2019 IEEE International Conference on Edge Computing (EDGE)

Towards Analyzing the Performance of Hybrid Edge-Cloud Processing

Dumitrel Loghin
Department of Computer Science
National University of Singapore
dumitrel@comp.nus.edu.sg

Lavanya Ramapantulu
International Institute of Information Technology
Hyderabad, India
lavanya.r@iiit.ac.in

Yong Meng Teo
Department of Computer Science
National University of Singapore
teoym@comp.nus.edu.sg

***Abstract*—While edge computing is gaining traction, organizations operating in geographically distributed locations are still using cloud computing to collect and post-process data. In this context, it is useful to analyze the performance trade-offs of cloud-only, edge-only and hybrid edge-cloud processing. To facilitate this analysis, we provide an analytic model validated by measurements on representative edge and cloud platforms. Our model is easy to apply even without performing measurements on the target edge hardware, as long as useful performance specifications are available. Our measurement-driven analysis reveals a diverse performance landscape where there is no clear winner among cloud-only, edge-only and hybrid processing. However, application characteristics and edge-cloud transfer bandwidth are the key factors affecting performance.**

***Index Terms*—edge computing; cloud computing; hybrid edge-cloud computing; performance analysis; analytic model; measurements**

I. Introduction

The last few years have seen the rise of edge and fog computing [19], where processing is not done exclusively on the cloud, but rather at the edge of the network. Edge computing is enabled by three key factors, (i) the exponential increase in data collected by the countless number of devices connected to the Internet, (ii) the performance improvements of low-power devices and (iii) the real-time requirements of critical decision-making services, such as autonomous driving. The exponential increase of data exerts pressure on the networking connections between edge devices and cloud processing clusters [20]. But data processing at the edge has the potential to reduce both the networking pressure and service response time. This edge processing is possible today due to the increasing performance of edge devices [13].

Many real-world setups that include edge computing are also using cloud to aggregate, store and post-process the data [19]. In the context of hybrid edge-cloud computing, it is not straight-forward to decide how much processing should be done at the edge and on the cloud to meet execution time deadlines. This decision must also consider the edge-cloud bandwidth and the fact that it may fluctuate with time. Our aim is to conduct an analysis using both models and measurements to assess the effect of the edge-cloud bandwidth on the overall execution time of big data analytics. Our analysis exposes the cases where edge-only or hybrid edge-cloud processing is faster than cloud-only processing.

The huge amount of data collected by organizations gave rise to big data analytics where data is analyzed meaningfully. The most popular big data analytics programming model and framework are MapReduce [6] and Hadoop [3], respectively. MapReduce processes data in batches, with the focus on increasing the processing throughput. More recently, stream data processing gained traction, being enabled by frameworks such as Spark Streaming [25] and Google Cloud Dataflow [2]. In this paper, we present measurements of big data analytics running on Hadoop at the edge, on the cloud and in hybrid edge-cloud setups.

In the last decade, we have witnessed the performance improvement of low-power devices stimulated mainly by the development of the mobile sector. These low-power devices are almost exclusively placed at the edge of the network, in the form of mobile phones, smart devices, Internet of Things (IoT), among others. In concordance with this trend, we use two low-power devices to perform measurements at the edge. Nvidia Jetson TK1 [16] and Jetson TX1 [8] are low-power systems based on 32- and 64-bit ARM processors, respectively. These systems run a Linux-based operating system (OS) and are capable of executing complex programs, such as MapReduce applications.

In analyzing the cases where workloads benefit from executing at the edge or on hybrid edge-cloud setups in contrast to cloud-only computing, we make the following contributions:

- We provide a model to decide when edge-only or hybrid edge-cloud processing is faster than cloud-only processing. This model uses both workload parameters, representing the ratio of input and output data size, and system parameters assessing cloud speedup over the edge and the computation-to-communication ratio.
- Using measurements, we validate this model and analyze the performance of MapReduce on low-power edge devices and on Amazon cloud. Our analysis reveals a complex landscape where cloud-only processing is not always faster. The key factors affecting cloud-only processing are (i) link bandwidth used to upload the data collected at the edge and (ii) application characteristics.
- Using two types of MapReduce processing on hybrid edge-cloud systems, we draw useful insights on the effect of application characteristics on performance.

978-1-7281-2708-8/19/$31.00 ©2019 IEEE
DOI 10.1109/EDGE.2019.00029

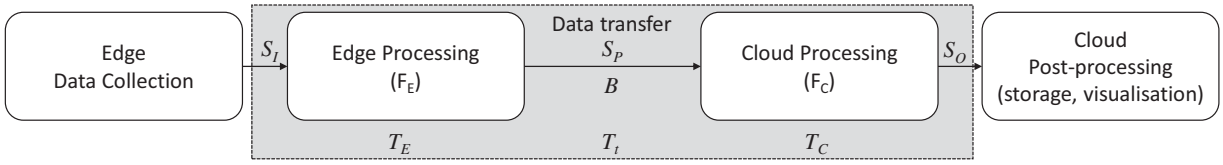


Fig. 1: Hybrid Edge-Cloud Processing

The rest of this paper is organized as follows. In the next section, we present a model to analyze the execution time of edge-cloud systems. In Section III, we analyze the execution time performance of edge-cloud systems running MapReduce applications using measurements. In Section IV, we survey the related work and in Section V we conclude the paper.

