An Experimental Study on the Impact of Execution Location in Edge-Cloud Computing

Dimitrios Melissourgos, Sishun Wang, Shigang Chen, Youlin Zhang, Olufemi Odegbile and Yuanda Wang

Department of Computer & Information Science & Engineering
University of Florida, Gainesville, Florida 32603
Email: dmelissourgos@ufl.edu

***Abstract*—On the one hand, edge computing has the advantage of distributing the load to the edges of a computer network. Local computation at the edge is bandwidth-efficient and anonymous. On the other hand, cloud computing is the choice when it comes to computationally demanding tasks and big data. In this paper, we argue for edge-cloud computing (which blends the two together) with an experimental study on the impact of execution location on application performance. We answer the question of how to determine whether it should compute a task at the edge or on the cloud and what the criteria are. We analyze the factors of response time, memory space, data availability and privacy policy. We experimentally evaluate the impact of these factors on execution location based on a network visualizer software.**

Index Terms **- location of execution, edge computing, cloud computing**

I. Introduction

In half a century, computing has evolved from centralized mainframes, which operate under the time-sharing model, to networked computers under the client-server model, then back to centralizing hardware, software, infrastructure and IT support in huge data centers. Recently, it reverses again towards the edge of the Internet under a new model of edge computing [4]. By moving computation to where the users are and where the data is produced, the edge computing model is very appealing in reducing application response time, improving user experience, saving bandwidth cost, and enhancing data privacy [1]. It is defined as "a paradigm in which resources for communication, computation, control and storage are placed at the edge of the Internet, in close proximity to mobile devices, sensors, actuators, connected things and end users." [2]

While edge computing sits on one end of the spectrum, the other end is cloud computing, which offers a flexible, reliable and cost effective infrastructure to end-users. In-between, there are other models that take advantage of both edge and cloud computing [13], [15], [18]. Platforms such as cloud offloading extenuate the heavy load of computationally demanding applications from the edge machines and shift it towards the more capable data centers [3]. There are two challenges in these edge-cloud models: One is how to determine which components of an application program should run at the edge devices and which components should run on the cloud. The other problem is how to execute a program with some of its components stored at different locations. The second problem has been extensively studied in the literature, in the form of cloud offloading, mobile cloud computing and mobile edge computing. In this paper, we focus on the first problem through an experimental study to confirm the importance of allowing the flexibility in location choices for application tasks.

We first establish the factors that contribute the most to determining whether a component should be executed at the edge or in the cloud. We then carry out an experiment-based investigation to evaluate the impact of execution location on application performance. We use a software system developed in our lab as the experimental platform. Our experiments confirm the hypothesis that if the data is available to both the edge and the cloud, it is beneficial to partition the tasks and execute them either at the edge or in the cloud to optimize the application response time. In particular, low-complexity tasks are faster processed using only the edge device, but for high-complexity tasks, the cloud often outperforms the edge by far, even after including the communication time for exchanging the tasks and their results between the edge and the cloud. Moreover, we show that depending on data size and location, space restrictions and time constraints, computation tasks within the same application can benefit from both edge computing and cloud computing.

The rest of the paper is structured as follows: In Section II we present the background to our work and in Section III we examine the different factors that are significant for the decision of the location of execution of an application. In Section IV we conduct a set of experiments, in order to determine which parts of the same application are better executed on the cloud and which perform better when they are executed on an edge machine. In Section V we explain the similarities and differences between our work and virtual machines, cloud offloading and edge computing. Finally, we draw the conclusion and discuss our future work in Section VI.

II. Background

For today's software development, the distinction between an application running in the cloud and an application running at an edge device is clear — the software is designed according to the resource constraints, the data privacy policies

This research is supported in part by National Science Foundation under grant CNS-1909077 and in part by a grant from Florida Center for Cybersecurity.

978-1-7281-8275-9/20/$31.00 ©2020 IEEE
DOI 10.1109/BigCom51056.2020.00028

and the response time requirements specific to the machines that the app is intended for.

