A Benchmark to Evaluate Mobile Video Upload to Cloud Infrastructures

Afsin Akdogan(✉), Hien To, Seon Ho Kim, and Cyrus Shahabi

Integrated Media Systems Center, University of Southern California,
Los Angeles, CA, USA
{aakdogan, hto, seonkim, shahabi}@usc.edu

Abstract. The number of mobile devices (e.g., smartphones, tablets, wearable devices) is rapidly growing. In line with this trend, a massive amount of mobile videos with metadata (e.g., geospatial properties), which are captured using the sensors available on these devices, are being collected. Clearly, a computing infrastructure is needed to store and manage this ever-growing large-scale video dataset with its structured data. Meanwhile, cloud computing service providers such as Amazon, Google and Microsoft allow users to lease computing resources with varying combinations of computing resources such as disk, network and CPU capacities. To effectively use these emerging cloud platforms in support of mobile video applications, the application workflow and resources required at each stage must be clearly defined. In this paper, we deploy a mobile video application (dubbed *MediaQ*), which manages a large amount of user-generated mobile videos, to Amazon EC2. We define a typical video upload workflow consisting of three phases: (1) video transmission and archival, (2) metadata insertion to database, and (3) video transcoding. While this workflow has a heterogeneous load profile, we introduce a single metric, frames-per-second, for video upload benchmarking and evaluation purposes on various cloud server types. This single metric enables us to quantitatively compare main system resources (disk, CPU, and network) with each other towards selecting the right server types on cloud infrastructure for this workflow.

Keywords: Mobile video systems · Spatial databases · Cloud computing · Big video data · Benchmarking

1 Introduction

With the recent advances in video technologies and mobile devices (e.g., smartphones, tablets, wearable devices), massive amounts of user generated mobile videos are being collected and stored. According to Cisco's forecast [7], there will be over 10 billion mobile devices by 2018 and 54 % of them will be *smart* devices, up from 21 % in 2013. Accordingly, mobile video will increase 14-fold between 2013 and 2018, accounting for 69 % of total mobile data traffic by the end of the forecasted period. Clearly, this vast amount of data brings a major scalability problem in any computing infrastructure. On the other hand, cloud computing provides flexible resource arrangements that can instantaneously scale up and down to accommodate varying workloads. It is projected that the total economic impact of cloud technology could be

J. Zhan et al. (Eds.): BPOE 2014, LNCS 8807, pp. 57–70, 2014.
DOI: 10.1007/978-3-319-13021-7_5

$1.7 trillion to $6.2 trillion annually in 2025 [8]. Thus, the large IT service providers such as Amazon, Google, and Microsoft, are ramping up cloud infrastructures.

One key question is how to evaluate the performance of mobile video applications on these cloud infrastructures and select the appropriate set of resources for a given application. Suppose a mobile user wants to upload a video to a cloud server along with its metadata (e.g., geospatial properties of video such as camera location and viewing direction), which are captured and extracted using the sensors embedded on the mobile devices. Note that this kind of geospatial metadata enables advanced data management, especially in very large-scale video repositories. For example, the performance of a spatial query such as a range query, which can find all video frames that overlap with a user-specified region [2], can be significantly enhanced using spatial metadata. When we upload captured videos with metadata from mobile device to cloud, this upload operation consists of three stages which require different computing system resources: (1) *network* to transfer videos from mobile clients to the cloud servers (i.e., network bandwidth), (2) *database* to insert metadata about the uploaded videos (i.e., database transaction), and (3) *video transcoding* to change the resolution of uploaded videos to use less storage and bandwidth (i.e., CPU processing power). These phases are executed in sequence; therefore, inefficiency in any step slows down the performance of video applications. A benchmark to evaluate such an application needs to identify the system resources used at each stage, compare them with one another *quantitatively* and spot which resource(s) becomes the *bottleneck* in the workflow of the application. Once the bottlenecks are detected, the servers with the right specifications can be selected and configured accordingly on cloud.

There exists a challenge in evaluating the performance of a large scale video application on cloud because most of the benchmarking studies in the cloud computing context focus on evaluating either the performance of Big Data processing frameworks such as Hadoop and Hive [25, 26] or NoSQL data-stores rather than considering all system resources a mobile video application requires. In particular, some benchmarks are designed for social networking applications [17], online transaction processing (OLTP) [9, 10, 19] and simple key-value based put-get operations [16, 18]. These benchmarks only emphasize the impact of the database system on the overall performance. In addition, a recent study measures the impact of virtualization on the networking performance in the data centers [6]. However, this study only measures packet delays and TCP/UDP throughput, and packet loss among virtual machines.