II. Model-driven Analysis

In this section, we analyze the edge-only, cloud-only and hybrid edge-cloud environments from a theoretical perspective, using analytic models for computation and communication.

A. Model

Firstly, we describe the model using the notations in Table I and the hybrid edge-cloud setup depicted in Figure 1. Data collection is always done at the edge and data post-processing, which may refer to storage, visualization or further processing, is always done on the cloud. In this paper, we are only interested in the main processing and edge-cloud data transfers. Our aim is to analyze the cases where edge-only or hybrid edge-cloud processing is faster than cloud-only computing. Hence, data collection and post-processing are excluded from our analysis.

We assume that a set of collected data of size S_I is processed at the edge by applying a processing function F_E. The partial results of size S_P are transferred to the cloud over a link with bandwidth B. On the cloud, these data are further processed by a function F_C to produce an output of size S_O. However, one may choose to perform the entire processing $(F_E + F_C)$ on the cloud or at the edge. Thus, we distinguish three cases:

1) *cloud-only processing* where the entire collected data is transferred to and processed on the cloud,

$$T_1 = T_t(S_I) + T'_C(S_I) \tag{1}$$

2) *edge-only processing* where the data is processed at the edge and only the final results are transferred to the cloud,

$$T_2 = T'_E(S_I) + T_t(S_O) \tag{2}$$

3) *hybrid edge-cloud processing* where the edge is performing the first part of the processing and the cloud is doing the second part. The execution time in this case is

$$T_3 = T_E(S_I) + T_t(S_P) + T_C(S_P) \tag{3}$$

Among the reasons for using cloud computing, scalability enables users to achieve faster execution compared to self-hosted, edge systems. In this paper, we assume that cloud computing is always faster than edge computing, or at least it achieves the same performance. Hence, for a given workload, cloud speedup is defined as

$$\alpha = \frac{T_E}{T_C} \geq 1 \tag{4}$$

In some edge computing setups, there may be multiple locations or clusters processing data at the same time. All these clusters send the results to a single cloud location to aggregate the data. Hence, we assume that the cloud processing or post-processing starts only after all n edge locations have transferred their data,

$$\begin{aligned} T_2 &= \max_{i=1}^{n}[T'_{E,i}(S_{I,i}) + T_{t,i}(S_{P,i})] \\ T_3 &= \max_{i=1}^{n}[T_{E,i}(S_{I,i}) + T_{t,i}(S_{P,i})] + T_C(\sum_{i=1}^{n} S_{P,i}) \end{aligned} \tag{5}$$

Without losing generality, we further consider a single edge location. For example, this is the location exhibiting the maximum execution time among all n edge locations.

Next, we define the input/output ratio of a given application, also called selectivity [9]. For edge- and cloud-only cases, the selectivity, σ, considers only the input and the final output. In the case of a hybrid setup, there are two selectivity values, of edge processing, σ_E, and cloud processing, σ_C, respectively. The following equations summarize selectivity,

$$\sigma = \frac{S_I}{S_O}, \sigma_E = \frac{S_I}{S_P}, \sigma_C = \frac{S_P}{S_O} \tag{6}$$

TABLE I: Notations

Symbol	Description
F_E	application running at the edge
F_C	application running on the cloud
T_E	processing time at the edge (hybrid processing)
T'_E	processing time at the edge (edge-only processing)
T_C	processing time on the cloud (hybrid processing)
T'_C	processing time on the cloud (cloud-only processing)
T_t	data transfer time
S_I	input size of a given application
S_O	output size of a given application
S_P	partial processing output size
χ	processing throughput
B	edge-cloud networking bandwidth
α	cloud processing speedup
σ	input/output ratio (selectivity) of a given application
γ	computation-to-communication ratio
η_E	edge-only speedup
η_H	hybrid edge-cloud speedup

The input or output data is transferred to the cloud in a period T_t over a link with bandwidth B. This communication time may dominate the entire execution, as we shall see in Section III. To quantify this, we define the ratio between computation and communication with respect to the cloud,

$$\gamma = \frac{T_C}{T_t} \tag{7}$$

If the bandwidth is constant, this parameter becomes the ratio between link bandwidth and cloud throughput when processing a given workload. Let this cloud throughput be defined as

$$\chi_C(S) = \frac{S}{T_C(S)} \tag{8}$$

Then, the computation-to-communication ratio[1] becomes

$$\gamma = \frac{B}{\chi_C(S_I)} \tag{9}$$

In summary, the edge-cloud execution time model depends on one workload parameter, σ, and two parameters, α and γ, representing the interaction between hardware devices and software applications. We observe that all three parameters are application-dependent.

B. Edge-only versus Cloud-only

We analyze the case where edge-only processing is faster compared to cloud-only processing, $T_2 \leq T_1$. For simplicity, we assume there is a single edge cluster. Using the equations presented in the previous section, this case is summarized as

$$T'_E(S_I) - T'_C(S_I) \leq T_t(S_I) - T_t(S_O) \tag{10}$$

We observe that a necessary condition is to transfer the edge processing output, S_O, faster than transferring the initial input data, S_I, to the cloud. If the bandwidth is constant, the condition is to have super-unitary selectivity, $\sigma \geq 1$. This means that the final output must be smaller than the input, $S_O \leq S_I$.

