Received: 22 December 2020 | Revised: 11 July 2021 | Accepted: 5 November 2021

DOI: 10.1002/cpe.6823

RESEARCH ARTICLE

WILEY

Performance modeling for I/O-intensive applications on virtual machines

Tathagata Bhattacharya | Xiaopu Peng | Jianzhou Mao | Chaowei Zhang | Taha Takreeti | Ye Wang | Ting Cao | Xiao Qin

Samuel Ginn College of Engineering, Auburn University, Auburn, Alabama, USA

Correspondence
Tathagata Bhattacharya, Samuel Ginn College of Engineering, Auburn University, 449 North Donahue Drive, Apt B11, Auburn, AL 36832, USA.
Email: tzb0063@auburn.edu

Abstract

Models for virtual machines running on cloud computing systems. Modeling system behaviors of clouds is a grand challenge because the resource utilization in VMs is heterogeneous due to variability in workload conditions. We address this challenging issue by uniquely (1) objectifying the usage prediction of virtualized resources and (2) predicting the performance trends of programs running on clouds. At the heart of the modeling system, we pay particular attention to CPU cores, disk size, main memory space, and input data volume, which serve as important factors for the developed prediction module. We devise two resource-utilization prediction algorithms driven by two distinctive sets of I/O and CPU intensive benchmarks, where one algorithm deals with execution time and the other one revolves around input data size. We investigate the correlation between CPU/disk utilization and VM live migrations. Our system aims at not only providing performance optimization for virtualized resources but also ensuring service level agreement (SLA) and Quality of Service (QoS). The model fits the curve quite well, thereby advocating for the efficiency of the algorithm. The case studies conducted in this project draw the comparisons between the performance of striped and monolithic disks as well as bringing forth the problem of cache coherence that causes hindrance to the experiment. We also deal with the cache-coherence problem to improve the accuracy of our prediction algorithms

KEYWORDS
cache coherence, I/O intensive, linear model, monolithic and split disk, resource utilization

1 | INTRODUCTION

This article is motivated by the rapid growth in demand for computation resources in data centers and the widespread of cloud computing. Modeling system behaviors of clouds is a grand challenge because the resource utilization in VMs is heterogeneous due to variability in workload conditions. We address this challenging issue by uniquely (1) objectifying the usage prediction of virtualized resources and (2) predicting the performance trends of programs running on clouds.

This research is inspired by the following four motivations.

Motivation 1: virtualization. Cloud computing has changed the landscape of computing resources delivery models. Large-scale virtualized data centers gain popularity thanks to the rapid growth in demand for computational power-driven by modern service applications combined with the shift to the cloud computing model.[1] The virtualization technique emulates physical computers with a mirrored operating system, virtual disks, virtual CPUs, and virtual memory resources. The emergence of virtual machines promises to streamline the on-demand provisioning of software, hardware, and data as a service, achieving economies of scale in IT solutions' deployment and operation.[2] There is broad adoption of virtualization in modern business and computing environments because virtual machines offer resource utilization advantages. When a physical host machine is limited to a single dedicated operating system, the physical machine is inevitably underutilized during a long period of inactivity.[3] To address this issue, virtual machines or guest machines improve the resource utilization of the host system by simplifying resource allocations.[4] Proper resource utilization can drastically improve the efficiency of virtual environments. Virtualization is often used in cloud computing platforms for its several advantages in efficiently managing resources.[5] Resource utilization of a virtual machine is consequential as it provides the performance optimization of the physical machine.[4] High resource utilization is achieved by partitioning the resources of large physical servers into multiple smaller independent machines or a virtual machine or VM. Each VM executes in apparent isolation running its operating system and applications.[6] Resource providers focus on performance and utilization of resources considering the constraints of service level agreement. Resource performance is achieved by virtualization techniques, which share the infrastructure of the resource provider between different virtual machines.[7]

But this virtualization comes at a great cost endangering the sustainability of the earth. Therefore, we wanted to model the CPU and I/O utilization of different benchmarks (e.g., zip–unzip and install–uninstall) so that it will be easier for the power and system engineers to work toward sustainability. Through these utilization models described in the following sections below, power and system engineers would be able to estimate the energy usage of these important units of cloud infrastructure. This energy estimation would be beneficial in meeting the ever-increasing energy demand.

Motivation 2: VM consolidation. Resource utilization has become a core development objective for modern cloud infrastructures. Virtualization provides a "virtualized" view of resources used to instantiate virtual machines (VMs).[4] Resource utilization is a very challenging factor as VMs are heterogeneous due to variability in resource load. As the load shrinks and grows, the resource utilization trend varies significantly. Dynamically changing resource utilization offers ample opportunities for virtual machines to improve the energy efficiency of computing systems. Specifically, virtual machine consolidation aims at reducing the number of active physical servers in a data center, thereby decreasing the total power consumption while maintaining high computation.

Most existing VM consolidation schemes[8,9] is focused on resources like CPU and memory ignoring other vital resource types like disks and network interface. Our observation has revealed that I/O plays a very significant role in shaping the energy usage of cloud infrastructure for certain I/O intensive benchmarks (see Sections 4.3 and 4.4). We are motivated to address this technological gap by investigating I/O activities in virtual machines. Our goal is to provide performance models for I/O-intensive applications managed by cutting-edge VM consolidation techniques.

Motivation 3: modeling I/O-intensive applications. In the past decade, computing systems have become highly data-intensive; I/O intensive applications mainly signify the usage of data storage systems. To maximize storage-system usage in general and disk usage in particular, one has to delve into the behaviors of I/O-intensive applications. Unfortunately, most existing modeling methods are concentrated on either CPU or main memory utilization,[10-12] thereby overlooking the behaviors of I/O-intensive applications. The existing models are incapable of pinpointing performance issues caused by processing large data. This problem becomes pronounced when it comes to virtual-machine-based computing environments.

After thorough research we observe modeling the I/O behavior is extremely important today since most of the global applications have become data-intensive. Data centers managers and power engineers are focusing more on reducing the energy usage of the data centers in various ways. We believe our I/O intensive energy usage model would be a key to deal with the ever-increasing energy demand in the globe. Therefore, it is extremely important to design resource utilization models for I/O-intensive applications running on virtual machines.

After a thorough comparison with the previously existing models for CPU, memory, and disk usage, we figure out all the existing strategies and models have focused either on overall usage of a cloud infrastructure[13] or a generalized approach toward measuring CPU and memory usage for a virtualized environment.[14,15] Our research is unique as it deals in a microlevel and tries to figure out how the CPU, memory, and I/O load changes with various benchmarks. We experiment with most commonly used benchmarks like install–uninstall and zip–unzip. Our study reveals that each of these benchmarks applications exhibits different behavior; also, they need separate utilization models to figure out the total energy usage. Working at a microlevel with the utilization models for different benchmarks and figuring out the need for building CPU, I/O, and memory usage models in optimizing the total workload for a cloud infrastructure give our work uniqueness in every direction.

The overarching goal of this study is to construct a modeling framework for CPU- and I/O-intensive applications on clouds powered by virtual machines. The first step toward this goal is to pay particular attention to the performance profiles of a set of I/O-intensive applications running on a group of virtual machines. We develop models to shed light on the CPU and I/O performance trends of these applications in a virtual-machine-based computing environment. The modeling results obtained from this study signify that CPU utilization holds a strong correlation with disk usage. Though memory utilization only varies marginally compared to CPU and disk usage, memory usage is an affecting factor to determine the correlation between the other resource utilization. This article is objectified to provide:

The main contributions of this article are summarized as follows:

- We profile resource usage of I/O-intensive applications in virtualized computing environments.
- We design models to predict the resource utilization using linear regressions; we evaluate the performance of the predictors using mean squared error and mean absolute error.

- We conduct a comparative study between striped and monolithic virtual disks.
- We investigate performance impacts of contiguous files and fragmented files on resource utilization in virtual machines.

2 | RELATED WORK

2.1 | Virtual machine consolidation

Existing virtual machines (VMs) scheduling schemes have mainly focused on enhancing the cluster resource utilization and reducing power consumption by improving the legacy "bin-packing" algorithm.[16] It is considered to be one of the utmost important aspects of data centers and cluster computing. One of the challenges of virtualization lies in a fluctuation of resource utilization over time.[17] The workloads experience a deviation in resource utilization due to the difference in the configuration of the virtual machines as well as caching and pipelining. Live migration is a technology that replaces running VMs seamlessly across distinct physical hosts where resource management is an important parameter. Existing work[18] shows the importance of the virtual machine placement problem, and we can benefit from the appropriate placement policy on energy saving. Monitoring of resource usage is also a notable factor in live virtual machine migration. Resource utilization signifies the usage of the hardware resources for I/O intensive applications.

Existing VM consolidation techniques take into account memory and network resources since the CPU is not the only critical resource in cloud data centers. For example, a prior study shows that memory and network bandwidth can become a bottleneck, possibly causing violations in the service level agreement.[19]

We have a unique approach to this VM consolidation. We have incorporated I/O intensive data applications in our experiment which has added a new dimension to this study. I/O intensive applications have not only made it easier to determine the resource utilization of the virtual machines but also eliminated the data and resource hazard problems. This approach has proved to be quite efficient in predicting the workload as well as balancing load in a virtualized environment.