On the one hand, when a client machine sends a request to be computed in the cloud, time has to be spent for the request to be packed and propagated to the cloud and also for the results to be sent back to the client machine. Therefore, for low-complexity tasks, it may be preferred to execute the tasks at the client (edge). In addition, there may be strict delay constraints on the tasks or privacy requirements on data transfer, which prevent certain computations from being done in the cloud.

On the other hand, there are tasks that are not suitable for local execution. For example, when applications perform complex computation or process big data, the client (edge) machine may not possess sufficient storage, RAM and processing power to finish the tasks on time. In such cases, cloud computing is preferable as it offers virtually unlimited resources.

Now consider an application with tasks which can be executed either locally or in the cloud. Suppose there are a mix of trivial and complex tasks that are user-interactive. Further suppose that its input consists of both small and large amounts of data, with some data stored locally and other data in the cloud. For locally stored data, there are two main reasons: 1) It is collected locally and it is bandwidth-efficient to store locally, and 2) there may be a privacy policy that prevents the data from being uploaded onto the cloud.

For such an application (like the one used in our experimental study), an integrated edge-cloud computing model appears to be desired, with some tasks running locally at the edge and other tasks running remotely on the cloud. The question that we want to answer in this paper is which factors will impact the decision on execution location. To answer the question, we will first give our hypothesis on a list of factors and perform an experimental study to investigate the validity of the hypothetical factors.

III. Factors on Execution Location

The purpose of this paper is to establish some basic facts underlying the principles in determining which location each part of the application should be executed at. Our method is to use an experimental study to examine the impact of various factors on the choice of execution location and the resulting application performance. Before carrying out the experimental investigation, in this section, we analyze several potential factors including response time, memory space, data availability and privacy policy. We conjure through reasoning their possible impacts, which will be verified through experiments in the next section.

A. Factor 1: Response Time

Response time is a key factor on user experience with application performance. To choose between whether executing an application component at the edge or in the cloud, one deciding criterion is which choice results in shorter response time. On the one hand, if it takes more time for the edge to complete processing than the combined time of processing at the cloud and transferring the result back to the edge, we shall let the cloud do the computation for smaller response time to user requests. With the backup of cloud computing, the edge device can appear to users that it has unbounded computing scalability. On the other hand, if it takes less time to compute at the edge, we shall do so. More specifically, consider a computation $A = \{a_1, a_2, ..., a_n\}$, where a_i, $1 \leq i \leq n$, are tasks of A. Suppose that task a_i takes x time to compute on the edge device and y time to compute on the cloud, where $x > y$ due to the cloud's superior resources. Suppose that it takes z time for the edge device to pack the request and send it to the cloud, and z' time for the result to be sent back. By properly determining the execution location, we can control the execution time of a_i to $\min\{x, y + z + z'\}$. In order to learn whether certain computation is faster at the edge or on the cloud, we need to perform the computation on both until adequate execution data (such as x, y, z and z') are collected. Based on them, future decisions on the placement of execution will be made.

B. Factor 2: Memory Space

Memory space is another factor in considering execution placement especially for resource-limited edge devices. If a computation requires more system memory than what is available on an edge device, or if it produces too much data for the device to store in its permanent storage, we will need the cloud to process the data instead. In our experimental platform, this scenario is prominent. However, the amount of system memory or disk space needed by a computation task is often not known before hand. Therefore, whether to execute the task at the edge or the cloud due to the space factor may not be pre-known. In this case, we need, again, to collect execution data on space usage by running tasks on both edge and cloud initially or periodically for updates.

C. Factor 3: Data Availability

Data availability is an important factor for execution location, considering the overhead of data transfer. When data is produced at an edge device (such as a camera), it makes sense to process the data at the edge. However, if that device does not have adequate resources (such as system memory) for processing the data, or it takes an excessive amount of time to complete the task, an alternative will be transferring the data to the cloud and processing it there. There also exist situations where data is stored on the cloud, not at the edge. For example, when a user runs an application which displays a network traffic map, the traffic data generated from the routers will be likely stored at the server. In this case, it is natural to execute data-processing tasks by the server and transfer the results to the user's computer for display. More complicated is the situation where partial data is at the edge and partial data is at the cloud. We will have to either move the data to one location or partition the computation between the edge and the cloud.