In this paper, we define a single (cross-resource) metric to evaluate the uploading workflow of video applications on cloud and present an end-to-end benchmark. In particular, we use a throughput, the number of *processed frames per second*, as the metric and compare the performance of system resources (e.g., network, disk, CPU) with one another. To this extent, we deployed one exemplary mobile video application called MediaQ, which we developed on the Amazon EC2 platform, and conducted extensive experiments on various server types. Specifically, we used the smallest and the largest instance at each server group (e.g., *disk-optimized, CPU-optimized, general-purpose*) to identify the *lower and upper performance bound*. Our experimental results show that CPU drastically slows down the entire system and becomes the bottleneck in the overall performance. Our experiments also show that simply selecting high-end CPU-optimized servers does not resolve the problem entirely. Therefore, we propose

two techniques to enhance the CPU throughput: (1) reducing video quality and (2) enabling multithreading. Our study serves as the first step towards understanding the end-to-end performance characteristics of cloud resources in terms of resource-demanding video applications.

The remainder of this paper is organized as follows. Section 2 provides the necessary background. The benchmark design and experimental results are presented in Sects. 3 and 4, respectively. Related work is discussed in Sect. 5. Subsequently Sect. 6 concludes the paper with the directions for future work.

2 Background

Before we present our results and findings, we briefly introduce a typical example of resource intensive mobile video application (MediaQ) and available server types in the data centers to prepare for the rest of the discussion.

2.1 MediaQ: Mobile Multimedia Management System

MediaQ [2, 3] is an online media management framework that includes functions to collect, organize, search, and share user-generated mobile videos using automatically tagged geospatial metadata. MediaQ consists of a MediaQ server and a mobile app for smartphones and tablets using iOS and Android. User-generated-videos (UGV) can be uploaded to the MediaQ server from users' smartphones and they are then displayed accurately on a map interface according to their automatically collected geo-tags and other metadata information such as the recorded real time, camera location, and the specific direction the camera was pointing. Media content can be collected in a casual or on-demand manner. Logged in participants can post specific content requests that will automatically generate an alert with other participants who are near a desired content assignment location to entice them to record using their phones.

The schematic design of the MediaQ system is summarized in Fig. 1. Client-side components are for user interaction, i.e., the Mobile App and the Web App. The Mobile App is mainly for video capturing with sensed metadata and their uploading. The Web App allows searching the videos and issuing spatial crowdsourcing task requests to collect specific videos. Server-side components consist of Web Services, Video Processing, GeoCrowd Engine, Query Processing, Account Management, and Data Store. The Web Service is the interface between client-side and server-side components. The Video Processing component performs transcoding of uploaded videos so that they can be served in various players. At the same time, uploaded videos are analyzed by the visual analytics module to extract extra information about their content such as the number of people in a scene. We can plug in open source visual analytics algorithms here to achieve more advanced analyses such as face recognition among a small group of people such as a user's family or friends. Automatic keyword tagging is also performed at this stage in parallel to reduce the latency delay at the server. Metadata (captured sensor data, extracted keywords, and results from visual analytics) are stored separately from uploaded media content within the Data Store.

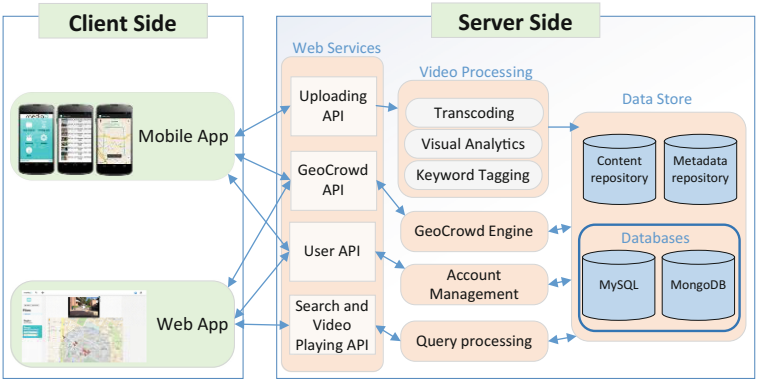


Fig. 1. Overall structure of MediaQ system

Query Processing supports effective searching for video content using the metadata in the database. Finally, task management for spatial crowdsourcing can be performed via the GeoCrowd engine.