Next, we assume that the above condition is satisfied and, for ease of analysis, we consider the case where selectivity reflects into the transfer time,

$$\sigma = \frac{T_t(S_I)}{T_t(S_O)} \tag{11}$$

For example, this is true when the bandwidth is constant. Using Equation 10, we define edge-only speedup reported to cloud-only execution as

$$\eta_E = \frac{T_1}{T_2} = \frac{\gamma + 1}{\alpha\gamma + \frac{1}{\sigma}} \tag{12}$$

Hence, edge-only execution is faster when

$$\eta_E > 1 \Leftrightarrow \gamma(\alpha - 1) + \frac{1}{\sigma} < 1 \tag{13}$$

[1]Higher values of γ indicate that more time is spent in processing compared to data transfers. For example, let there be two applications that transfer the same amount of data to the cloud over a link of constant bandwidth. The application with lower cloud processing throughput has a higher computation-to-communication ratio.

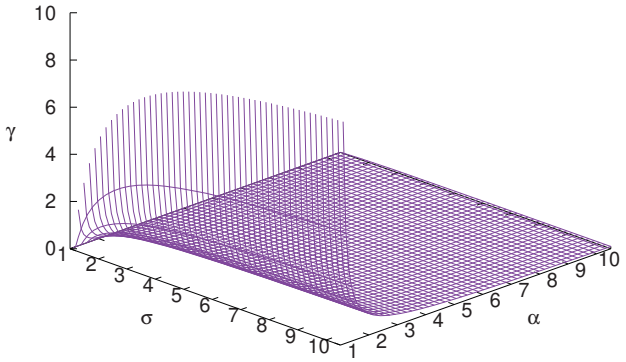


Fig. 2: γ as function of α and σ

Since the selectivity is supra-unitary, we observe that edge-only computing is faster than cloud-only computing when the actual processing exhibits similar execution time (cloud speedup, α, is low) and the transfer time dominates the processing time (γ is small). These cases are depicted by the points under the surface plotted in Figure 2.

C. Hybrid versus Cloud-only

Next, we analyze the case where hybrid processing is faster than cloud-only processing. The other case where edge processing is faster than hybrid processing is not considered in this paper since the main assumption is that data needs to be post-processed on the cloud, as shown in Figure 1. This is a real-world scenario for applications that need to aggregate data across multiple locations.

Using the equations defined in Section II-A, the scenario under analysis, $T_3 \leq T_1$, becomes

$$T_E(S_I) + T_C(S_P) - T'_C(S_I) \leq T_t(S_I) - T_t(S_P) \tag{14}$$

The partial processing output should be smaller compared to the input, $S_P < S_I$ ($\sigma_E > 1$), but is not uncommon to have the opposite situation where an application produces more intermediate data compared to its input, as we shall see in Section III. Using input and intermediate data sizes, and cloud computing as reference, we define the following computation-to-communication ratios,

$$\gamma_I = \frac{T_C(S_I)}{T_t(S_I)}, \gamma_P = \frac{T_C(S_P)}{T_t(S_P)}, \gamma'_I = \frac{T'_C(S_I)}{T_t(S_I)} \tag{15}$$

Using Equation 14, we define hybrid-edge speedup over cloud-only execution as

$$\eta_H = \frac{T_1}{T_3} = \frac{\sigma_E(\gamma'_I + 1)}{\sigma_E\alpha\gamma_I + \gamma_P + 1} \tag{16}$$

Hybrid computing is faster than cloud-only computing if

$$\eta_H > 1 \Leftrightarrow \sigma_E \frac{\gamma'_I - \alpha\gamma_I + 1}{\gamma_P + 1} > 1 \tag{17}$$

We observe that one can decide when hybrid computing is faster using cloud and application parameters, except for the cloud speedup which involves measuring edge execution time.

On the other hand, edge system designers can select the best edge hardware that achieves the required speedup to surpass cloud-only performance.

D. Discussion

In this section, we discuss how our model is applied by an organisation or application developer to explore edge computing and to select the best hardware platform for a set of given requirements. Starting from the applications currently running on the cloud, the developer has to assess if these applications can be executed in an edge-only or hybrid edge-cloud setup. In any case, the developer can compute the selectivity, σ, for each application, independent of the edge hardware platform. Moreover, the computation-to-communication ratio, γ, can be computed based on the cloud throughput for each application, which is already known, and the edge-cloud bandwidth which is partially dependent on the edge hardware. For example, if the edge devices have 100 Mbps Ethernet networking links, the upper bound for bandwidth would be 12.5 MB/s, which is hard to achieve in practice. Similar estimations can be done if wireless links are used.

The only parameter that needs to be measured on the edge hardware is the cloud speedup, α. However, one can use hardware specifications or existing literature to estimate the edge-cloud performance gap. For example, the developer uses a Jetson TX1 [8] as edge computing platform, while the cloud is running on Intel Xeon processors. Using benchmarking databases, such as Geekbench [1], the Xeon cloud speedup is estimated to be two. In the literature, the Xeon cloud speedup compared to Jetson TX1 when running Java applications reaches a value of three [13]. Indeed, our measurements in Section III expose a cloud speedup between 1.9 and 3.6, depending on the workload.

Next, the developer determines the selectivity of the applications that can run faster at the edge, when the cloud speedup is set, $\alpha = 3$. From Equation 13, the computation-to-communication ratio has to be $\gamma < \frac{1}{2}$. Let us set $\gamma = 0.2$. Then the selectivity range becomes $\sigma > 1.67$. We shall see in Section III that applications satisfying these conditions run faster at the edge or on hybrid edge-cloud setups that use TX1.