D. Factor 4: Data Privacy Policy

Finally we consider data privacy. Applications often process private data, such as the name, location, Social Security Number or medical records of an individual. For these types

of applications privacy concerns arise as to whether the data should be transferred to a data center or stay at the edge. A data privacy policy may restrict the transfer of data entirely or partially, which should be factored into the consideration of which processing tasks are performed locally and which are performed remotely with appropriate data transfer.

In practice, the location of execution is likely to be affected not by just one factor, but by multiple ones. In other words, in order to determine where to perform a computation, we need to consider where the data is, whether the edge has sufficient system memory, whether it will take the edge too much time to finish, and whether there is a policy that will dictate the location of computation.

IV. Experimental Study

We perform an experimental study on the importance of execution location and how the factors in Section III will impact on the decision of execution placement.

A. Experiment Platform

The experimental platform is a software developed in our lab, called *network visualizer*, which captures network traffic data, processes them, and answers user queries on communication patterns and statistics in graphical presentation. Our data sets are raw traffic records from campus routers. To support the experimental study in this paper, the software is made to run in two modes: the *edge mode* where processing/querying tasks are performed at an edge computer, and the *edge-cloud mode* where processing/querying tasks are offloaded from the edge to the cloud. The experiments are performed by comparing the performance of the two modes to demonstrate the importance of execution location. With the two modes together, we will be able to divide tasks between the edge and the cloud.

The software supports three types of tasks. (1) Queries: given a source (e.g., continents, countries, ISPs, subnets, individual IP addresses, etc.) and a destination, answer fine-grained traffic statistics. (2) Attack Analysis: identify attacks (e.g., scanning, worms, and DDoS attacks). (3) Traffic Processing: parse raw traffic records to set up the data structures for queries. The tasks we perform with this application vary significantly in demand for computation, system memory and data, offering a suitable test case for collaborative edge-cloud computing.

We use a desktop to emulate an edge device from which a user accesses network statistics and a high-power server to emulate the cloud, which provides large backup resources for the edge. The desktop edge device runs on an AMD FX-8350 CPU with 8GB of RAM, Windows 10 64-bit and a hard drive of 1 TB. It uses a commercial 100 Mbps download / 10 Mbps upload internet connection to communicate with the server, which is located 5 miles away from the edge device. The server is equipped with an Intel ES-2643v4 CPU, 256 GB of RAM, Linux Ubuntu 18.04 LTS 64-bit, and several hard drives of total 24 TB. The network traffic records we use in the experiments are 21.7 GB, collected over a span of 2 hours, with 68 million TCP flows of 4.5 billion packets.

Raw traffic data are compressed NetFlow records. We parse them into a complex internal data structure (discussed further in this section), where we store all the IP addresses/prefixes that correlate to continents, countries, ISPs and organizations. We map the IP addresses/prefixes in the NetFlow records to their owners for display purpose. We categorize the information from traffic records in a hierarchical structure of hashmaps. The total setup time it took the desktop to parse all of our traffic records into the hashmaps was 2.4 hours, while the server needed 112 seconds for the same setup.

For the experiments of this paper, the network visualizer software we developed is specifically designed to implement several functions, with the goal of proclaiming the behavior of various systems in different situations that could be encountered in a day-to-day use. Although only one simulation software was used, the internal data structure is designed to demonstrate the benefits and drawbacks of the edge and edge-cloud modes in situations where all four factors (response time, memory space, data location and privacy policy) are considered.

Also, in order to provide clarity about the experimental process, it is important to provide further explanation about the internal data structure and functionality of our software. Initially, we correlate all the IPs to their respective ISPs. For example, when the user poses queries such as “Visualize the connections between a specific campus IP and the Google ISP” (Table I, Query 12), our software will provide an interactive graph of connections between the campus IP and all the IPs that are owned by Google. The information of which IPs belong to which ISP has been loaded into the system memory during the setup period mentioned above. More specifically, the software will show the duration of the connections, the number of packets transferred and the total size of information exchanged. Subsequently, the user can choose to click on one of the provided IPs and the network visualizer will show the details of that specific connection.