2.2 Cloud Server Type Classification

Recently the computing resources on cloud have become more granular since service providers use virtualization techniques to manage physical servers and provide a wide selection of server types optimized to fit different use cases [1]. These types comprise varying combinations of CPU, memory, storage, and networking capacity and give users the flexibility to choose an appropriate combination of resources. Specifically, server types are clustered into six groups where each group consists of several options with varying computational capabilities. Table 1 depicts a classification of the server groups presently offered by the biggest three service providers along with the prices (dollars/hour) of the smallest and the largest server at each group. As shown, the pricing varies widely across

Table 1. Categorization of the server types with the prices (dollars/hour) of the smallest and the largest servers at each group.

Type	Amazon EC2		Microsoft Azure		Google Compute	
	Price ($/hour)		Price ($/hour)		Price ($/hour)	
	Smallest	Largest	Smallest	Largest	Smallest	Largest
General purpose (*m*)	0.07	0.56	0.02	0.72	0.077	1.232
Compute optimized (*c*)	0.105	1.68	2.45	4.9	0.096	0.768
Memory optimized (*r*)	0.175	2.8	0.33	1.32	0.18	1.44
Disk optimized (*i*)	0.853	6.82	–	–	–	–
Micro (*t*)	0.02	0.044	–	–	0.014	0.0385
GPU	0.65	0.65	–	–	–	–

the server types within each service provider. For example, the most expensive machine in Microsoft Azure is *245* times more costly than the cheapest one (*4.9*/*0.02*). This ratio is *97* for Amazon EC2 and *102* for Google Compute Engine, respectively. Clearly, such a huge discrepancy across the server types makes the selection of appropriate set of resources critical in hosting an application on these cloud platforms.

3 Benchmark Design

In this section, we first explain our measurement methodology and then discuss the metric we used in the experimental evaluation.

3.1 Methodology

There are three main components in the performance evaluation of large scale mobile video systems such as MediaQ which requires different system resources: (1) **network** to transfer videos from mobile clients to a cloud server (i.e., network bandwidth), (2) **database** to insert metadata about the uploaded videos (i.e., database transaction), and (3) **video transcoding** to change the resolution of uploaded videos which is a common operation in video services (i.e., CPU processing power). Specifically, we measure the *upload performance* which involves these three phases that are executed in sequence. Upon recording a video, mobile clients retrieve metadata (i.e., GPS signals, field of views, etc.) from the video. Subsequently, along with the video data, they upload the metadata in JSON format to the server. Once a video is uploaded, the metadata is inserted into the database and the video is transcoded, which is required to either support different formats (e.g., MP4, WAV) or to reduce video quality due to limited network bandwidth when being displayed later. Therefore, the videos are not retrievable until transcoding task is completed, and hence overhead in any component can degrade the overall performance of video applications. Our goal is to define a single metric, examine these components individually using this metric, and detect which phase slows down the system. To this extent, we deployed MediaQ server side code on the EC2 servers running a video upload service implemented in PHP. The service can receive multiple video files simultaneously. We then run multiple clients which transfer large amount of videos concurrently using the upload service.

3.2 Metric

We introduce a single metric, *processed-frames-per-second*, to evaluate the performance of three main components. For network performance, we straightforwardly report the number of transferred frames per second. For database performance, we report the number of frames inserted per second. Note that we do not insert the video data but its spatio-temporal metadata to the database. The metadata are collected at video capturing time by mobile devices and transferred to the server, thus the database cost is only composed of inserting a set of metadata (i.e., per frame) from memory into database. Similar to the standalone version of MediaQ, we selected MySQL database

Table 2. Hardware specifications of the smallest and largest servers of 4 server types on EC2.

Type	Memory	CPU	Disk	Network bandwidth
m-small	3.75 GB	1 VCPU	4 GB SSD	No info.
c-small	3.75 GB	2 VCPUs	32 GB SSD	No info.
r-small	15.25 GB	2 VCPUs	32 GB SSD	No info.
i-small	30.5 GB	4 vCPUs	800 GB SSD	No info.
m-large	30 GB	8 VCPU	160 GB SSD	No info.
c- large	60 GB	32 VCPUs	640 GB SSD	No info.
r- large	244 GB	32 VCPUs	2 × 320 GB SSD	10 Gigabit Ethernet
i- large	244 GB	32 vCPUs	8 × 800 GB SSD	10 Gigabit Ethernet

installed on EC2 servers. For transcoding performance, we use the number of transcoded frames per second. Once a video arrives at the server, MediaQ transcodes it using FFMPEG [13], which is a widely used video solution. In order to measure the maximum throughput, we perform a stress test on the cloud server by generating a large amount of real videos and uploading them to the server simultaneously and continuously for a significant amount of time.