III. Measurements-driven Analysis

In this section, we analyze the trade-off between edge and cloud when processing data analytics using the well-known MapReduce [6] programming model and its open source implementation, Hadoop [3]. For this analysis, we select both (i) applications that can be executed in a hybrid edge-cloud setting and (ii) applications that are not suitable for hybrid execution. For example, MapReduce applications where either the Map or Reduce task forwards the data without any processing do not expose hybrid execution characteristics.

In our experiments, the hybrid execution consists of two distinct MapReduce jobs, one running at the edge and one running on the cloud. The edge job retains the Map function of the initial, non-hybrid, MapReduce application and has a Reduce function that forwards the data to the output. The cloud job retains the Reduce function of the initial MapReduce application and has a Map function that forwards the input to the next phase. Another possible setup consists of creating a MapReduce cluster comprising both edge and cloud nodes. However, such setup may incur more data transfer due to replication and housekeeping. We compare these two hybrid setups in Section III-E.

A. Applications

Seven MapReduce applications covering multiple domains are selected to run with large inputs, as shown in Table II. Pi Estimation (**PI**) generates random points in a unit square and determines the value of π based on how many points are inside the unit circle. Each Map function generates a point and evaluates if it is inside the circle. The Reduce task counts how many points are inside and outside the circle, and computes the final value for π. Kmeans (**KM**) groups a set of n points into k clusters based on the distance among the points. Starting from k centroids, Map task determines the closest centroid for each input point. The Reduce task computes the new centroids based on the points associated with each cluster during the Map phase. The process may be repeated for a number of times or until there is no change in the computed centroids. Word Count (**WC**) is the textbook example for MapReduce computation [6] which computes the number of appearances for each word in an input text file. The Map task splits a line of the input text into words and outputs each word as a key with an associated count of one. The Reduce task sums-up the counts for each word. Grep (**GR**) counts the number of lines from an input text containing a given regular expression. Map task filters the lines containing the regular expression and outputs their count. Reduce task sums-up all these counts. In our experiments, both WC and GR take as input a subset of Wikipedia's articles dump. The hybrid edge-cloud implementations of PI, KM, WC and GR apply the Map task at the edge and the Reduce task on the cloud.

BlackScholes (**BS**) determines the price of financial options based on some input parameters. Since the Reduce function of BS is forwarding, this application is not suitable for hybrid processing. Matrix Multiplication (**MM**) computes the product of two square matrices, A and B, of size n. In our implementation, each input line contains one row from matrix A and one column from matrix B. The Map task computes the elements of the result matrix and the Reduce task just builds this matrix. Similarity Score (**SS**) computes the similarity of n pairs of vectors with m elements. BS, MM and SS do not have a hybrid version since the Reduce task just forwards Map results to the output.

In summary, only four applications support hybrid execution, PI, KM, WC and GR. All our experiments are run at least three times and the average execution time is reported.

B. Systems

For this analysis, we use the popular Amazon Web Services (AWS) as representative for cloud computing. On the other hand, we select two low-power systems based on the emerging

TABLE II: Applications

Application	Hybrid?	Input Size [MB]	Map Output Size [MB]	Reduce Output Size [MB]	α_{TK1}	α_{TX1}	σ	σ_E	σ_C	γ_{low}	γ_{high}
Pi Estimation (**PI**)	Yes	189.7	143.1	0 (10 B)	3.1	2.7	19,888,888.5	1.3	15,000,000.0	3.9	114.6
Kmeans (**KM**)	Yes	19,801.6	20,199.2	0 (621 B)	3.9	1.9	33,435,490.0	1.0	34,106,891.0	0.03	1.03
Word Count (**WC**)	Yes	22,888.2	27,022.3	4,001.1	-	2.9	5.7	0.8	6.8	0.06	2.2
Grep (**GR**)	Yes	22,888.2	187.1	0 (13 B)	3.1	3.6	1,846,154,151.0	122.4	15,087,944.8	0.72	0.23
BlackScholes (**BS**)	No	24,802.3	4,942.2	4,942.2	5.1	3.0	5	5	1	0.07	0.54
Matrix Multiplication (**MM**)	No	25,780.4	26.9	26.9	2.2	3.6	959.6	959.6	1.0	4.7	0.2
Similarity Score (**SS**)	No	25,780.4	2.1	33.3	2.3	3.6	774.1	12014.5	0.1	58.7	0.2

TABLE III: Edge-only Speedup (η_E)

System	Application						
	PI	KM	WC	GR	BS	MM	SS
TK1, B_{low}	0.4	9.9	-	54.8	3.8	84.6	80.3
TX1, B_{low}	0.5	20.3	3.1	47.8	4.2	53.8	53.7
TK1, B_{high}	0.3	0.5	-	1.7	0.5	2.8	1.7
TX1, B_{high}	0.4	1.0	0.5	1.5	0.8	2.7	1.8

ARM architecture as representative of edge computing [13], [26]. Nvidia Jetson **TK1** is a 32-bit ARM system with a quad-core ARM Cortex-A15 CPU and 2GB RAM [16]. Nvidia Jetson **TX1** is a 64-bit ARM system with a quad-core ARM Cortex-A57 and 4GB RAM [8]. We added to these systems a 2 TB hard-disk (HDD). Both systems have Gigabit Ethernet networking cards.