Furthermore, we correlate the ISPs to the countries they belong to. Based on that categorization, if the user requests the visualization between a campus router and Canada (Table I, Query 8), the software will provide them with a graph that shows every Canadian ISP which has been used to provide communication with the requested campus router. Finally, we follow the same categorization pattern to correlate countries with continents for queries such as Table I, Query 5.

B. Response Time

Table I shows the response time for a list of queries under both the edge mode and the edge/cloud mode. The first column is the query number, and the second column gives the source and the destination of each query. The third column, Edge Mode, shows the response time when queries are processed at the edge where all data are stored at the edge as well. The fifth column, Edge-Cloud Mode, shows the response time of processing the queries in the cloud, including the time it takes the edge to send the query to the cloud and the time it takes the cloud to send the result back. Under this mode, all the raw data is stored on the cloud.

Query	Source-Destination	Response Time (ms)	
		Edge Mode	Edge-Cloud Mode
1	All Campus Subnets - All Continents	15	94
2	Subnets behind Campus Router 1 - All Continents	16	125
3	All Continents - Subnets behind Campus Router 1	2797	187
4	Campus IP 10.248.91.67 - All Continents	16	125
5	All Campus Subnets - Countries in North America	4323	140
6	Subnets behind Campus Router 1 - Countries in North America	2187	141
7	Campus IP 10.248.24.67 - Countries in North America	0	125
8	Subnets behind Campus Router 1 - ISPs in Canada	31	218
9	Campus IP 10.248.22.165 - ISPs in the United States	15	78
10	All Google LLC IPs - All Campus Subnets	60	109
11	Subnets behind Campus Router 1 - All Google LLC IPs	48	94
12	Campus IP 10.248.29.53 - All Google LLC IPs	13	125
13	IP 8.8.8.8 - All Campus Subnets	11	78
14	IP 8.8.8.8 - Subnets behind Campus Router 1	9	125

TABLE I
RESPONSE TIMES OF TRAFFIC QUERIES

In our first query of Table I, we request the traffic statistics between all the campus subnets and the rest of the world. This query takes 15 ms to complete under the edge mode and 94 ms under the edge-cloud mode. It is clear that such a query should be performed at the edge without going through the cloud. Even though we measured that the time it takes for the server to compute the task is only 2 ms, the communication time makes the edge-cloud choice slower.

However, in other cases such as query 5, we request the statistics between all the campus subnets and the continent of North America. Our software has to produce the statistics between all campus subnets and all countries in North America, create a graph with the information, and display it to the user. It takes 4323 ms to complete the query under the edge mode, whereas it takes only 140ms to complete under the edge-cloud mode, which includes the communication time. For this task, we prefer the query to be processed remotely in the cloud.

Although some of the queries in Table I seem to have similar Source-Destination pairs with different response times, it is important to explain which calculations take place after a query is posed. For example, the queries 2 and 6 from Table I might seem similar but they have very different response times. In the case of query 2, it takes the Edge Mode 16 ms and the Edge-Cloud Mode 125 ms to provide the answer. On the other hand, in the case of query 6 it takes the Edge Mode 2187 ms and the Edge-Cloud Mode 141 ms to answer the query. As mentioned in Section IV-A, in query 2 the system only has to process data from the seven continents after fetching the data from the main memory. The calculations of which connections have taken place between which specific countries, ISPs and IPs have already taken place during the setup time. The system merely has to process the aggregate data from only seven sources. In comparison, the calculation that takes place in query 6 is vastly larger. The system has to calculate the duration of the connections, the number of packets transferred and the total size of packets for all 23 countries of North America. Additionally, it is not surprising that the system processes a large volume of data when the query has North America as its destination, since most packets are transferred within the United States.

The same observation is true for the queries 2 and 3, where there is a large gap between their respective response times. They are seemingly the same query with reversed source and destination. However, the distinction between the source and destination makes a big difference. In the case of query 2, the system performs a lookup between campus router 1 and the seven continents. In the case of query 3 the system has to perform a lookup for far more than seven components, since there are several subnets behind campus router 1.

Overall, Table I demonstrates the significant impact of execution location on query response time. There are queries that save more than an order of magnitude in response time by executing on the cloud instead of at the edge. There are other queries with multifold reduction in response time when executing at the edge instead of on the cloud. Therefore, it is highly beneficial for an application to have the flexibility of choosing the location of execution for its tasks.