Though it is difficult to construct a predictive model for resource utilization due to unavoidable issues like a variation in software used for different benchmarks or differences in operating system and so forth, we have managed to build a predictive model that not only predicts the resource utilization at a particular instance but also graphically represent the behavior of these benchmarks. Therefore, having short-term prediction helps the cloud provider to make informed decisions and migrate the virtual machines (VMs) between data center sites in the absence of renewable energy.[20]

The modern era has witnessed massive growth in data-intensive applications. To maintain overall system performance in data centers, big-data applications that process massive amounts of data ranging from 10 to 100 MB are fed to big-data systems per second.[21] Even in para-virtualized environments, it is extremely important to figure out the I/O response as the virtualization is handled by the virtualized layer, without the hardware.[22] Traditional computers often cannot meet the demand for performance when running these applications acquiring and handling massive data.[23] But Cloud platforms have provided a great opportunity to measure the workflow through virtual mirrors. To increase the system utilization of the physical host and the server, disk utilization has been given a high priority.[24] The static and dynamic approaches toward handling I/O-intensive applications become highly essential in the field of high-performance computing.[25-27] These days even Multimedia computing is considering approaches to efficiently handle continuous media. This proliferous growth of the volume in data centers has motivated us to develop models for I/O-intensive applications running in a virtualized environment. Thus modeling I/O intensive resource utilization and deriving its impact on scheduling algorithms can not only fix up the bottleneck problem but also brings a panacea to the whole system.[28]

2.2 | I/O-intensive computing

Though the era of software has made it easier to deal with challenges in modeling I/O behaviors, it is still a challenge to maintain the quality of service of I/O intensive applications running on a virtual machine. The context switching overhead and events in VMM make the I/O behavior unpredictable to the end user.[29] In fact, there is no graphical approach that can demonstrate the I/O behaviors in certain I/O applications.[30] A very recent trend of CloudSim and PerficientCloudSim supports large-scale computation in a real environment.[31] Other than that researchers have come up with I/O energy models to illustrate that the I/O unit significantly contributes to the power consumption in a distributed environment.[32] Our work goals toward improving this I/O utilization for a set of the virtual machine which is running on cloud. Even this model extends the chance to predict the virtual machine migration for a set of over and under-utilized hosts running on a cloud. The uniqueness of this article lies in modeling these resource utilization graphs for this virtualized environment as well as predicting the CPU, memory, and I/O behavior of these virtual machines.

2.3 | Modeling resource utilization for virtual machines

Modeling resource utilization of virtual machines running on the cloud infrastructure has been always challenging.[33] Bohra and Chaudhary[34] suggested that though plenty of research is ongoing toward developing an integrated cloud management system to perform proper resource utilization

TABLE 1 Comparison table

Paper	VM consolidation	I/O intensive computing	Modeling resource usage
[1-7,11-23]	Yes	No	No
[24-27]	No	Yes	No
[8-10,28-41]	No	No	Yes
MOOSE	Yes	Yes	Yes

while reducing power consumption, most of these techniques provide online power monitoring based on the power consumption of a physical node running one or more virtual machines (VMs). As resource utilization is crucial to facilitate live VM migrations, a raft of models was developed to meet the never-ending need for energy.[35] Alzamil[36] proposed the ARIMA model that applies static and periodic consumption data to predict future VM energy consumption. Several novels approach toward modeling utilization of resources (e.g., CPU, memory, and I/O) of VMs have been proposed to characterize cloud computing workload conditions.[37-40]

Very recently Rafael et al. proposed a power consumption model that exclusively accounts for I/O intensive workload.[41] Mohamed et al. proposed a cloud usage trend study to measure the accuracy of VM replacement and migration policy.[42] Like Chaoqiang et al. we believe that to build energy or power consumption models, linear regression or polynomial regressions are apt.[43] Energy-aware scheduling[44] and power consumption model of cloud data centers based on feature selection, deep learning,[45] and ANN[46] is quite rigorous in the field of power consumption modeling of VMs.

Unlike the aforementioned studies, our work is centered around a set of models that deliver a good fit to predict the resource utilization for specific applications running on virtual machines in cloud infrastructures. Amid our empirical study we come across intriguing challenges (see Section 3.3) and crucial case studies (see Section 5) that bring forth a whole new dimension to the research. Our work mainly focuses on keeping track of changes in I/O and CPU behaviors concerning various benchmarks running on clouds. More importantly, in this study, we graphically represent the behaviors of virtual CPUs and I/O modules to shed a bright light on the performance of I/O-intensive applications on VMs.

To make it easier for the readers to compare the above-mentioned references with our paper, we have provided a comparison table below (see Table 1):

3 | SYSTEM DESIGN

3.1 | Framework

Figure 1 depicts the framework of the *MOOSE* system. We have given name of this architecture as MOOSE after the animal Moose. MOOSE orchestrates a set of virtual machines running independently on a cloud-computing infrastructure. MOOSE system monitors and models the resource utilization of the VMs running independently on the cloud infrastructure. As virtual machines are deployed in order to increase the resource utilization of physical host machines, it is of utmost importance to configure these virtual machines in a proper way. We run experiments on the Auburn cloud offering computing resources in the form of VM instances according to the *IaaS* delivery model. The main functionality of MOOSE is to model and project the resource usages of the virtual machines running on this cloud platform. After MOOSE collects the resource usage patterns from applications running on the virtual machines, a model is constructed to predict future resource utilization from workload characteristics imposed by applications.

Now we denote a cloud computing form as an array $PM = \{P_1, P_2, \dots, P_i, \dots, P_n\}$ of n physical machines. Given a set $VM = \{v_1, v_2, \dots, v_j, \dots, v_m\}$ of m virtual machines, we assign these virtual machines to physical machines PM. Suppose the number of virtual machine running on the ith physical machine is q_i, we express the set PM_i of virtual machines as

$$P_i = \{v_{i1}, v_{i2}, \dots, v_{iq_i}\}. \tag{1}$$

Virtual machine set VM is expressed as a summation of all the virtual machines assigned to the physical machine PM. Thus, we have

$$VM = \{v_1, v_2, \dots, v_j, \dots, v_m\} = \bigcup_{i=1}^{m} P_i,$$
$$\text{where } \forall P_i, P_j : P_i \cap P_j = \emptyset. \tag{2}$$

The condition $\forall P_i, P_j : P_i \cap P_j = \emptyset$ in (2) entails that each virtual machine can only run on a single physical machine; any pair of two physical machines share no virtual machine.

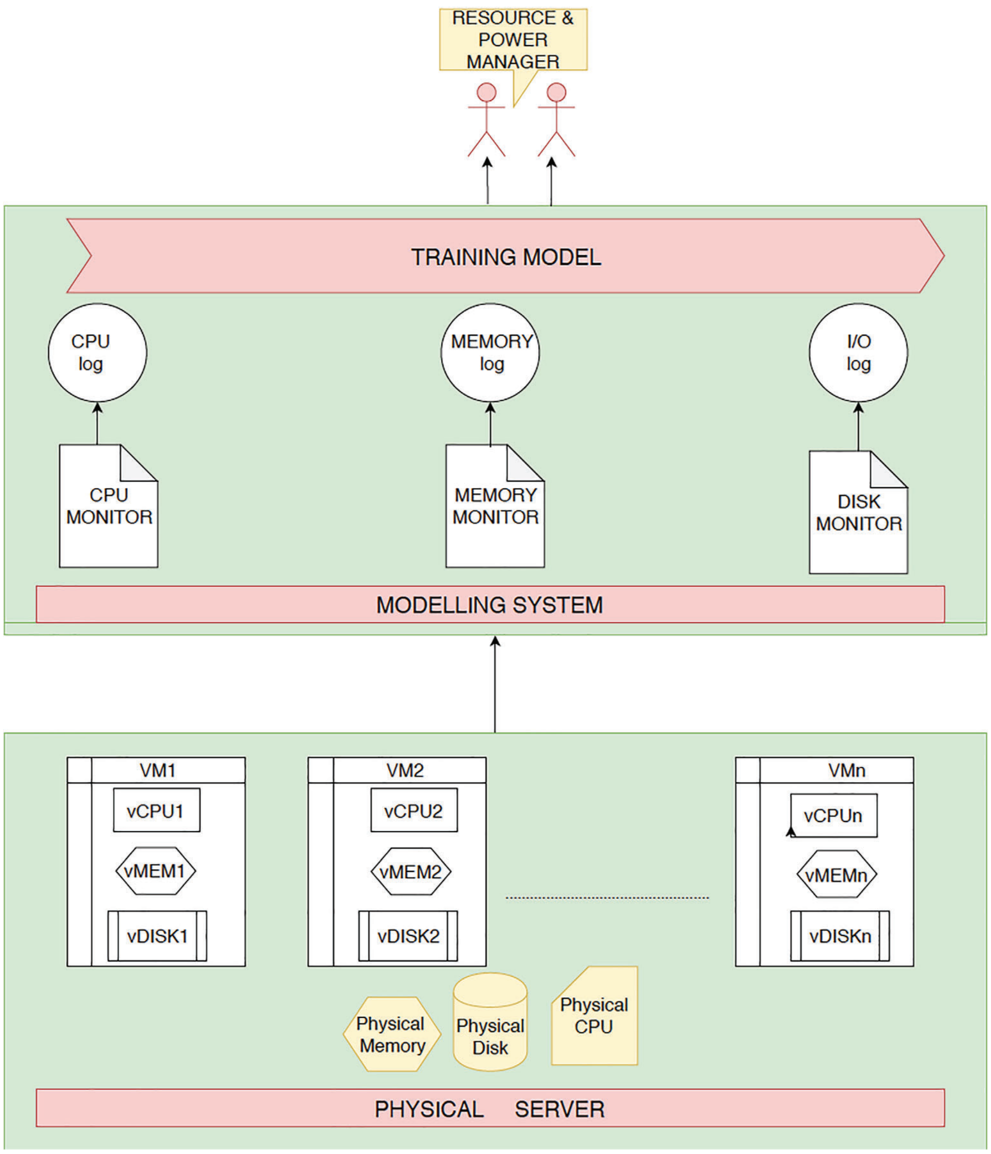


FIGURE 1 The MOOSE system orchestrates a set of virtual machines running independently on a cloud-computing infrastructure. After historic resource usages are collected by MOOSE, a model is constructed to predict future resource utilization patterns from workload characteristics

We pay heed to the three types of resources in virtual machines, namely, processors, main memory, and disk I/Os. Let r_j be the resources of the ith virtual machine v_j. Because r_j embraces three resource types, we model r_j as $v_j = \{r_j^{CPU}, r_j^{mem}, r_h^{IO}\}$. $r_j^{CPU}, r_j^{mem}, r_h^{IO}$ indicate the utilization of the CPU, memory, and disk resources in virtual machine v_j. The total CPU resource utilization of physical machine P_i is derived from the CPU utilization of all the virtual machines running on P_i. Thus, we express CPU utilization R_i^{CPU} of the ith physical machine as

$$R_i^{CPU} = R_{i,init}^{CPU} + R_{i,vm}^{CPU} = R_{i,init}^{CPU} + \sum_{v_j \in P_i} r_j^{CPU}, \tag{3}$$

where $R_{i,init}^{CPU}$ is the baseline CPU utilization of machine P_i without orchestrating any virtual machine, $R_{i,vm}^{CPU}$ is the CPU utilization contributed by the running virtual machines. Similarly, we gauge the overall memory and disk utilization of the ith physical machine as

$$R_i^{mem} = R_{i,init}^{mem} + \sum_{v_j \in P_i} r_j^{mem}, \text{ and} \tag{4}$$

$$R_i^{IO} = R_{i,init}^{IO} + \sum_{v_j \in P_i} r_j^{IO}, \tag{5}$$

where $R^{mem}_{i,init}$ and $R^{IO}_{i,init}$ are the baseline main memory and disk utilization measures of machine P_i when no virtual machine is running on P_i.