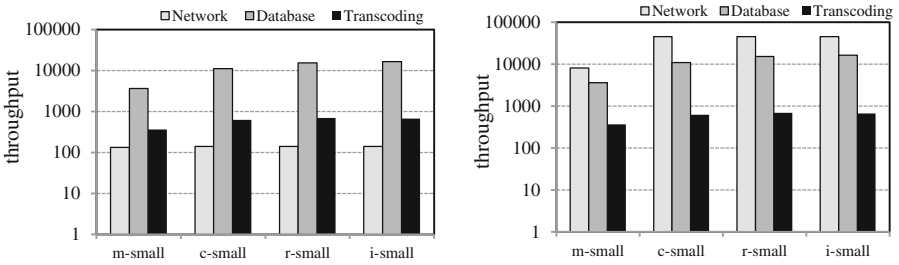
4 Performance Evaluation

In this section, we first present an overall cost analysis of the three components in the workflow and show how the server types impact the performance. Subsequently, we evaluate transcoding and database components in more detail, and finally present performance-cost results.

4.1 Overall Cost Analysis

In this set of experiments, we used the smallest servers on Amazon EC2 in four instance families: *general purpose (m), compute-optimized (c), memory-optimized (r) and disk-optimized (i)* and measured the throughputs (See Table 2 for hardware specifications of Amazon EC2). To fully utilize multi-core CPUs available at the servers, we enabled multi-threading on database and transcoding parts. Specifically, we first run the experiments using one thread ($T = 1$), and then increase the number of threads T by one to run the experiment again. The point where throughput cannot be improved further is the maximum throughput that the server can achieve. Note that there is no index built on the metadata table in the database and we take advantage of bulk insert, where 1,000 rows are written into disk as one transaction which reduces the disk I/O significantly. For transcoding tasks, we reduce the video resolution from 960 × 540 to 480 × 270.

Figure 2a illustrates the throughput comparison where a single large video with 24 fps (frame per second) was uploaded to the server. We observed that other than general purpose instance, the performance difference between the optimized servers (*c*, *r*, *i*) is not significant even though the prices vary widely such that *i-small* is *8* times



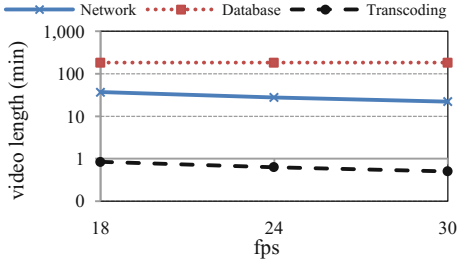
a) single video **b)** multiple videos

Fig. 2. Comparison of system components on smallest servers of 4 server types (log-scale).

more expensive than *c-small*. As shown, database can handle almost *two* and *three orders of magnitude* more frames than network and transcoding, respectively.

Fully utilizing the network bandwidth, real systems can handle concurrent video uploads to the server; therefore, in the next experiments, we did a stress test where multiple videos were uploaded simultaneously until the network bandwidth was saturated. As illustrated in Fig. 2b, network throughput increases significantly; however, database and transcoding remain almost constant. This is because we already enabled bulk insert and multi-threading to ensure the maximum performance even in the case of a single video upload. Another observation is that *transcoding*, which shows the lowest throughput, becomes a major *bottleneck* in the workflow.

The frames per second (*fps*) value in video recording has a direct impact on the performance in our experiments. However, database throughput is independent of *fps* in our target application MediaQ. This is because, *fps* value ranges from 15 to 120 in new generation smartphone cameras; however, regardless of *fps*, we select one metadata per each second using a sampling technique [3] and store it for all the frames in the corresponding second. This is a real-world phenomenon since metadata includes geospatial attributes such as the camera location and viewing direction which do not change significantly within a second. This approach widens the gap between the throughput of database and those of other components even further. Figure 3a depicts a comparison of the system resources under various *fps* values on a c-small instance,



a) Impact of fps (log-scale)

fps	Network	Database	Transcoding
24	- 25.1%	%0	- 23.3%
30	- 39.8%	%0	- 38.3%

b) Decrease over fps=18

Fig. 3. The video length (min) that each component can process in a second for various *fps*.