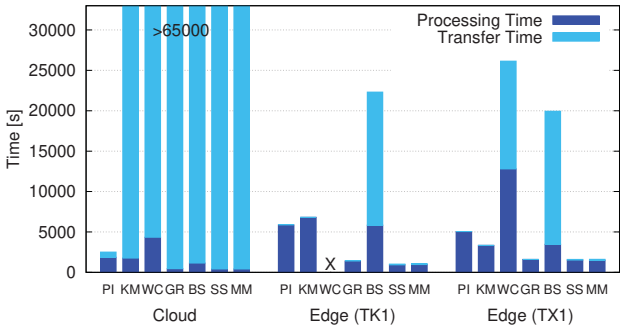
We measured the bandwidth between our local edge systems and the AWS cloud, as well as between all available regions of AWS. The average bandwidth is 11.4 Mbits/s (1.4 MB/s) with a standard deviation of 16.7. The minimum and maximum measured bandwidths are 2.33 Mbits/s ($B_{low} = 0.3MB/s$) and 92.8 Mbits/s ($B_{high} = 11.6MB/s$), respectively. These results show that edge-cloud and inter-cloud bandwidths are highly variable, while most fall on the low side of the spectrum. Throughout our analysis in the remainder of this paper, we use the minimum and maximum measured bandwidths, B_{low} and B_{high}, respectively.

C. Edge-only versus Cloud-only

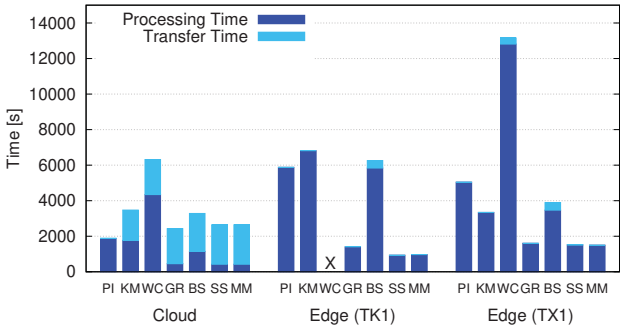
We compare edge-only with cloud-only processing using all seven MapReduce applications. The execution time breakdown showing processing and transfer times is plotted in Figure 3 for both low and high bandwidth. WC on Jetson TK1 does not finish execution due to memory and I/O failure, and it is marked accordingly in the plots.

Even if cloud processing is 2 to 5 times faster compared to the edge, as shown in Table II, the transfer time affects the overall performance of applications with large inputs. For example, only PI achieves better execution time on the cloud when the bandwidth is low because its input size is much smaller compared to the other six applications. On the other hand, the edge is faster than the cloud at processing SS, MM and GR even when the bandwidth is relatively high.

Using Equation 13, we determine the minimum bandwidth such that the cloud is faster than the edge when running all seven applications. We obtain 50.6 MB/s and 23.9 MB/s with TK1 and TX1 edge nodes, respectively. For example, the edge-cloud networking link needs more than twice the measured



(a) With Low Bandwidth (0.3 MB/s)



(b) With High Bandwidth (11.6 MB/s)

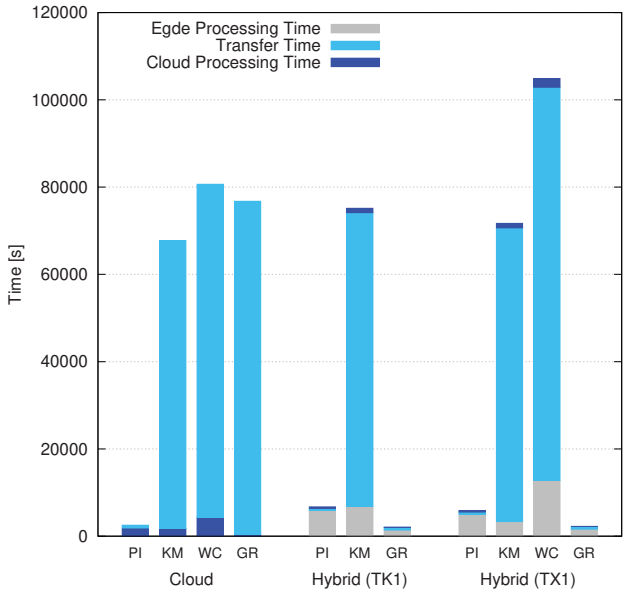
Fig. 3: Execution Time Comparison between Cloud- and Edge-only Processing

bandwidth of 11.6 MB/s for the cloud to be faster than TX1 edge. Thus, our approach enables better hardware-software co-design by identifying the system bottleneck.

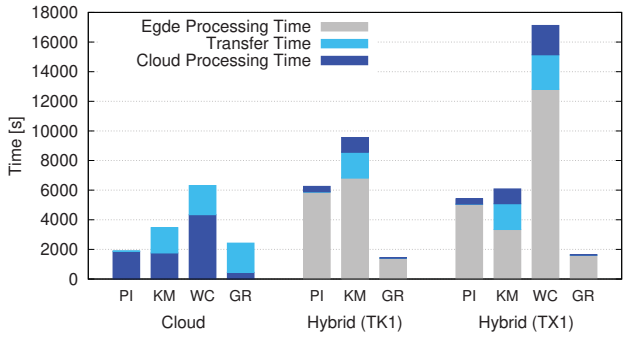
Next, we validate the analytic model in Equation 13 by computing the edge-only speedup. Values greater than one reveal that edge computing is faster than cloud computing. The results summarized in Table III show that modeled speedup is in concordance with the measurements plotted in Figure 3. For example, the edge-only speedup is 1.0 and 0.8 when the edge-cloud bandwidth is high and TX1 is running KM and BS, respectively. These values are close to one, revealing that edge-only and cloud-only processing achieve similar performance, as shown in Figure 3b.