We express this whole details of virtual infrastructure in Figure 2

3.2 Experimental procedure

To experiment efficiently, we carefully design and update our methodology in each iteration. We conduct two sets of experiments to profile the behaviors of CPU- and I/O-intensive applications. Initially, we conduct our experiment in a generalized way by focusing only on normal resource utilization. Once we figure out that varying data size for I/O intensive benchmark applications and varying time for CPU intensive applications opens a different horizon for resource usage for different benchmarks, we plan to update our research in the following ways. We realize that based on the network traffic, the CPU usage changes drastically for the same application at different times of the day. Also, our research suggests that for I/O intensive applications like zipping and unzip, changing the file size would vary the I/O usage. Therefore, we update our experiment by focusing on two main points to construct an optimized resource usage model. The first set of experiments performed on I/O intensive benchmark applications are focused on the impact of *data size*, whereas in the second group of experiments we repeatedly run the CPU-intensive applications *multiple times on different days*. More specifically, we gauge profiling data in the empirical study centered around the following two experiment groups.

- **Experiment group 1.** To stress CPU workload on the virtual machines, we install and uninstall a wide range of software tools on various days over a period of one month.
- **Experiment group 2.** To profiling the zip and unzip benchmark, we vary input data size to model the impacts of data size on a load of virtual disks.

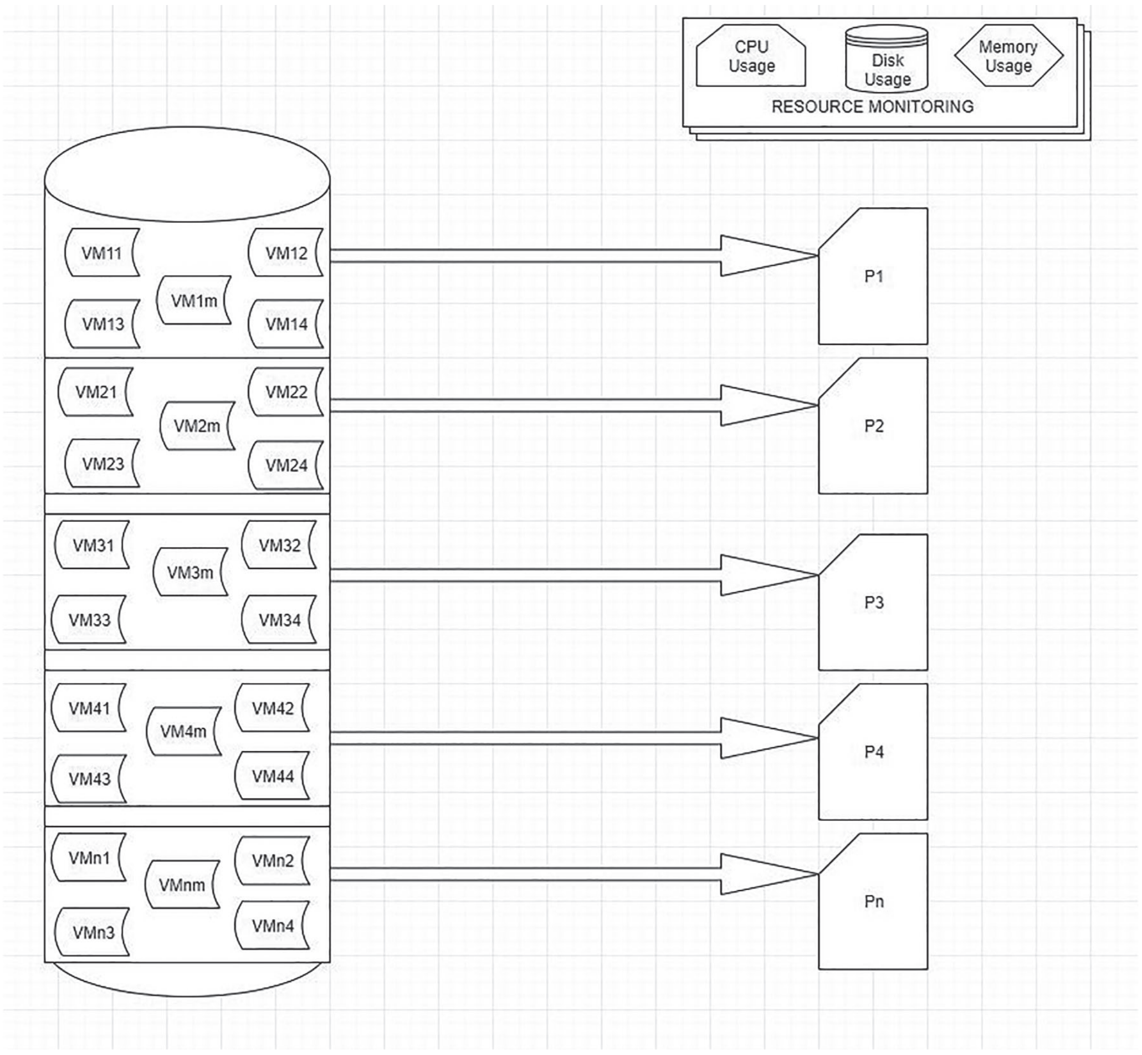


FIGURE 2 The MOOSE system demonstrates n number of physical machines of which each physical machine has m number of virtual machines and no two physical machine have shared virtual machine

Now we outline the two-step profiling methodology of this study (see Figure 1 for the framework). The first step is to create a set of virtual machines running on Auburn's cloud computing infrastructure. The heterogeneous configurations of virtual machines are reliance on the capacity of physical resources (e.g., the number of cores of CPU, the size of the main memory, and the I/O bandwidth). Next, we kick in I/O-intensive applications coupled with non-I/O-intensive counterparts executing on these virtual machines. We ensure that all the applications are running for a long time period to calibrate any random errors.

We partition the collected data into two sets: the first dataset is dedicated to model training; the second dataset is reserved for the purpose of testing the accuracy of our model. Initially, we put a tremendous amount of effort to standardize our experiments, setting up the assumptions as well as evaluation objectives (see also challenges in Section 3.3). We have to rigorously go through the entire experimental results to process data cleaning, thereby pruning suspected data abnormalities.

During the initial tests, we observe that the I/O, as well as the CPU utilization, falls sharply when we run the benchmarks due to the caching effect (see also the challenges in Section 3.3). The CPU and I/O utilization measures vary dramatically because cache coherence and prefetching have become significant contributors in our empirical study. Therefore, we decide to manually run the profiling experiments rather than relying on an automation script. More specifically, the time interval between two consecutive experiments is one day; we run 30 experiments during a period of one month. We reckon that this methodology more accurately resembles real-life settings than running tests back-to-back with data cleaning or buffer refreshing. Our methodology is effective and practical because there is no need to clear the cache or calibrated results. Also as different benchmarks impose various I/O and CPU utilization, we fix or standardize particular software and browsers to conduct our research. Henceforth, we decide to conduct our experiment with the effects of cache coherence and prefetching just to provide the world a more realistic experience

The profiling experiments are conducted in the following five steps:

- To build linear models, we separately collect results for each benchmark. We partition the results into two datasets: one is designed for training the models, whereas another one is dedicated to testing the efficiency of the models.
- Once the data is collected, we denote the correlation of the parameters so that we can build a perfectly fitted model with correlated parameters.
- We analyze the results affected by cache coherence and prefetching to show the discrepancy between the erroneous and calibrated results.
- The parameters that have the best goodness of fit are selected to build the models, which are used to predict resource utilization of the test dataset to determine the accuracy and mean absolute error of the models.
- We graphically plot our findings to intuitively show the trends of the results.

3.3 | Challenges

We face a handful of significant challenges while carrying out the experiments.

Challenge 1: Cache coherence. Caching is a popular mechanism to boost I/O performance in modern computer systems. The main purpose of deploying caches is to speed up system performance by storing the most frequently used data blocks. Cache coherence is a critical issue to be addressed in multiprocessing environments; this cache property becomes a peril for our experiments. Due to caching frequently used data blocks, disk utilization drops sharply and the profiling results become fallacious.

We compress and uncompress a group of files multiple times. Initially, the I/O load is high in the first round of the experiments. When we repeatedly zip and unzip the files, the I/O load significantly drops thanks to the caching effect. We decide to overcome the caching effect to optimize the experimental results. To mitigate the side-effect imposed by caching, we clear the I/O buffers after each zip/unzip operation. Figure 3 show the caching effect of the profiling experiments.

Caching is a very practical effect for almost all applications in the world of computing. Though the caching effect is noise for our experiment, we decide to keep the effect of caching in our research to provide a real and practical feel for the network and power managers.

Challenge 2: Prefetching. The concept of prefetching lies in retrieving data prior to its access. The prefetching technique speeds up I/O performance by hiding I/O accesses. For example, all data that are used in loops are prefetched into I/O buffers before being accessed by a processor. By reducing the I/O latency, this technique shortens the time spent in manipulating data and instructions that are used multiple times in a loop.

The prefetching technique becomes a hazard during the course of our performance modeling process. Though the benchmarks run fast, I/O and CPU utilization fall sharply as the benchmarks are processing the same files. In order to eliminate the side effects imposed by prefetching, we decide to run one experiment per day. This strategy resembles real-world scenarios where warm data are accessed no more than once per day. Running one experiment per day allows us to successfully collect flawless data to build performance models.