where the metric is total length of videos (minute) that each component can process within a unit time (second). For this experiment, we only changed the fps values [14] of videos and kept the original resolution. As shown, database throughput remains constant as *fps* increases while others diminish and the percentage of decrease over fps = 18 is listed in Fig. 3b. For higher frame rates, the length of video that network can handle decreases since the size of the videos grow and saturate the fixed bandwidth capacity. Similarly, transcoding can process a shorter amount of video per second as *fps* increases since its throughput on a specific server is fixed. In conclusion, these preliminary experiments verify that transcoding slows down the workflow dramatically; therefore, in the following section we propose several approaches to enhance this piece and measure the impact of each proposed technique.

4.2 Transcoding Performance

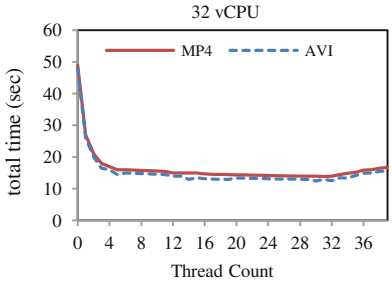
It is crucial to improve the transcoding performance because newly uploaded videos are not retrievable for use until their transcoding tasks are completed. That is the main reason behind the delay between the uploading and viewing time in many video-based applications. There are two ways to make transcoding faster: (1) enabling multi-threading, and (2) reducing the size of the output file, which results in a lower video quality. We explain these two approaches in turn.

Multi-threading. One natural way to improve the performance is utilizing multi-core CPUs available in the servers and *scale-up*. There are two techniques to increase the throughput on cloud. First, running a multi-threaded *ffmpeg* process (*MT*) on a single video and decrease the total amount of time to transcode it. Second, running a single-threaded *ffmpeg* processes in parallel on multiple videos (*PST*). In the following set of experiments, we use the largest *compute-optimized* (c3.xlarge) server with 32 vCPU's.

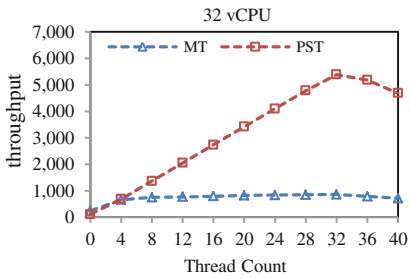
Figure 4a illustrates the effect of varying number of threads while transcoding a video. In this specific experiment, we used a 230 MB video in *AVI* format as input and reduced the resolution from 960 × 540 to 480 × 270 in two different video output types, MP4 and AVI. As shown, the total time does not decrease linearly as the number of threads increases. As stated in Amdahl's Law [6], a parallel algorithm is as fast as its sequential, non-parallelizable portion which dominates the total execution time. For *ffmpeg*, after *4 threads* the performance gain becomes insignificant no matter how many CPUs are used, which verifies that ffmpeg does not scale up.

Another way to increase throughput is running single-thread *ffmpeg* processes in parallel where each thread handles a single video. Figure 4b depicts the throughput performance of these two techniques. Since *ffmpeg* does not scale well as the number of threads increases, the throughput remains almost constant. However, throughput increases almost linearly for *PST* until all 32 CPUs are fully utilized. That is because while *MT* technique suffers from low parallelism, *PST* can utilize available CPUs better. After the CPUs are saturated, the performance goes down for both *MT* and *PST* due to resource contention across the threads.

Reducing Video Quality. In this set of experiment, we investigate the impact of *resolution* and *type* of the outputted video on the performance. Table 3 presents the



a) multi-thread ffmpeg on the same video for different video output types



b) multi-thread (MT) vs. parallel single-thread (PST) ffmpegs

Fig. 4. Scale up performance of transcoding.

transcoding throughputs for converting a video with dimensions 960 × 540 to smaller resolutions. The percentage of increase in the throughput is listed as well for clarity of the presentation. As expected, the result shows that throughput increases significantly as the resolution decreases. However, the percentage improvement diminishes when the output video resolution becomes too smaller (i.e., 60 × 32) because loading the input video, frame by frame, is a constant cost which largely contributes to the total transcoding cost. In addition, we also observe that the results are similar for different output formats (*mp4* and *avi*).

Table 3. Transcoding throughput for mp4 and avi types with various output resolutions. The input video is in.m4v format with 960 × 540 resolution.