C. Memory Space

Memory space is another important factor in the decision of execution location. In contrast to the response time discussed in the previous Section, where a computation might take a long time to complete, if a task requires more system memory than what is available on the edge machine, the computation will not be completed at all. In this Section we explore the effect of limited system memory on an edge machine, by making requests on the network visualizer software that are more demanding than those of Table I.

During our testing there were some cases (presented at Table II), where the edge machine run our of system memory when operating as a standalone, since 8 GB of RAM was not enough for the computation. For example, the first query of

Query	Source-Destination	Query Response Time (ms)	
		Edge Mode	Edge-Cloud Mode
1	All Campus Subnets - ISPs in the United States	Out of Memory	1234
2	ISPs in the United States - All Campus Subnets	Out of Memory	125
3	ISPs in the United States - Subnets behind Campus Router 1	Out of Memory	94
4	All Campus Subnets - All Google LLC IPs	Out of Memory	422

TABLE II
QUERIES THAT CAUSED AN OUT OF MEMORY ERROR

Attack	Threshold	Query Response Time (ms)	
		Edge Mode	Edge-Cloud M.
Scanner	1,000,000	Out of Memory	4156
	10,000	Out of Memory	6250
	100	Out of Memory	6422
Worm	1,000,000	Out of Memory	8150
	10,000	Out of Memory	7157
	100	Out of Memory	6558
DDoS	1,000,000	Out of Memory	2953
	10,000	Out of Memory	2375
	100	Out of Memory	2390

TABLE III
RESPONSE TIME FOR DETECTING MALICIOUS USE OF THE NETWORK

Table II requests the different flow sizes between the campus routers and all the ISPs of the United States. Our software retrieves the flow size of each ISP in the country and creates the graph. The data from all the United States ISPs is too large to be held at the edge computer's RAM and the software shows an "Out of Memory" error. However, the server is able to handle such requests and completes the computation.

In addition to our previous testing, we simulated three different network attacks: a Scanner, a Worm and a DDoS (Table III). Each one of them produced enough data to cause an "Out of Memory" error to the client standalone, while the server was able to handle the load. The "Threshold" value mentioned in Table III is the number of connections to the network before it is considered malicious and detected as such.

Lastly, it is worth noting that the data used in this experiment occupied only 21.7 GB of storage space. Therefore, the edge's 1 TB hard drive was able to store it locally. However, a similar application that processes bigger data can easily run out of storage. To summarize, the limited system memory and storage of an edge system will not be sufficient for demanding computations and these types of tasks will always have to be transmitted to the cloud.

D. Data Availability and Privacy Policy

During the first set of experiments, discussed in the previous two subsections, we keep the raw data on the edge machine, whenever it was computing as a standalone device. In the scenario where the requests were sent to the server, we kept all the data on the server. In this Section we consider the importance of the location of the data, as well as the anonymity restrictions that could be placed on it. In order to simulate a scenario of a privacy policy enforcement which dictates that the data has to be stored locally, we run a second set of experiments and keep the data exclusively on the edge machine. Therefore, in this subsection we explore the data location and the privacy policy factors and discuss how they affect the computation.

For this experiment, we feed the raw traffic data to the edge computer and measure the time it takes it to create the internal data structure. Afterwards, we follow the same process for the edge/cloud system, but instead of completing the computation locally, we make the edge send the data to the cloud. Therefore, the transmission includes the request of creating the data structure and the data that populates it. The first column of Table IV represents the data size in about 750 MB intervals. Each raw data file contains 5 minutes of network traffic between the campus routers and internal or external IPs. The second column named "Edge Mode" represents the time in seconds it takes for the edge machine to process the data and create the data structure. Finally, the column "Edge/Cloud Mode" shows the total time it takes to complete all three of the following tasks: 1) The edge sends the data and its request to the cloud, 2) The cloud processes the data and creates the data structure and 3) The cloud sends the data structure back to the edge.