But, we decide to keep the effect of prefetching in our experiment to give it a more real-world feel. Since most of the applications today in a cloud infrastructure deal with prefetching that speedup the throughput, we decide not to eliminate the effect of prefetching in our study.

Challenge 3: Standardization. The problem of standardization arises for all the benchmarks tested in our profiling study. A benchmark might exhibit various I/O and CPU utilization when processing the same amount of input data. For example, to install the 38-MB Road Rash game, the Mozilla Firefox and Google Chrome browsers have heterogeneous resource utilization and installation times. Likewise, the performance of zipping

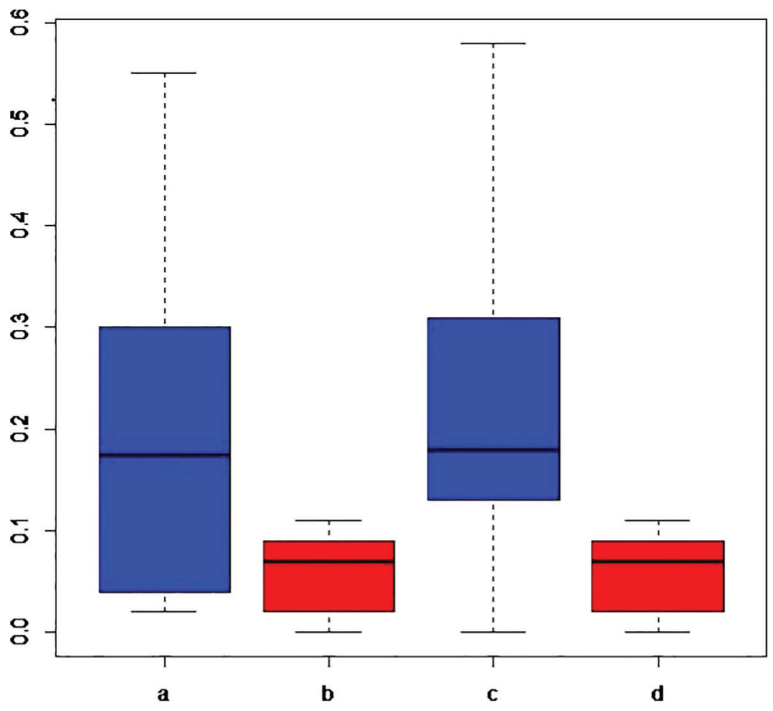


FIGURE 3 The image determines the I/O utilization for the benchmarks Zip and UnZip before and after Cache Coherence. The a and b box plots signify the I/O utilization before and cache coherence effects for zip whereas c and d box plots signify The blue box plots signify the I/O utilization before and cache coherence effects for unzip

and unzipping data depends on the evaluated benchmarks. For instance, 7 Zip shows 50% CPU utilization for zipping, whereas WinZip uses 80% CPU utilization to compress the same file. Thereafter, it becomes nontrivial to systematically gauge measures in experiments to build models. To address this concern, we decided to profile a particular application (a.k.a., standard application) for each type of the benchmarks. This strategy simplifies our procedure of conducting experiments and managing data without losing generality.

4 | MODELING VIRTUAL MACHINES

4.1 | TestBed

We profile the resource utilization of virtual machines running on multiple physical servers. Due to the challenges (see Section 3.3), we manually test the virtual machines without relying on an automation procedure. The server is equipped with Intel(R) Xenon(R) CPU 3.70 Hz, 16 GB main memory, and a hard disk of 476 GB. The parameters of the testbed are summarized in Table 2 below.

4.2 | Configuring heterogeneous virtual machines

To profile the resource utilization of the virtual machines running independently on the Auburn Cloud infrastructure, we run four different types of benchmarks, namely, zip, unzip, install, and uninstall. Our experiments diverge from each other and the configuration details are described as follows (Table 3).

TABLE 2 The configuration of the testbed

Hardware	Configuration
Processor	Intel(R) Xenon(R) CPU-1245v6 @3.70 Hz
RAM	16 GB
System type	64 bit operating System, x64-based processor
Disk	476 GB

TABLE 3 Experiment configuration

VM ID	CPU cores (%)	Disk size (GB)	Memory size (GB)
1	2	35	4
2	6	20	3
3	8	30	1.5

For each tested physical machine, we run three virtual machines with heterogeneous configurations. For simplicity, convenience, and clarity we have limited the usage of virtual machines to be 3. More specifically, the first virtual machine has a dual-core processor along with a 35 GB disk and a 4 GB main memory. The second virtual machine comprises a hex-core processor, 20 GB disk, and 3 GB memory. The third virtual machine is configured with an octa-core processor accompanied by 30 GB disk and 1.5 GB main memory. These combinations of heterogeneous CPUs, memories and I/O units are randomly chosen so that none of these can have a biased effect on the experimental results. All these virtual machines are running independently and concurrently on a set of physical machines on the Auburn cloud. We also need to mention that heterogeneity of processor, memory, and disk contributed significant diversity on the results of our experiment mentioned below in the graphs and figures. The experiments signify that only processor configuration or I/O or memory size is not an important factor to speed up. We need to make a careful combination of all three to increase the throughput and enhance parallelism in the overall architecture.

Zip, unzip, install and uninstall are few important benchmarks for profiling the I/O and CPU behavior. These I/O intensive benchmarks are used to compress, decompress, install and remove files from our workstations. We project how different combination of heterogeneous combinations of CPU and I/O contributes significant diversity in this project. We finalize these four benchmarks since they have a crucial impact on the I/O and CPU utilization. Let us dig into the details of these benchmarks.

To summarize, we claim we use a heterogeneous cloud infrastructure with heterogeneous configurations for virtual CPU, memory, and disk to provide an optimized model for specific benchmark applications running on cloud infrastructure. We intentionally pick different combinations of CPU, memory, and I/O units for the experiment. In the process, we realize that these different combinations will optimize the model and make it work well with diverse processors, RAM, and disk configurations. Therefore we claim that our model works for almost all the combinations for CPU, memory, and disks. This model has not restricted itself just to a particular combination of VM resources.

4.3 | Modeling zip on virtual machines

To evaluate the intensive I/O workload, we intend to model Zip's I/O activities on virtual machines. Figure 4 plots the I/O behaviors with the perspective of the three disks using different configurations. Though we capture the CPU and I/O behavior for zipping files of different sizes (e.g., 177,

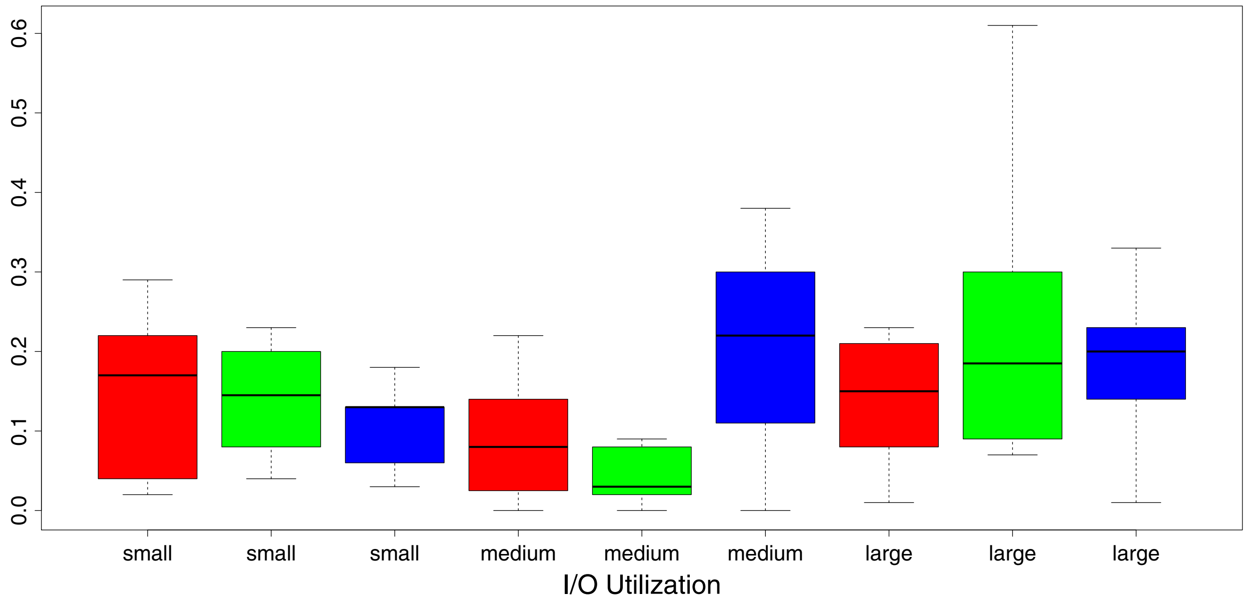
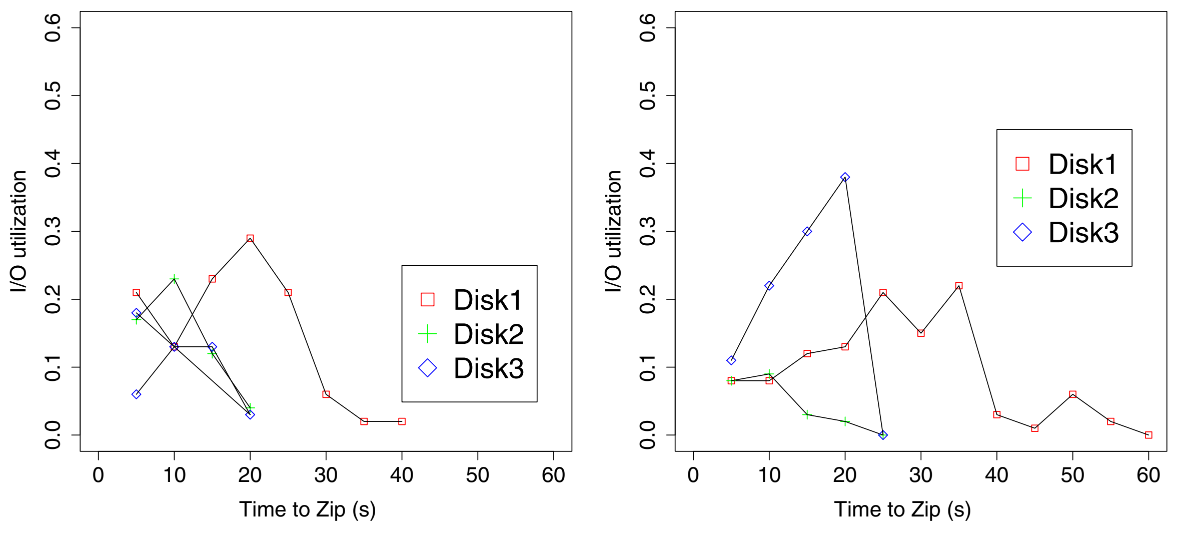