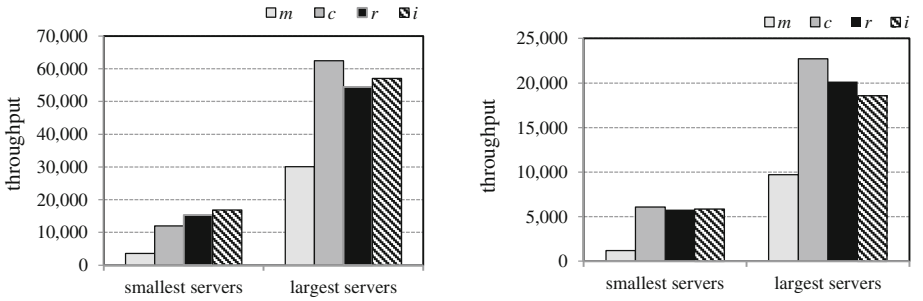
Output resolution	MP4		AVI	
	Throughput	% improvement	Throughput	% improvement
480 × 270	623	–	626	–
240 × 136	842	35 %	839	34 %
120 × 68	980	57 %	982	57 %
60 × 34	1038	66 %	1048	67 %

4.3 Database Performance

In this section, we measure the database throughput on both the smallest and largest instances at each server group to show the lower and upper *performance bounds*. In addition, we investigate the impact of indexing on throughput and test if it changes the best performer server. Throughput is measured using iterative multi-threading approach. First, we run the experiment with a single thread and repeat the experiment increase the number of threads by one until no throughput improvement is observed. Then, we report the maximum throughput as the result.

Metadata information is stored in *video_metadata* database table which consists of *13* columns where average length of a row is *319* bytes. Figure 5a and b illustrate the

effect of server types on the database throughput where the smallest and largest instances in general purpose (*m*), compute-optimized (*c*), memory-optimized (*r*) and disk-optimized (*i*) are clustered together. As shown in Fig. 5a, where there is no index on *video_metadata* table, the smallest disk-optimized server (*i*) slightly outperforms others. With the largest instances, compute-optimized (*c*) server provides slightly better performance than others. This is because while small servers contain 2 to 4 CPUs, large ones have 8 to 32 CPUs and compute-optimized machines might be better in managing *concurrent threads*. Note that, even though metadata insertion is an I/O-intensive task, disk-optimized machines do not expressively outperform other instances. The reason is that video data is mostly *append-only*, where the updates to the dataset are rare after the insertion. Disk-optimized instances are tuned to provide fast *random I/O*; however, in *append-only* datasets random access is not much used. Another observation is that optimized machines perform at least *2.5* times better than the general purpose one.



a) no index

b) with index

Fig. 5. Database insertion throughput on the smallest and largest servers in 4 instance families.

Effect of Indexing. To investigate how indexing influences throughput at each server, we built 2 indices on *video_metadata* table. Specifically, a B-tree index on the time field and hash index on the keywords fields, which is a good indexing strategy that allows efficient range search over the time and effective equality search on the keywords associated with the videos. As depicted in Fig. 5b, for both smallest and largest server groups, *compute-optimized* instances show better performance unlike the no-index scenario. The reason is that indices are kept in memory and index update is a CPU intensive task. Another observation is that indexing degrades the performance considerably, where throughput approximately drops to *1/3* of the no-index scenario due to extra high index maintenance cost.

4.4 Performance-Cost Analysis

In this section, we discuss how the performance-to-price ratios of different server types. Figure 6 illustrates the number of frames per dollar that each component can process using the smallest servers in each server group. In this specific example, we uploaded

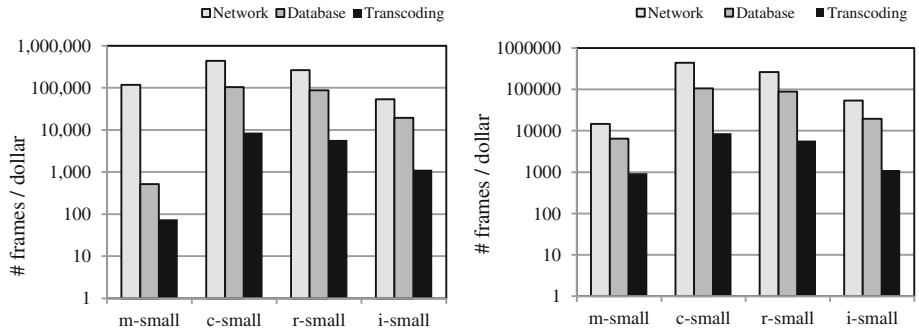


Fig. 6. Number of frames that can be processed for each dollar spent (log-scale).