Raw Data Size (GB)	Setup Time (sec)	
	Edge Mode	Edge/Cloud Mode
0.78	44	332
1.52	58	705
2.26	72	984
3.02	91	1316
3.78	204	1648
4.54	780	1971
5.32	1648	2314
6.08	3727	2642
6.84	4095	2965
7.58	4305	3288
8.32	4805	3607
9.08	5305	3939

TABLE IV
SETUP TIME WHEN RAW DATA IS SENT ALONG WITH THE QUERY

In this scenario, whenever the computation needs to take place on the cloud, the edge has to transmit the data related to the query, in addition to the query itself. In a real world

application, this data would have to be anonymized or encrypted before it is sent to the cloud. Interestingly, we found that the more data that is sent to the cloud, the more efficient the request becomes. In Table IV we observe a turning point after 5.32 GB of data. Until that request, it is preferable for the edge computer to perform the computation by itself. From 6.08 GB of input onward, it consistently becomes more time efficient to transfer the data to the cloud for computation. The reason is that even though the transmission takes longer whenever more data is transmitted, the computation on the edge machine also takes longer, due to the larger input and more complex computation.

E. Summary

During our evaluation we explore all four performance factors discussed in Section III. We evaluate the "Memory Space" factors of Edge-Cloud Computing, by observing which requests cause our client to run out of system memory. Additionally, for this proof of concept design we used 2 hours worth of gathered data from our campus, which resulted in 21.7 GB of data, for our first set of experiments and 9.08 GB of data for the second set. In a perpetually running data center, the traffic data would be significantly larger and the storage capacity of an edge machine will be insufficient, which is another element of the "Memory Space" factor. We also explored the "Response Time" factor by accurately measuring the execution time for each task on each system, and comparing their performance. Finally, we simulated a privacy policy enforcement scenario, which prevents the data from being stored remotely. Based on that policy, we measured the time it takes to create the data structure remotely, by sending all the data needed for this task. We found that even though for small requests it is preferable to execute the task on the client, there is a turning point at which the computation becomes more time efficient when executed by the cloud.

This experiment demonstrates that any application similar to ours, would perform better if the location of execution was not predetermined. If the code is available to both the edge and the cloud, and there existed a mechanism to decide where each query should be computed at, we would observe a big performance improvement, by assigning the individual sub-tasks to the proper machine. This decision can be made based on the four performance factors discussed in Section III. Finally, we verified that all of our proposed performance factors significantly affect the optimal location of execution.

V. Related Work

Virtual machines [5]–[12] can have a seemingly similar result to our design, but there are some key differentiating factors. In the case of virtual machines, the edge computer at the user side serves only as a terminal and the computation is entirely performed by virtual machine instances at the data center. Current virtual computing resources often refer to a computer augmented with virtual disks or other resources from a remote server. This is hardware augmentation to a computer, not an integration of local and remote resources at the level of program execution of individual applications, let alone managing execution locations of program components on the fly for optimized responsiveness and scalability. Our approach differentiates from that model, since it integrates local and remote resources at the level of program execution and manages the execution locations of program components.

Our work shows similarities to cloud computing [19]–[24] and mobile edge computing [25]–[33]. However, the location of execution is known and determined before the execution starts, in both the cloud computing and edge computing models. The location of execution is irrespective of the application and it certainly cannot be dynamically changed during the execution. In cloud computing, the computation only takes place at the cloud or the server, while in mobile edge computing the computation takes place at the edges of the network.

The model that is closest to our approach is cloud offloading [13]–[17]. Examples of the cloud offloading architecture are the MAUI [18], the CloneCloud [15] and the ThinkAir [13] projects. This prior work mainly examines the technical aspect of dividing an application in smaller parts, in order to save energy on mobile devices. More specifically, MAUI constructs a linear programming formulation based on the offloading benefit measurements. These measurements are mostly determined by the energy consumption savings and CPU cycles savings. Similarly, "...CloneCloud partitions applications using a framework that combines static program analysis with dynamic program profiling and optimizes execution time or energy consumption using an optimization solver." [15]. Finally, ThinkAir implements cloud offloading by creating multiple Virtual Machines on the cloud. Again, the only parameters used to optimize their model are execution time and energy. The user can choose to optimize for energy savings, fast execution time or a balance between them. However, there is no mention of the data location and privacy policy enforcement. Both of these parameters can vary, which would greatly affect the location of execution and subsequently the performance.