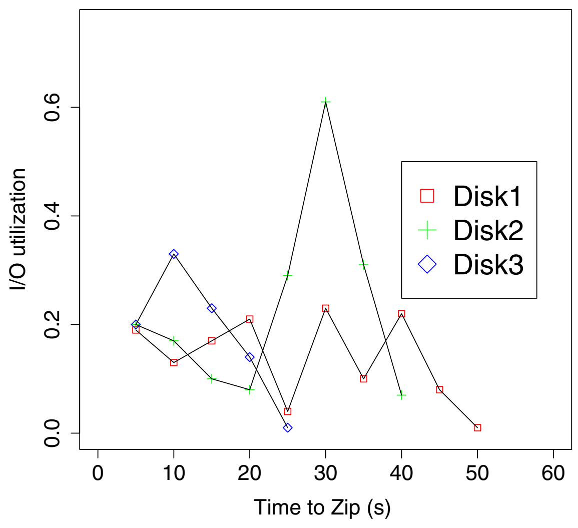
FIGURE 4 I/O utilization of the Zip benchmark compressing small, medium, and large files stored in the heterogeneous disks. The first group of three bars plots the I/O utilization for compressing small files on a virtual disk of size 35 GB (red color), 20 GB (blue color), and 30 GB (green color). The second group of three bars represents the I/O utilization for compressing medium size files on the same virtual disks consecutively. The third group of three bars represents the I/O utilization for compressing large files on the same virtual disks

247, 40, 222, and 90 MB) for constructing a strong linear model we decide to project the I/O and CPU usage behavior for three different file sizes, that is, small (74 MB), medium (158 MB), and large (320 MB) for simplicity. We repeat this for every benchmark. Figure 5 captures the I/O behavior of zipping processing small, medium, and large size files. The I/O utilization trend is mainly contributed by I/O fragmentation and CPU configuration. Tables 4 and 5 summarize the average and maximum I/O utilization in the cases of heterogeneous disks.

Tables 4 and 5 unveil a very strong correlation (i.e., −0.6327944) between the CPU configuration and disk utilization. The modeling result signifies that I/O utilization is strongly dominated by the number of CPU cores. Therefore, disk 1 (i.e., disk size-35GB, and CPU cores-2) has higher utilization than that of disk 3 (i.e., disk size-35 GB, and CPU cores-6). Also, the negative correlation indicates that when the number of CPU cores goes up, the I/O utilization declines. The correlation between CPU usage and the I/O configuration is 0.2872345, meaning that such a correlation is fairly weak. Henceforth, we can conclude that CPU configuration has a strong impact in deducing the I/O utilization in the realm of compressing files.



(A) I/O utilization to zip a small size file (B) I/O utilization to zip a medium size file



(C) I/O utilization to zip a large size file

FIGURE 5 I/O utilization for compressing a small, medium and large size file

TABLE 4 Mean I/O utilization for zip

File size	Disk 1 (35 GB)	Disk 2 (20 GB)	Disk 3 (30 GB)
small	19%	10%	17%
medium	10%	20%	5%
large	17%	22%	20%

TABLE 5 Maximum utilization for zip

File size	Disk 1 (35 GB)	Disk 2 (20 GB)	Disk 3 (30 GB)
small	30%	27%	20%
medium	24%	10%	40%
large	25%	60%	32%

Figure 6 reveals the CPU utilization of this benchmark under the three settings (i.e., dual, hex, and oct cores). The result portrays that a dual-core processor has the highest utilization; the Hex-core processor's utilization level is medium, and the Oct-core processor shows the least utilization. We summarize the average and max utilization of CPU in Tables 6 and 7. As we expected, increasing the number of cores in a virtual machine reduces CPU utilization. To some extent, the CPU behavior for large files is affected by the disk size with a correlation of 0.2872345.

Figure 7 illustrates the CPU behaviors of dual-, hex-, and octa-core processors compressing files of different sizes. The three subfigures point out the discrepancy in CPU utilization when the small, medium and large files are compressed by the heterogeneous CPUs.

Next, we determine the time required to compress the same file using the three heterogeneous disks and CPU. Figure 8 depicts the time spent in compressing files of different sizes using dual-, hex-, and Oct-core processors. Figure 8 again advocates for the strong correlation between the CPU cores and the processing time. A greater number of cores leads to better performance. To be more precise, the number of cores reduces the execution time of this benchmark. Therefore, we conclude that the time to compress a file varies inversely proportional to the number of cores of the CPU. Table 8 lists the time required to compress the small, medium, and large files.

Now, We are in a position to build an I/O-utilization model for the ZIP benchmark running on virtual machines. Equation (6) represents a linear model to deduce disk usage for a virtual machine running independently on a cloud. To build this model, we make use of CPU usage (i.e., U_{CPU}) and the number of CPU Cores (i.e., N_{CPU}) as two significant parameters to predict disk usage (i.e., U_{disk}). The model has an R squared value of 0.1985, indicating moderate goodness of fit.

$$\begin{aligned} U_{disk} = {} & 0.1526495 \times U_{CPU} + 0.0276719 \times N_{CPU} \\ & + (-0.3190342) \times U_{memory} + (-0.0009533) \times S_{disk}. \end{aligned} \tag{6}$$

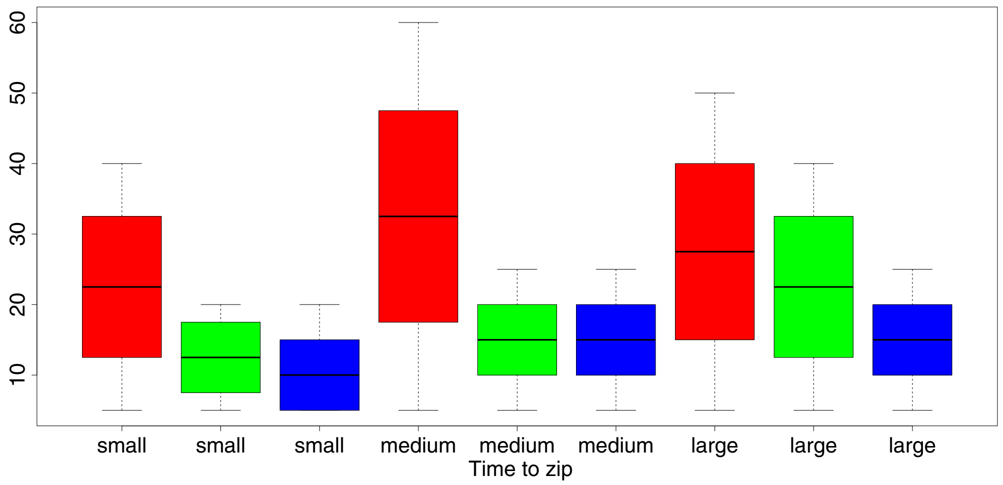


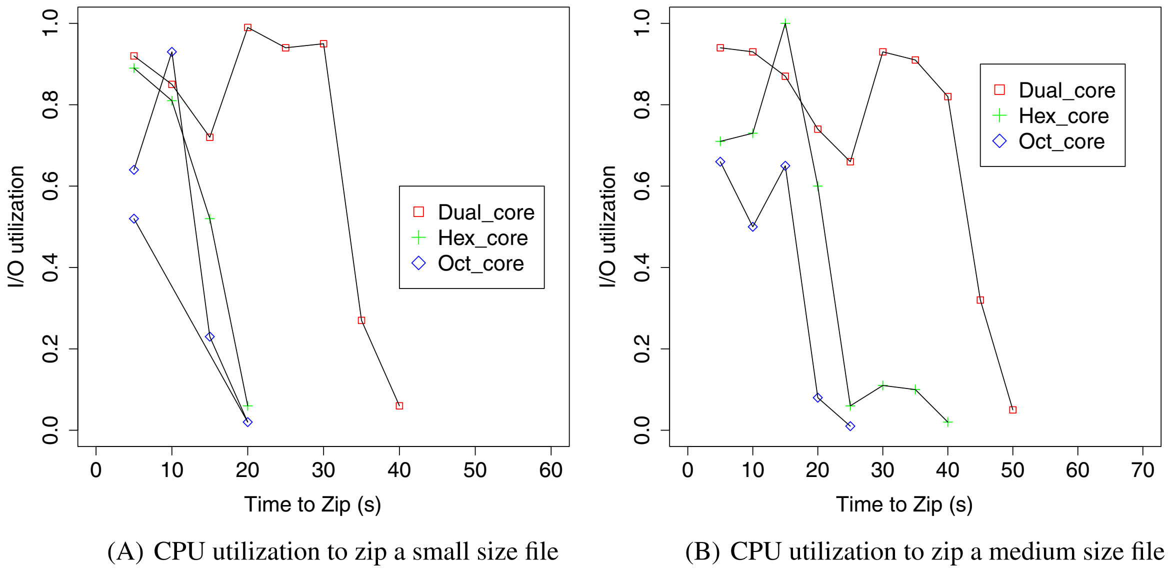
FIGURE 6 CPU utilization of the benchmarks zip for the heterogeneous disks, in which small, medium, and large files are compressed. The first group of three bar plots represents the CPU utilization for zipping small files by a dual-, hex- and oct- core processor. The second group of three bar plots represents the CPU utilization for compressing medium size files by the same group of heterogeneous processors. The third group of three bar plots represents the CPU utilization for compressing large-size files by the same group of heterogeneous processors