multiple videos with 24 fps to the servers and enabled multithreading mode both on database insertion and transcoding. Note that performance-price ratio is sensitive to the total amount of data we upload since more nodes need to be employed when the storage capacity of a single server is exceeded. In Fig. 6a, we present the result for 4 GB of data which can fit all server types (See Table 2 for server specifications). As shown in Fig. 6a, compute-optimized server (*c-small*) outperforms other server types in all aspects. In addition, disk-optimized server (*i-small*) is not cost efficient for video uploads since there is not much need for random disk access as discussed in the previous section and the data size is small. In Fig. 6b, we present the results for 32 GB of data which 8 times exceeds the m-small node. In this scenario, we need allocate 8 small servers; therefore, cost efficiency dramatically drops for m-small server.

5 Related Work

With the increasing popularity of Big Data processing frameworks, several benchmarks have been proposed to evaluate various offline operations (e.g., grep, sort, aggregation, etc.) on popular frameworks such as Hadoop and Hive [25, 26]. Meanwhile, a number of benchmarks have been developed to measure the scalability of NoSQL and NewSQL databases. These benchmarks only emphasize the impact of the database system on the overall performance rather than considering all the resources an end-to-end mobile video application requires. In particular, some benchmarks are designed for social networking applications [17], online transaction processing (OLTP) [9, 10, 19] and simple key-value based put-get operations, which are heavily used in web applications [16, 18]. In addition, there are a few recent studies that measure the impact of virtualization on the networking performance in the data centers [6, 20]. However, these studies only measure packet delays and TCP/UDP throughput, and packet loss among virtual machines. Similar to our approach, CloudCmp [21] measures the performance of elastic computing and persistent storage services offered by cloud service providers. However, CloudCmp separates computing and storage instances, and employs different metrics for performance evaluation and cross-platform comparisons.

Moreover, while we focus on multimedia applications CloudCmp covers a wide range of web applications where the workloads are composed of put and get requests. In addition, a few measurement techniques have been studied to assess the energy consumption of cloud platforms [12].

In the multimedia context, several benchmarking approaches have been proposed as well. ALPBench [24] focuses on multi-core architecture, and measures the thread and instruction-level parallelism of complex media applications such as speech and face recognition. Also traditional benchmark suites such as SPEC [22] and MiBench [21] are not adequate to characterize the performance of all system resources used in the workflow from mobile clients to cloud servers.

6 Conclusion and Future Directions

In this paper, we proposed a single frame-based metric which can measure the performance of three main system components on cloud infrastructure for a large-scale mobile video application, especially for uploading videos from mobile clients to cloud servers. To this extent, we first deployed our mobile video management system, MediaQ, to Amazon EC2, separated video upload workflow into three phases and identified the system resources used at each stage. Using our metric, we spotted CPU as the main *bottleneck* that slows down the entire system performance. Subsequently, we proposed several approaches to enhance CPU throughput and concluded that running multiple single-threaded transcoding processes increases throughput linearly with the number of CPUs. In addition, to benchmark various server types available on EC2, we conducted our experiments on four different server families, specifically, on the smallest and the largest instance of servers to identify the *lower* and *upper performance bound*. Our experimental results show that compute-optimized machines provide the best performance for a resource intensive mobile video application.

We believe that our approach will help users to make more informed decisions in choosing server types while deploying mobile video applications to cloud infrastructures. In addition, such a cross-resource metric can be used to calculate performance-to-price ratios. As a next step, we plan to extend our frame-based metric to measure: (1) mobile devices' computing and storage capabilities, and (2) other server side processes such as query processing (e.g., range query). Moreover, we also would like to partition our dataset and scale out to multiple servers.

Acknowledgements. This research has been funded in part by NSF grants IIS-1115153 and IIS-1320149, the USC Integrated Media Systems Center (IMSC), and unrestricted cash gifts from Google, Northrop Grumman, Microsoft, and Oracle. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of the sponsors such as the National Science Foundation.