Edge-Cloud Computing embodies the benefit of caching and offloading, but its division of work between the edge and the cloud can be dynamic and does not have to be determined before hand. It may automatically evolve as components of a program may change their locations of execution over time when opportunity of performance improvement arises. In this paper we also attempt to establish the performance metrics that define the decision for the location of execution. We go beyond the time and energy factors, since an edge device does not necessarily have to be a mobile device. This work is absent from the prior art.

VI. Conclusion and Future Work

While both data centers and personal computers are used to complete computing tasks, the location of execution of each program is currently predetermined. The software developers are usually aware of the resources, delay tolerance and privacy policy of the computer which their application is going to run on. Therefore, they are able to create their program in order to execute the code on a specific type of computing environment, which is either a personal computer or a data center. In any case, the location of execution of that code

is pre-known. There also exist system architectures which propose a dynamic execution model, utilizing both the edge machine and the cloud. However, their models only consider the "time" and "energy savings" factors. In this paper we examine the possibility of software execution under the prism of a non-predetermined environment. Then we proceed to define the factors that mostly influence the decision for the execution location. Finally, we evaluate our factors in Section IV and verify that all of them contribute to the decision for the location of execution of a task.

Our work has a novel way of looking at this problem by examining the processing times for various scenarios and proposing a new model where applications, or parts of them, would run either on the edge or the cloud, without that location being predetermined. It can be decided based of the factors that affect the preferable final result.

In the future we will construct experiments which test our proposed work in the aforementioned key factors more extensively, by diversifying our applications even more. We are also determined to examine these factors and create models that utilize both locations of execution, as well as propose how the code for such models should be technically partitioned. Finally, we are going to design a method which makes the choice of location of execution automatically, without user input and minimal overhead, based on our performance factors.

Machine learning algorithms can also be deployed in order for a machine to be trained on which application metrics distinguish the location of execution between the edge and the server. This will result in the machine extrapolating its previous knowledge towards applications it has never seen before.

REFERENCES

[1] W. Shi and J. Cao and Q. Zhang and Y. Li and L. Xu, "Edge Computing: Vision and Challenges", IEEE Internet of Things Journal, 3.5 (2016): 637-646

[2] NSF Workshop Report on Grand Challenges in Edge Computing, http://iot.eng.wayne.edu/edge/NSF%20Edge%20Workshop%20Report.pdf, 2016.

[3] Ma, X., Zhao, Y., Zhang, L., Wang, H. and Peng, L., 2013. When mobile terminals meet the cloud: computation offloading as the bridge. IEEE Network, 27(5), pp.28-33.

[4] Shi, Weisong, and Schahram Dustdar. "The promise of edge computing." Computer 49.5 (2016): 78-81.

[5] "Amazon Elastic Compute Cloud (Amazon EC2)", http://aws.amazon.com/ec2/

[6] VMware Inc. "VMware Capacity Planner", http://www.vmware.com/products/capacity-planner/

[7] X. Li and J. Wu and S. Tang and S. Lu. "Let's Stay Together: Towards Traffic Aware Virtual Machine Placement in Data Centers" in Proc. of IEEE INFOCOM, April 2014.

[8] L. Chen and H. Shen. "Consolidating Complementary VMs with Spatial Temporal-awareness in Cloud Datacenters" in Proc. of IEEE International Conference on Computer Communication(INFOCOM), April 2014.

[9] P. Hoenisch and C. Hochreiner and D. Schuller and S. Schulte and J. Mendling and S. Dustdar. "Cost-Efficient Scheduling of Elastic Processes in Hybrid Clouds" Proc. of IEEE International Conference on Cloud Computing, June 2015.

[10] S. Shi and C. Wu and Z. Li. "Cost-Minimizing Online VM Purchasing for Application Service Providers with Arbitrary Demands" Proc. of IEEE International Conference on Cloud Computing, June 2015.