D. Hybrid versus Cloud-only

We compare the performance of cloud-only and hybrid processing using the four MapReduce applications that support hybrid execution, as discussed in Section III-A. The execution



(a) With Low Bandwidth (0.3 MB/s)



(b) With High Bandwidth (11.6 MB/s)

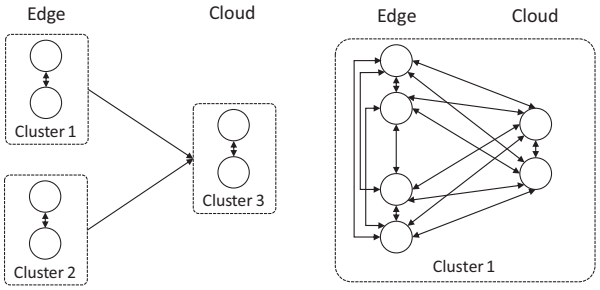
Fig. 4: Execution Time Comparison between Cloud-only and Hybrid Processing

time breakdown showing edge and cloud processing, and edge-cloud transfer time is plotted in Figure 4. Applications with low data movement, such as PI, are always faster on the cloud. On the other hand, applications with large inputs but small intermediate data size, such as GR, are always faster on hybrid setups. But applications with large intermediate output, such as KM and WC, suffer when the edge-cloud bandwidth is low, as shown in Figure 4a. Nonetheless, when the edge-cloud bandwidth is high, the cloud is faster due to its higher computing power, as shown in Figure 4b.

In analogy with our edge-cloud comparison, we validate the analytic model by computing the hybrid edge-cloud speedup. Values greater than one reveal that hybrid processing is faster than cloud-only processing. The results summarized in Table IV show that only GR exhibits super-unitary speedup, a fact reinforced by the measurements presented in Figure 4. On the other hand, KM speedup when the bandwidth is low

TABLE IV: Hybrid Edge-Cloud Speedup (η_H)

System	Application			
	PI	KM	WC	GR
TK1, B_{low}	0.4	0.9	-	37.6
TX1, B_{low}	0.5	1.0	0.8	34.2
TK1, B_{high}	0.3	0.4	-	1.7
TX1, B_{high}	0.3	0.6	0.4	1.5



(a) Separate clusters (b) Single hybrid cluster

Fig. 5: Types of Hybrid Edge-Cloud Setups

is close to one, revealing similar performance on cloud-only and hybrid setups. Hence, using our model, organizations can decide if it is beneficial to move their workloads on a hybrid edge-cloud setup.

E. Hybrid Edge-Cloud MapReduce

In this section, we analyze the performance of two types of hybrid edge-cloud MapReduce setups, as shown in Figure 5. The objective of this analysis is to determine application characteristics that impact performance in a hybrid edge-cloud setup, thus, aiding in developing further optimizations.

The first setup consists of creating separate MapReduce clusters at the edge and on the cloud, with explicit intermediate data transfers, as depicted in Figure 5a where each cluster has two nodes. Given an initial MapReduce application that can be executed on a hybrid edge-cloud setup, the developer has to write two new MapReduce applications, one for the edge and one for the cloud. The edge MapReduce application retains the Map function of the initial application and employs a forwarding Reduce function. The new MapReduce application running on the cloud retains the initial Reduce function and employs a forwarding Map function. The second setup consists of creating a single MapReduce cluster over both the edge and cloud, as shown in Figure 5b. This cluster runs the original MapReduce application, while all data transfers are handled by the Hadoop framework. However, Hadoop is also doing replication and housekeeping over the same network, thus, we expect a lower performance compared to the first setup.

We create these two setups using three AWS cloud regions, such that two of them simulate the edge and the third one represents the cloud. We choose to simulate the edge using cloud instances because we want to analyze the influence of transfer time and bandwidth rather than the effect of hardware on cloud speedup over the edge. By using the same type of nodes for the edge and cloud, we minimize the effect of cloud

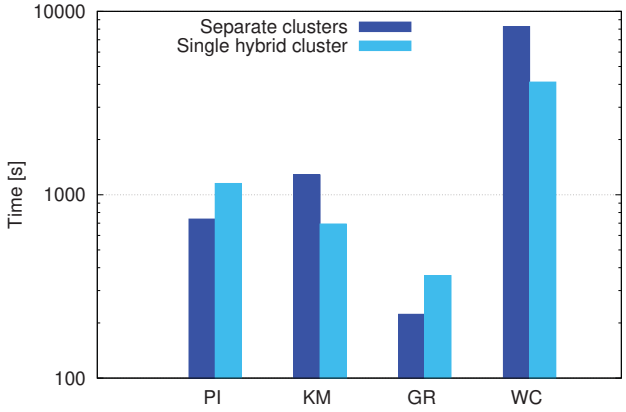


Fig. 6: Comparison between Hybrid Setups

speedup. Moreover, we use two regions for the edge because in real-world scenarios organizations have more than one edge cluster to aggregate the data from.

We run the four hybrid MapReduce applications on both hybrid setups and present the execution times in Figure 6. The reported values do not include loading the data into the Hadoop Distributed File System (HDFS). Including this loading time would increase the performance gap since the single edge-cloud hybrid cluster needs more time to upload and replicate the data.