References

1. Amazon EC2. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html
2. MediaQ Framework. http://mediaq.usc.edu

3. Kim S.H., Lu Y., Constantinou, G., Shahabi, C, Wang, G, Zimmermann, R.: MediaQ: mobile multimedia management system. In:5th ACM Multimedia Systems Conference, pp. 224–235. ACM, New York (2014)
4. Oracle. http://docs.oracle.com/cd/B12037_01/appdev.101/b10795/adfns_in.htm
5. Wang, G., Eugene, T.S.: The impact of virtualization on network performance of amazon EC2 data center. In: 29th Conference on Information Communications (INFOCOM), pp. 1163–1171. IEEE Press, Piscataway (2010)
6. Amdahl G.: Validity of the single processor approach to achieving large-scale computing capabilities. In: Spring Joint Conference (AFIPS), pp. 483–485. ACM, New York (1967)
7. Cisco's Forecast. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf
8. Mc Kinsey's Forecast. http://www.mckinsey.com/insights/business_technology/disruptive_technologies
9. Curino, C., Difallah, D.E., Pavlo, A., Cudre-Mauroux, P.: Benchmarking OLTP/Web databases in the cloud: the OLTP-bench framework. In: 4th International Workshop on Cloud Data Management, pp. 17–20. ACM, New York (2012)
10. Kossmann, D., Kraska, T., Loesing, S.: An evaluation of alternative architectures for transaction processing in the cloud. In: International Conference on Management of Data (SIGMOD), pp. 579–590. ACM, New York (2010)
11. TPC: TPC-W 1.8. TPC Council (2002)
12. Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.): CD-ARES 2013. LNCS, vol. 8127, pp. 272–288. Springer, Heidelberg (2013)
13. Ffmpeg Library. www.ffmpeg.org
14. Android. http://developer.android.com/reference/android/hardware/Camera.Parameters.html#setPreviewFpsRange
15. Venkata, S., Ahn, I., Jeon, D., Gupta, A., Louie, C., Garcia, S., Belongie, S., Taylor, M.: Sd-vbs: The San Diego vision benchmark suite. In: International Symposium on Workload Characterization (IISWC), pp. 55–64. IEEE, Washington, DC (2009)
16. Cooper, B.F., Silberstein, A., Tam, E., Ramakrishnan, R., Sears, R.: Benchmarking cloud serving systems with YCSB. In: 1st ACM Symposium on Cloud Computing (SoCC), pp. 143–154. ACM, New York (2010)
17. Barahmand, S, Ghandeharizadeh, S.: BG: a benchmark to evaluate interactive social networking actions. In: Sixth Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, CA, USA (2013)
18. Patil, S., Polte, M., Ren, K, Tantisiriroj, W., Xiao, L., López, J, Gibson, G, Fuchs, A., Rinaldi, B.: YCSB++: benchmarking and performance debugging advanced features in scalable table stores. In: 2nd ACM Symposium on Cloud Computing (SOCC). ACM, New York (2011)
19. Gray, J.: The Benchmarking Handbook for Database and Transactions Systems. Morgan Kaufman, San Francisco (1992)
20. Ballani, H., Costa, P., Karagiannis, T., Rowstron, A.: Towards predictable datacenter networks. In: 17th International Conference on Data Communications (SIGCOMM), pp. 242–253. ACM, New York (2011)
21. Li, A., Yang, X., Kandula, S., Zhang, M.: CloudCmp: comparing public cloud providers. In: 10th International SIGCOMM Conference on Internet Measurements, pp. 1–14. ACM, New York (2010)
22. The Standard Performance Evaluation Corporation (SPEC). www.specbench.org
23. Guthaus, M., Ringenberg, J., Ernst, D., Austin, T., Mudge, T., Brown, R.: Mibench: a free, commercially representative embedded benchmark suite. In: International Symposium on Workload Characterization, pp. 3–14

24. Li, M.L., Sasanka, R., Adve, S.V., Chen, Y.K., Debes, E.: The ALPBench benchmark suite for complex multimedia applications. In: International Symposium on Workload Characterization, pp. 34–45. IEEE, Washington, DC (2005)
25. Luo, C., Zhan, J., Jia, Z., Wang, L., Lu, G., Zhang, L., Xu, C.Z., Sun, N.: CloudRank-D: benchmarking and ranking cloud computing systems for data processing applications. J. Front. Comput. Sci. **6**(4), 347–362 (2012)
26. Wang, L., Zhan, J., Luo, C., Zhu, Y., Yang, Q., He, Y., Gao, W., Jia, Z., Shi, Y., Zhang, S., Zheng, C., Lu, G., Zhan, K., Li, X., Qiu, B.: BigDataBenchd: a big data benchmark suite from internet services. In: 20th IEEE International Symposium on High Performance Computer Architecture, pp. 488–499, Orlando, Florida, USA (2014)