TABLE 6 Mean CPU utilization for zip

File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
small	85%	65%	57%
medium	98%	62%	49%
large	85%	19%	20%

TABLE 7 Maximum CPU utilization for zip

File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
small	100%	85%	95%
medium	100%	98%	79%
large	95%	70%	37%



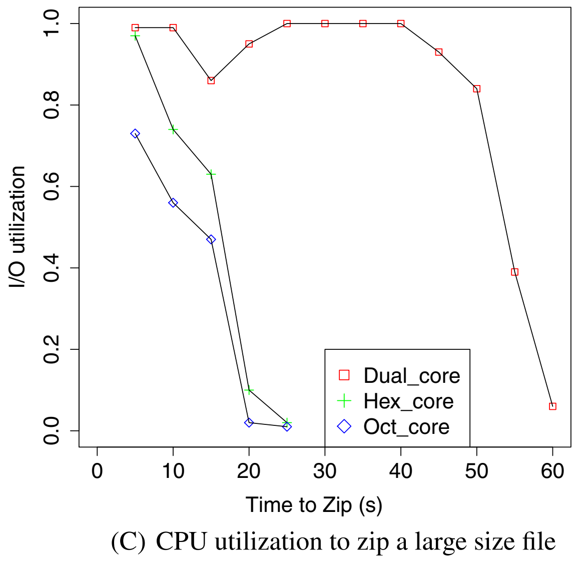


FIGURE 7 CPU utilization for compressing a small, medium, and large size file

Moving forward with our ideas, we develop a second model to determine the CPU utilization of these virtual machines for the benchmark zip. Equation (7) outlines a linear model to project CPU utilization of virtual machines. The parameters in In this model include CPU cores (i.e., N_{CPU}), memory size (i.e., S_{memory}), and disk utilization (i.e., U_{Disk}). The R squared value of the model is 0.2431, representing decent goodness of fit and significance.

$$U_{CPU} = -0.09753 \times N_{CPU} + (-0.12845) \times S_{memory} + 0.83598 \times U_{Disk}. \tag{7}$$

Next, we try to plot the I/O and CPU behavior of unzipping performed on virtual machines in the cloud infrastructure.

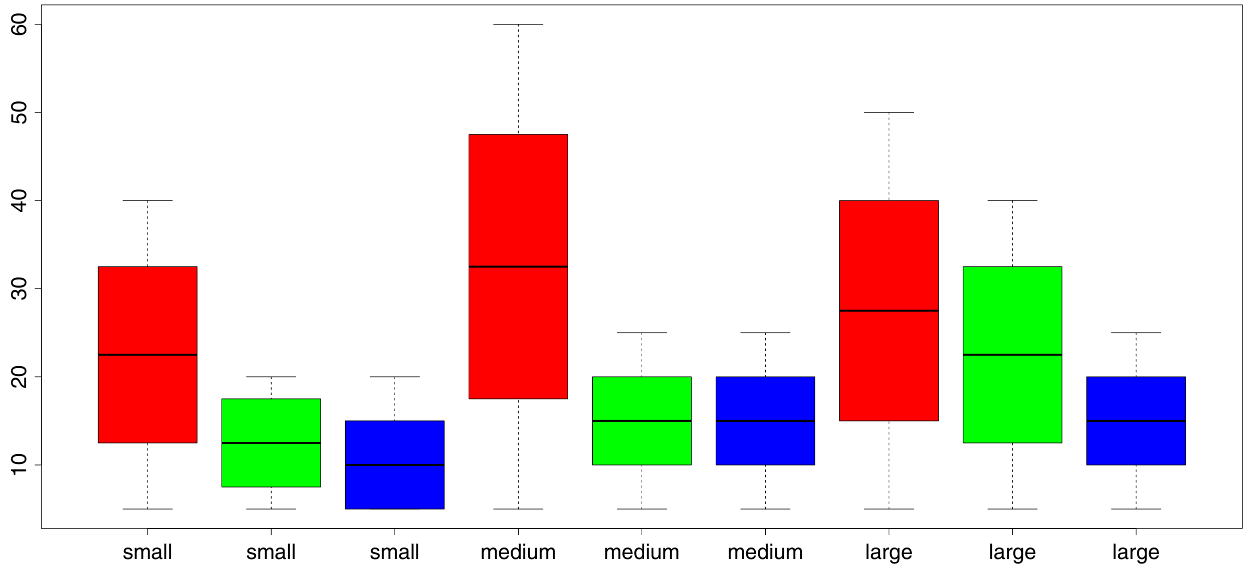


FIGURE 8 The image orchestrates the time taken to compress files of the small, medium, and large size by heterogeneous processors. The first group of three bar plots represents the time to zip small-size files by a dual-, hex-, and oct- core processor. The second group of three bar plots represents the time to zip medium-size files by the same heterogeneous processors. The third group of three bar plots represents the time to compress large-size files by the same group of processors

TABLE 8 Time to compress

File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
Small	40 s	20 s	19 s
Medium	60 s	25 s	23 s
Large	50 s	45 s	27 s

4.4 | Modeling unzip on virtual machines

Unzipping a compressed file is notably I/O intensive. We observe that unzipping has high I/O utilization as the I/O module constantly decompresses the compressed files to restore the original sizes. Figure 9 plots the I/O behavior of the heterogeneous disks where files with various sizes are

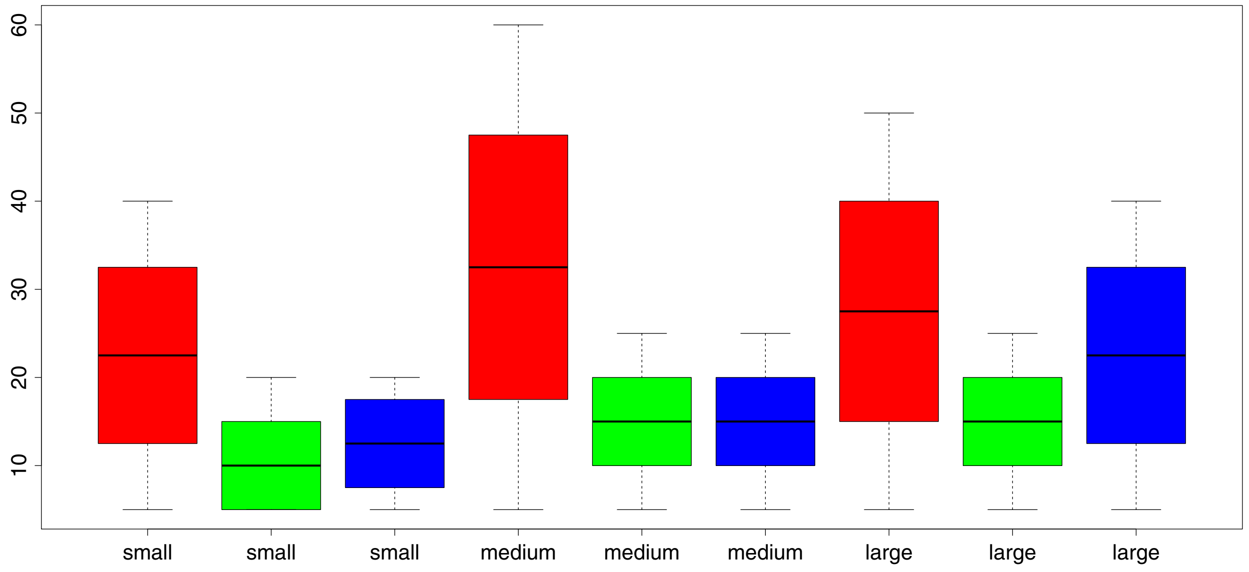


FIGURE 9 I/O utilization of the UnZip benchmark de-compressing small, medium, and large files stored in the heterogeneous disks. The first group of three bars plots the I/O utilization for unzipping small files on a virtual disk of size 35 GB (red color), 20 GB (blue color), and 30 GB (green color). The second group of three bars represents the I/O utilization for unzipping medium size files on the same virtual disks consecutively. The third group of three bars represents the I/O utilization for unzipping large files on the same virtual disks

decompressed. Figure 10 depicts the behavior of the heterogeneous disks, the average and maximum I/O utilization of which are summarized in Table 9 and 10.

After analyzing the result in Tables 9 and 10, We discover that there is no strong correlation between CPU configuration and disk utilization (i.e., 0.2665057). On the flip side, the correlation between disk size and disk usage is fairly strong (i.e., −0.423376). It signifies that in this benchmark, I/O utilization is strongly dominated by disk size. Also, the negative correlation signifies when the number of disk sizes goes up, the I/O Utilization declines. Unlike zip - a CPU-intensive benchmark, unzip is proved to be I/O-intensive.