[11] W. Song and Z. Xiao and Q. Chen and H. Luo. "Adaptive Resource Provisioning for the Cloud Using Online Bin Packing" IEEE Transactions on Computers, 63.11 (2015): 2647 - 2660, IEEE Computer Society.

[12] T. Carli and S. Henriot and J. Cohen and J. Tomasik. "A packing problem approach to energy-aware load distribution in Clouds" Sustainable Computing: Informatics and Systems, 2015, in press.

[13] Kosta, Sokol, et al. "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading." 2012 Proceedings IEEE Infocom. IEEE, 2012.

[14] Kumar, Karthik, and Yung-Hsiang Lu. "Cloud computing for mobile users: Can offloading computation save energy?." Computer 4 (2010): 51-56.

[15] Chun, Byung-Gon, et al. "Clonecloud: elastic execution between mobile device and cloud." Proceedings of the sixth conference on Computer systems. ACM, 2011.

[16] Rudenko, Alexey, et al. "Saving portable computer battery power through remote process execution." ACM SIGMOBILE Mobile Computing and Communications Review 2.1 (1998): 19-26.

[17] Hunt, Galen C., and Michael L. Scott. "A guided tour of the Coign automatic distributed partitioning system." Proceedings Second International Enterprise Distributed Object Computing (Cat. No. 98EX244). IEEE, 1998.

[18] Cuervo, E., Balasubramanian, A., Cho, D.K., Wolman, A., Saroiu, S., Chandra, R. and Bahl, P., 2010, June. MAUI: making smartphones last longer with code offload. In Proceedings of the 8th international conference on Mobile systems, applications, and services (pp. 49-62). ACM.

[19] Tong, Liang, Yong Li, and Wei Gao. "A hierarchical edge cloud architecture for mobile computing." IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. IEEE, 2016.

[20] Guo, Songtao, et al. "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing." IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. IEEE, 2016.

[21] Chang, Zheng, et al. "Energy efficient resource allocation for wireless power transfer enabled collaborative mobile clouds." IEEE Journal on Selected Areas in Communications 34.12 (2016): 3438-3450.

[22] Hou, I., et al. "Asymptotically optimal algorithm for online reconfiguration of edge-clouds." Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing. ACM, 2016.

[23] Chen, Meng-Hsi, Ben Liang, and Min Dong. "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point." IEEE INFOCOM 2017-IEEE Conference on Computer Communications. IEEE, 2017.

[24] Kao, Yi-Hsuan, et al. "Hermes: Latency optimal task assignment for resource-constrained mobile computing." IEEE Transactions on Mobile Computing 16.11 (2017): 3056-3069.

[25] A. Ahmed and E. Ahmed. "A Survey on Mobile Edge Computing." in Proc. of 10th IEEE International Conference on Intelligent System Control, 2016.

[26] Jararweh, Yaser, et al. "Software-defined system support for enabling ubiquitous mobile edge computing." The Computer Journal 60.10 (2017): 1443-1457.

[27] Hou, Tingting, et al. "Proactive content caching by exploiting transfer learning for mobile edge computing." International Journal of Communication Systems 31.11 (2018): e3706.

[28] Xu, Jie, Lixing Chen, and Shaolei Ren. "Online learning for offloading and autoscaling in energy harvesting mobile edge computing." IEEE Transactions on Cognitive Communications and Networking 3.3 (2017): 361-373.

[29] Taleb, Tarik, et al. "Mobile edge computing potential in making cities smarter." IEEE Communications Magazine 55.3 (2017).

[30] Jeong, Seongah, Osvaldo Simeone, and Joonhyuk Kang. "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning." IEEE Transactions on Vehicular Technology 67.3 (2017): 2049-2063.

[31] Wang, Feng, et al. "Joint offloading and computing optimization in wireless powered mobile-edge computing systems." IEEE Transactions on Wireless Communications 17.3 (2017): 1784-1797.

[32] Chen, Xu, et al. "Exploiting massive D2D collaboration for energy-efficient mobile edge computing." IEEE Wireless Communications 24.4 (2017): 64-71.

[33] Zhang, Ke, et al. "Cooperative content caching in 5G networks with mobile edge computing." IEEE Wireless Communications 25.3 (2018): 80-87.