computations. We received few feedback comments for the UI changes of the application and also other use-cases for the system.

7. DISCUSSION

At present we are offloading the computation to either edge server or the cloud server based on the input parameter. As part of the future work, we plan to develop computation offloading logic based on various factors like network parameters such as signal strength and other suitable application related logic. This offloading logic can also be integrated into mobile applications. For example, for a food order mobile application, features such as fast-checkout with one item requests can be offloaded to edge servers instead of cloud servers. This responsiveness with less latency can be a high business value for customer satisfaction. Additionally, we need to deploy our Java code on a powerful cloud instance which would relatively take less or no time in processing the high computation requests on the cloud. We also need to perform multiple user clients and multiple edge evaluation using our load balancing algorithm. We plan to implement more sophisticated load balancing algorithm such as Least Recently Used, Shortest Job First and also incorporate load balancing based on priority.

8. CONCLUSION

From the various experiments and evaluation results, we conclude that both cloud computing and edge computing, if used as Hybrid model would yield better performance and better Quality of Service for the mobile web applications. SmartCompute uses proposed hybrid model that uses edge server when the input size is less than the threshold value and comparatively requires less computation. Our system uses cloud services when the input size is greater than the threshold value if the application requires high computation that could be easily served at the cloud. SmartCompute consumes 43 percent lesser energy than the current state-of-the-art systems. While for the medium and low compute applications the energy consumed is around 32 percent lesser for the Mobile Edge server compared to the cloud solution. Hence, a real-time application that uses powerful cloud instance with higher CPU and RAM must send all the low and medium computational requests to the edge server which is closer to the end user whereas all the high computational requests must be offloaded to the cloud. Thus, by using this hybrid model we can reduce latency and energy consumption for all the low and medium requests by serving at the edge of the network and similarly we can save more energy and reduce latency significantly for all the requests that need high storage and computation be served at the cloud.

9. WORKBREAKDOWN

We contributed equally with the following key points:

- Initial code deployment of the mobile edge server and implementation of the simple calculator module on the Android based web application and Load Balancing design and implementation by Karan Kotabagi.
- Implementation of the Unit converter module and Nqueen's on the Android based web application, offloading logic and ARO data capture and Analysis by Karan Kamatgi.
- Implementation of the Unit Converter module and Nqueen's as a cloud service implemented on the Amazon AWS beanstalk server, and ARO data capture and analysis by Ramyashree DG.
- All of us have equally contributed to the paper.

10. REFERENCES

[1] Cloud, fog and edge computing âĂŞ whatâĂŹs the difference. 2018.
[2] Edge computing platform. 2018.
[3] Haproxy load-balancer. Web Page, 2018.
[4] Juniper networks. 2018.
[5] S. Carlini. The drivers and benefits of edge computing. 2018.