Figure 11 reveals the CPU utilization for this benchmark under the three settings (i.e., dual, hex, and oct cores). The result portrays that increasing the number of cores pushes down the CPU utilization. Similar to the zip case, the number of cores in a processor has a deep impact on CPU

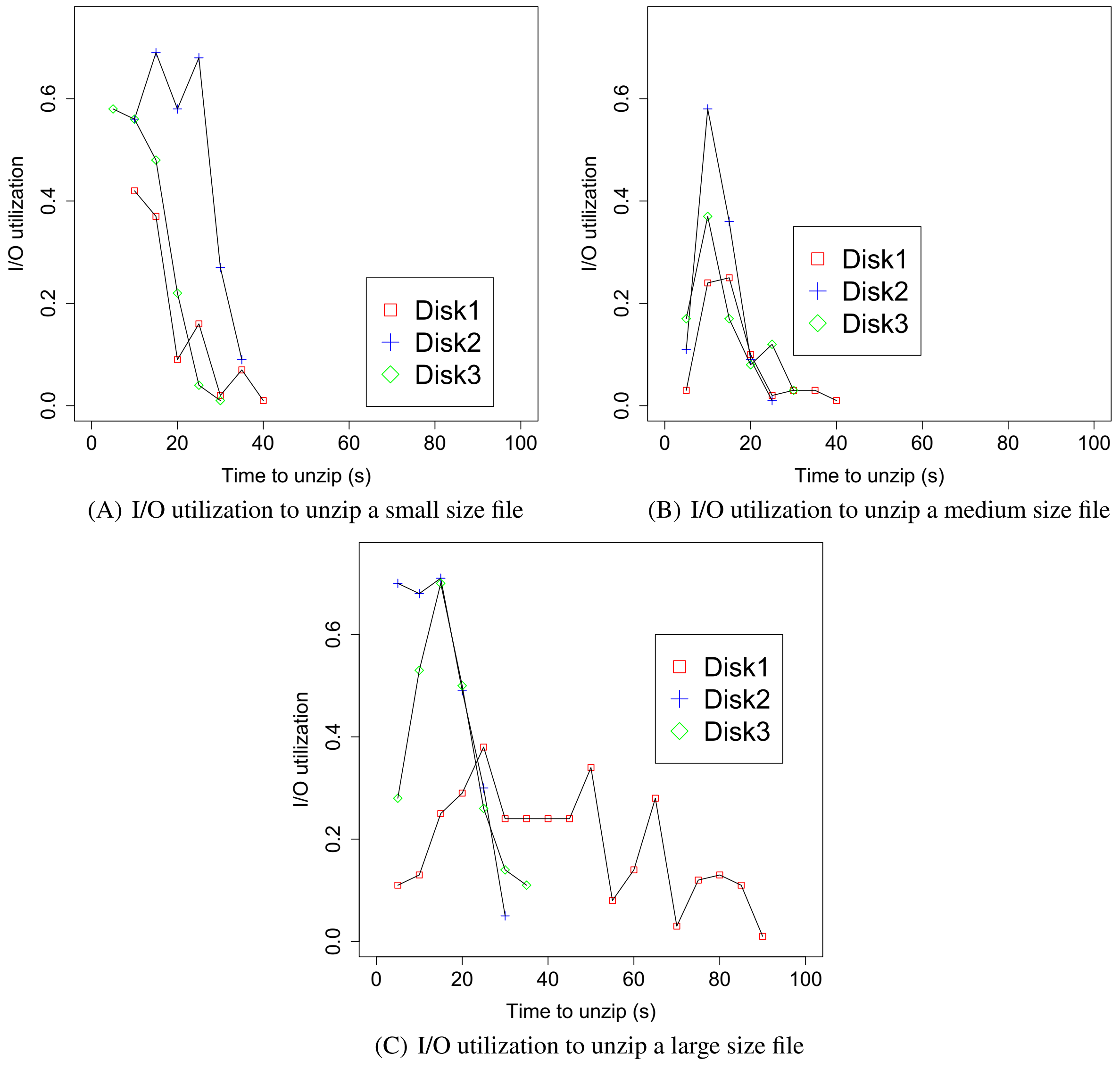


FIGURE 10 I/O Utilization for decompressing a small, medium, and large size file

TABLE 9 Mean I/O utilization for unzip

File size	Disk 1 (35 GB)	Disk 2 (20 GB)	Disk 3 (30 GB)
Small	18%	10%	12%
Medium	32%	8%	18%
Large	29%	19%	22%

TABLE 10 Maximum I/O utilization for unzip

File size	Disk 1 (35 GB)	Disk 2 (20 GB)	Disk 3 (30 GB)
Small	40%	20%	22%
Medium	20%	16%	25%
Large	50%	29%	39%

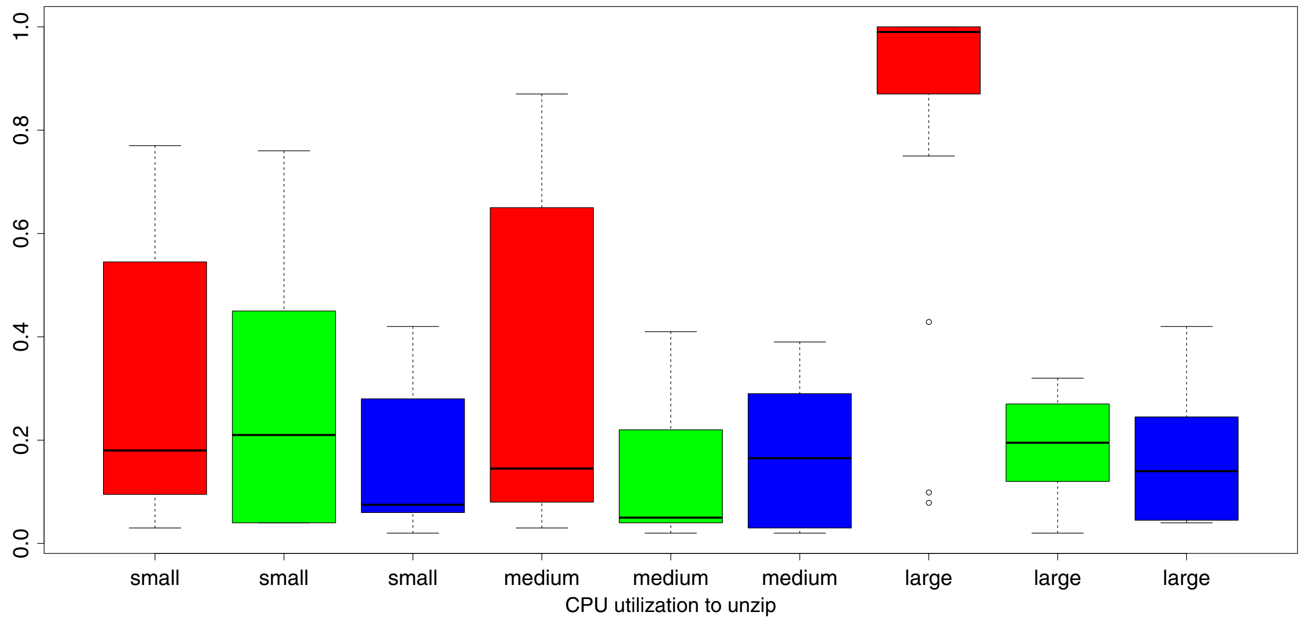


FIGURE 11 CPU utilization of the benchmarks unzip for the heterogeneous disks, in which small, medium, and large files are decompressed. The first group of three bar plots represents the CPU utilization for unzipping small files by a dual-, hex- and oct- core processor. The second group of three bar plots represents the CPU utilization for unzipping medium size files by the same group of heterogeneous processors. The third group of three bar plots represents the CPU utilization for unzipping large-size files by the same group of heterogeneous processors

utilization (i.e., –0.438795). The average and max utilization of CPU in the unzip case are presented in Tables 11 and 12. The results in the two tables suggest a strong correlation between the CPU cores and CPU utilization.

Figure 12 illustrates the CPU behaviors of dual-, hex-, and octa-core processors decompressing small-, medium-, and large-sized files. The trends are shown in Figure 12 are consistent with those demonstrated in Figure 11.

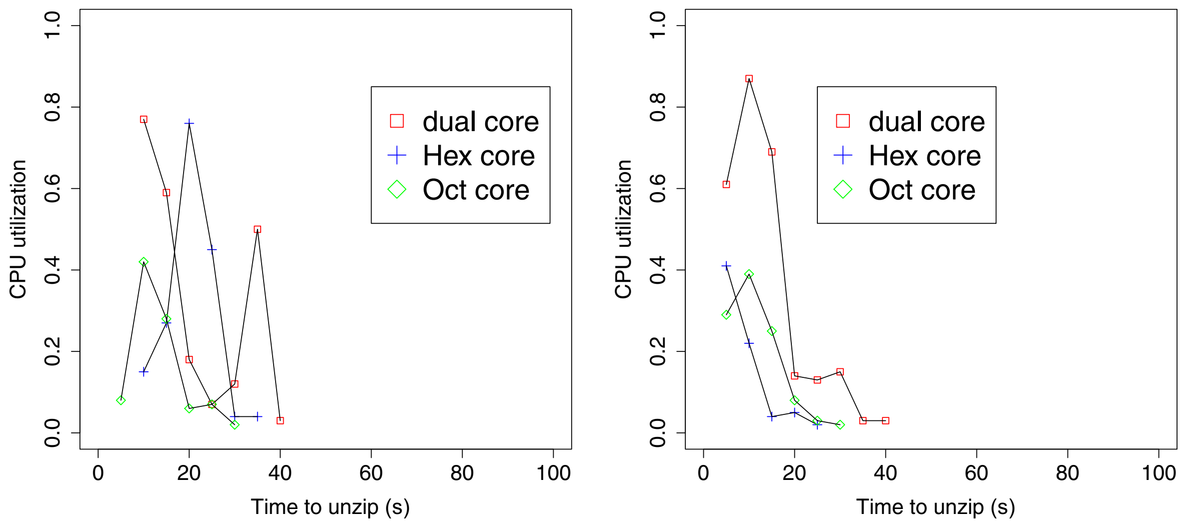
Figure 13 depicts the time spent in decompressing three files using dual-, hex-, and Oct-core processors. Again, the results confirm a strong correlation between the CPU cores and the time to unzip. Please refer to Table 13 for the time required to decompress the small, medium, and large files.