The results expose balanced performance. The single hybrid MapReduce cluster is 41% and 63% slower than separate edge and cloud clusters for PI and GR, respectively. On the other hand, the single cluster is 85% and 100% faster when processing KM and WC, respectively. This is surprising, but it can be explained by the intra- and inter-cluster networking links shown in Figure 5. In the single cluster setup, there are peer-to-peer connections among all nodes, hence, the intermediate data is disseminated fast. Moreover, Hadoop tries to reduce the amount of shuffled data by running the Reduce tasks closer to data location. On the other hand, the separate cluster setup has a single link per edge cluster, with a maximum bandwidth of 11.6 MB/s. This setup can be improved by using more links between the edge and the cloud, if (i) such multiple links are available and (ii) the bandwidth increases with the number of links used. However, it is hard to achieve this in practice.

In summary, this section presents two MapReduce hybrid edge-cloud setups and analyzes their time-performance trade-offs. The results show that application characteristics and edge-cloud bandwidth have high impact on performance.

IV. Related Work

We classify the related work into (i) edge computing architectures and models and (ii) generic performance analysis of MapReduce big data analytics.

A. Edge Computing Architectures and Models

With the advent of edge and fog computing, multiple architectures have been proposed [4], [15], [18], [19], [20]. For example, the survey in [15] distinguishes fog computing, multi-access edge computing and cloudlet computing. Fog computing [4], [15] is tightly coupled with the cloud and operates somewhere between edge devices and cloud servers. Multi-access edge computing or mobile edge computing (MEC) focuses on mobile edge devices, while computing services are provided by servers placed in mobile base-stations, one hop away from the devices. Cloudlet computing [15], [18] refers to the virtualization technology used at the edge, one hop away from the mobile devices. Cloudlets can work together with the cloud, being distributed at the edge based on application demands. In comparison with these architectures, our hybrid-edge cloud scenario is closer to MEC since the edge processing is done one hop away from the devices that collect data. However, our setup requires cloud connection to aggregate the data, similar to fog computing as defined in [15]. Hence, in this paper we do not distinguish between edge and fog, but we prefer to use the term edge computing.

The performance of edge and fog platforms has been analyzed using both models and measurements. Sarkar and Misra [17] model fog and cloud computing with focus on latency and energy consumption. However, their models are not validated against measurements. Loghin et al. [13] analyze the time, energy and cost performance of low-power edge devices in comparison with powerful cloud instances. Similar to their approach, we use low-power devices as representative of the edge, but we focus on analyzing hybrid edge-cloud setups rather than comparing the processing capabilities of low-power edge and high-performance cloud. Integrating decision-making in the platform, CloneCloud [5] decides when part of a mobile application can be executed on the cloud to achieve better time and energy performance. CloneCloud uses static program analysis and dynamic profiling to build a model that enables decision-making. While CloneCloud depends on dynamic measurements, our approach is more flexible since it can use estimated parameters, as discussed in Section II-D.

With MapReduce being used extensively in the cloud, some works study edge-cloud MapReduce setups [7], [13], [21]. For example, Drolia et al. [7] compare traditional cloud computing with mobile edge computing when running image processing on MapReduce. Similar to us, they show that not all applications are suitable for running at the edge, especially when using low-bandwidth wireless connections between edge nodes. On the other hand, Tang et al. [21] use MapReduce only on the cloud in a smart city architecture with four layers, namely IoT devices, edge, fog and cloud.

B. MapReduce Performance Models and Measurements

Since there are not many studies involving MapReduce at the edge, we summarize the works analyzing MapReduce on traditional self-hosted or cloud clusters. MapReduce performance has been extensively analyzed and modeled [9], [10], [11], [12], [14], [22], [23]. Herodotou et al. [9], [10] propose a detailed execution model for Hadoop which is reinforced by fine-grain profiling. However, such detailed profiling may affect application execution and, thus, the model based on the

profiled parameters overestimates the execution time. Verma et al. [23], [24] model lower and upper bounds for Hadoop execution based on lighter workload profiling. Khan et al. [11] refine this model using advanced regression techniques to achieve very low modeling errors. In contrast to these works, we do not model MapReduce execution. Instead, we use default Hadoop monitoring to determine application selectivity, and Map and Reduce execution times. Loghin et al. [12], [14] use measurements to analyze the performance of MapReduce on emerging low-power devices based on ARM processors in comparison with high-performance servers typically found in cloud clusters. However, their work does not consider hybrid edge-cloud MapReduce processing.

V. Conclusion

In this paper, we analyze and compare the performance of edge-only and hybrid edge-cloud processing with cloud-only processing. Given an application, we propose an analytic model based on three key parameters (i) application selectivity representing the ratio between input and output size, (ii) cloud speedup compared to edge processing and (iii) computation-to-communication ratio with respect to uploading and processing the input on the cloud. While our model can be used with parameters taken from the target system specifications or from the existing literature, higher accuracy is achieved through measurements on target edge and cloud systems.

We conducted measurements to validate our models and to further analyze the performance of edge and cloud computing. The measurements are done with seven representative MapReduce applications on two low-power edge devices and on AWS cloud. Firstly, we show that not all MapReduce applications are suitable for hybrid edge-cloud processing. For applications that are suitable for hybrid execution, we analyze two MapReduce setups consisting of both (i) separate edge and cloud clusters and (ii) a single edge-cloud cluster. Secondly, we reveal a complex landscape where there is no clear winner among hybrid, edge-only and cloud-only processing. Application characteristics, such as selectivity, and edge-cloud bandwidth are the key factors that affect the performance.

Acknowledgment

This work was supported in part by Singapore Ministry of Education through Academic Research Fund Tier 1 and Academic Research Fund Tier 3. We would like to thank Distinguished Professor Beng Chin Ooi for his insightful feedback that helped improve our work. We also thank Nvidia for providing hardware grants consisting of Jetson TK1 and Jetson TX1 systems, and Amazon for providing Elastic Compute Cloud credits.

References

[1] Geekbench Browser, https://bit.ly/2R4kYML, 2019.

[2] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. Fernández-Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt, S. Whittle, The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-scale, Unbounded, Out-of-order Data Processing, *PVLDB*, 8(12):1792–1803, 2015.

[3] Apache, Hadoop, http://tinyurl.com/5f4ojf, 2018.

[4] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog Computing and Its Role in the Internet of Things, *Proc. of 1st Edition of the MCC Workshop on Mobile Cloud Computing*, pages 13–16, 2012.

[5] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, Clonecloud: Elastic Execution between Mobile Device and Cloud, *Proc. of 6th Conference on Computer Systems*, pages 301–314, 2011.

[6] J. Dean, S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, *Proc. of 6th Conference on Symposium on Opearting Systems Design & Implementation*, pages 10–10, 2004.

[7] U. Drolia, R. Martins, J. Tan, A. Chheda, M. Sanghavi, R. Gandhi, P. Narasimhan, The Case for Mobile Edge-Clouds, *Proc. of IEEE 10th International Conference on Ubiquitous Intelligence and Computing and IEEE 10th International Conference on Autonomic and Trusted Computing*, pages 209–215, 2013.

[8] D. Franklin, Nvidia Jetson TX1 Supercomputer-on-Module Drives Next Wave of Autonomous Machines, http://www.webcitation.org/6qK3fusRx, 2015.

[9] H. Herodotou, S. Babu, Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs, *PVLDB*, 4(11):1111–1122, 2011.

[10] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, S. Babu, Starfish: A Self-tuning System for Big Data Analytics, *Proc. of 5th Biennial Conference on Innovative Data Systems Research*, pages 261–272, 2011.

[11] M. Khan, Y. Jin, M. Li, Y. Xiang, C. Jiang, Hadoop Performance Modeling for Job Estimation and Resource Provisioning, *IEEE Transactions on Parallel and Distributed Systems*, 27(2):441–454, 2016.

[12] D. Loghin, L. Ramapantulu, O. Barbu, Y. M. Teo, A Time-energy Performance Analysis of MapReduce on Heterogeneous Systems with GPUs, *Performance Evaluation*, 91(C):255–269, 2015.

[13] D. Loghin, L. Ramapantulu, Y. M. Teo, On Understanding Time, Energy and Cost Performance of Wimpy Heterogeneous Systems for Edge Computing, *Proc. of 2017 IEEE International Conference on Edge Computing (EDGE)*, pages 1–8, 2017.

[14] D. Loghin, B. M. Tudor, H. Zhang, B. C. Ooi, Y. M. Teo, A Performance Study of Big Data on Small Nodes, *PVLDB*, 8(7):762–773, 2014.

[15] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, P. A. Polakos, A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges, *IEEE Communications Surveys Tutorials*, 20(1):416–464, 2018.

[16] Nvidia, Nvidia Unveils First Mobile Supercomputer for Embedded Systems, http://www.webcitation.org/6VdkUISQn, 2014.

[17] S. Sarkar, S. Misra, Theoretical Modelling of Fog Computing: a Green Computing Paradigm to Support IoT Applications, *IET Networks*, 5(2):23–29, 2016.

[18] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The Case for VM-Based Cloudlets in Mobile Computing, *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[19] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge Computing: Vision and Challenges, *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[20] W. Shi, S. Dustdar, The Promise of Edge Computing, *Computer*, 49(5):78–81, 2016.

[21] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, Q. Yang, Incorporating Intelligence in Fog Computing for Big Data Analysis in Smart Cities, *IEEE Transactions on Industrial Informatics*, 13(5):2140–2150, 2017.

[22] F. Tian, K. Chen, Towards Optimal Resource Provisioning for Running MapReduce Programs in Public Clouds, *Proc. of 4th International Conference on Cloud Computing*, pages 155–162, 2011.

[23] A. Verma, L. Cherkasova, R. H. Campbell, Resource Provisioning Framework for MapReduce Jobs with Performance Goals, *Proc. of 12th International Middleware Conference*, pages 160–179, 2011.

[24] A. Verma, L. Cherkasovab, R. H. Campbellc, Profiling and Evaluating Hardware Choices for MapReduce Environments: An Application-aware Approach, *Performance Evaluation*, 79:328–344, 2014.

[25] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, I. Stoica, Discretized Streams: Fault-tolerant Streaming Computation at Scale, *Proc. of 24th ACM Symposium on Operating Systems Principles*, pages 423–438, 2013.

[26] D. Zhang, Y. Ma, Y. Zhang, S. Lin, X. S. Hu, D. Wang, A Real-Time and Non-Cooperative Task Allocation Framework for Social Sensing Applications in Edge Computing Systems, *Proc. of 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 316–326, 2018.