TABLE 11 Maximum CPU utilization for unzip

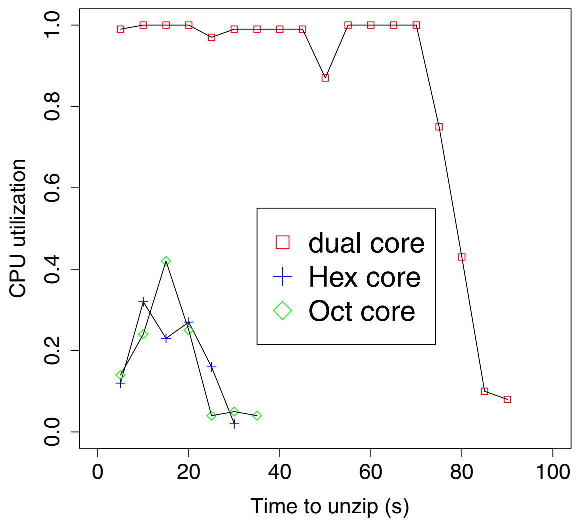
File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
Small	78%	76%	43%
Medium	85%	42%	40%
Large	100%	35%	42%

TABLE 12 Average CPU utilization for unzip

File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
Small	19%	21%	10%
Medium	17%	15%	18%
Large	98%	21%	16%



(A) CPU utilization to unzip a small size file (B) CPU utilization to unzip a medium size file



(C) CPU utilization to unzip a large size file

FIGURE 12 CPU Utilization to unzip a small, medium, and large size file. The small, medium, and large CPUs exhibit their behavior to unzip heterogeneous files

Equation (8)—an I/O-utilization model for the unzip benchmark—can be applied to estimate disk usage of a virtual machine running the benchmark. The model exhibits a strong fit because its *R* squared value is 0.37776. Here CPU cores (i.e., N_{CPU}) are not significant to evaluate the disk utilization, which asserts our theory of correlation.

$$\begin{aligned} U_{disk} = & -0.035963 \times U_{CPU} + 0.185791 \times N_{CPU} \\ & + 0.950243 \times U_{memory} + -0.018198 \times S_{disk}. \end{aligned} \tag{8}$$

We leverage Equation (9) as a linear model to derive CPU utilization of virtual machines, where CPU cores (N_{CPU}), memory size(S_{memory}), and disk utilization (U_{Disk}) are the three parameters. With the *R* squared value being 0.3877, the model has a goodness of fit and significance. Disk size (i.e., S_{disk}) in this model is insignificant with respective of evaluating CPU utilization.

$$\begin{aligned} U_{CPU} = & -0.108568 \times N_{CPU} + 1.376874 \times U_{memory} \\ & + 0.438104 \times U_{Disk} + 0.008921 \times S_{Disk}. \end{aligned} \tag{9}$$

4.5 | Installation

Installation is a benchmark to resemble CPU- and memory-intensive workload. In this experiment, we install various applications (like games, web browsers, software, etc.) of different sizes in order to make our model robust. Though the I/O load of Installation is fairly light, we build a model to

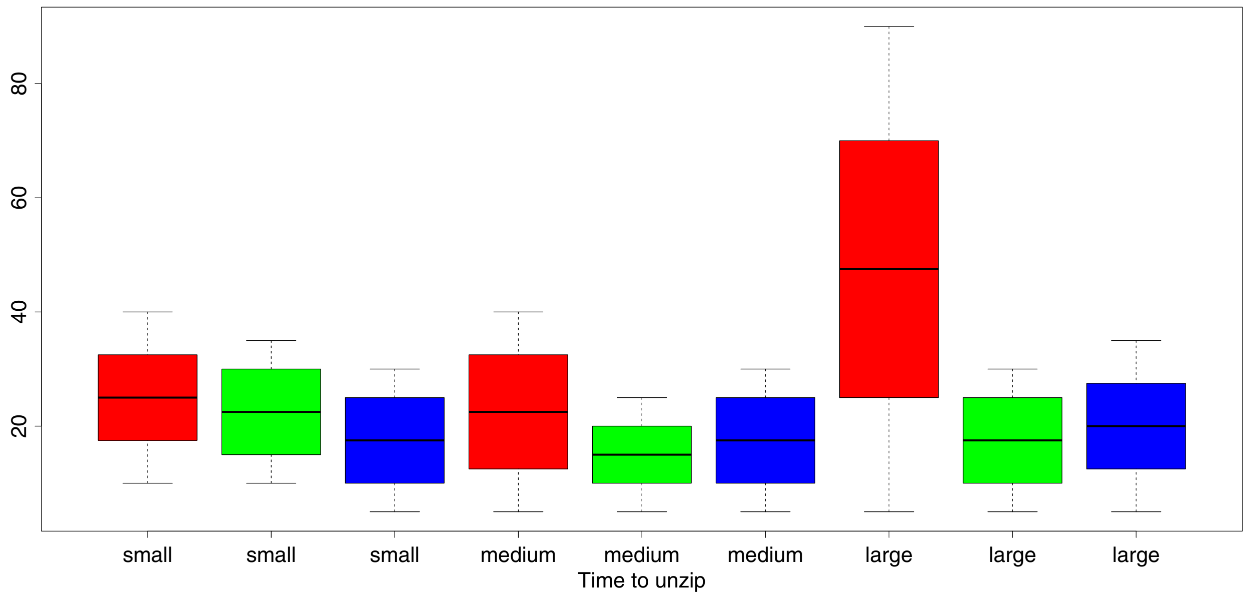


FIGURE 13 The image orchestrates the time taken to decompress files of the small, medium, and large size by heterogeneous processors. The first group of three bar plots represents the time to unzip small-size files by a dual-, hex-, and oct- core processor. The second group of three bar plots represents the time to unzip medium size files by the same heterogeneous processors. The third group of three bar plots represents the time to decompress large-size files by the same group of processors

TABLE 13 Time to decompress

File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
Small	40 s	37 s	30 s
Medium	42 s	35 s	37 s
Large	90 s	38 s	40 s

project the CPU and I/O behaviors of this benchmark with the data from the experiment we conduct. Though we used multiples files to install and uninstall, for simplicity we project the I/O and CPU behavior for installing and uninstalling a small, medium, and large file. Figure 14 unravels the CPU behavior for installation running on the virtual machines. We focus on CPU behaviors for small, medium, and large files (see also Figure 15). Again we summarize the detailed usage (i.e., the maximum and average CPU usage) in Tables 14 and 15.

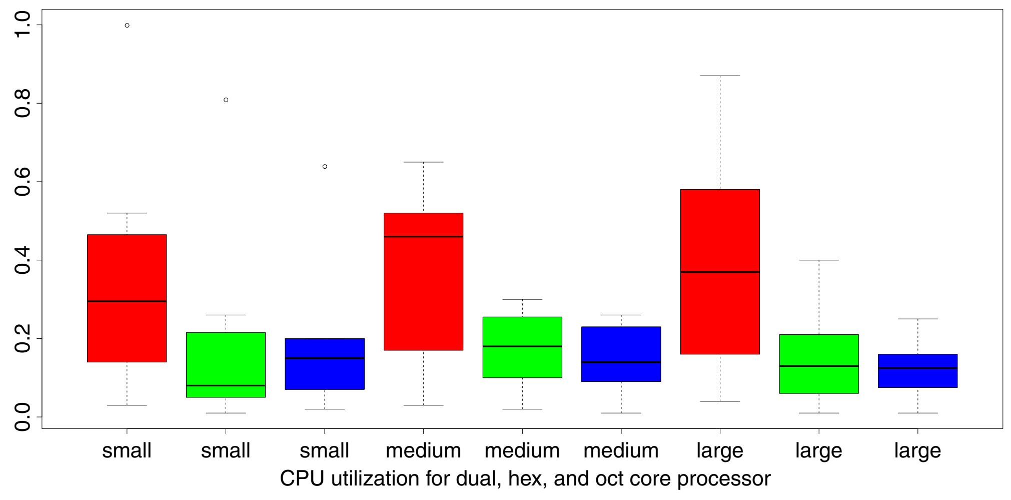
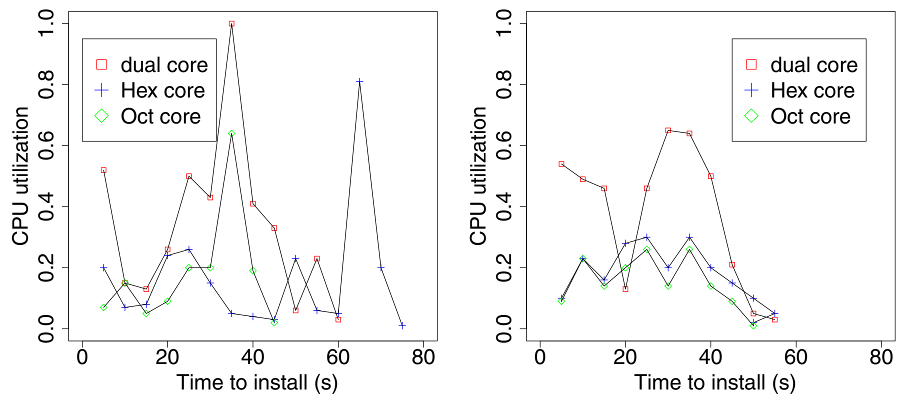
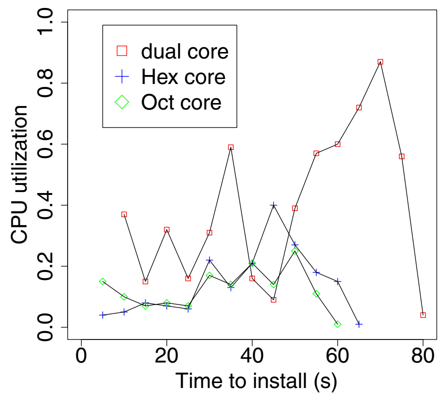


FIGURE 14 CPU utilization of the benchmarks install for the heterogeneous CPUs, in which small, medium, and large files are installed. The first group of three bar plots represents the CPU utilization for installing small files by a dual-, hex-, and oct- core processor. The second group of three bar plots represents the CPU utilization for installing medium size files by the same group of heterogeneous processors. The third group of three bar plots represents the CPU utilization for installing large-size files by the same group of heterogeneous processors



(A) CPU utilization to install a small size file (B) CPU utilization to install a medium size file



(C) CPU utilization to install a large size file

FIGURE 15 CPU Utilization to install a small, medium, and large size file. This image depicts more the number of cores of CPU more the utilization

TABLE 14 Maximum CPU utilization for install

File size	CPU 1 (2 cores)	CPU 2 (6 cores)	CPU 3 (8 cores)
Small	50%	22%	18%
Medium	65%	35%	30%
Large	85%	42%	37%

TABLE 15 Average CPU utilization for install

File size	CPU 1 (2 cores)	CPU 2 (6 cores)	CPU 3 (8 cores)
Small	25%	30%	10%
Medium	9%	37%	36%
Large	50%	25%	40%

Figure 14 depicts the CPU behaviors for dual-, hex-, and octa-core processors installing programs with various file sizes. Next, we present the I/O utilization graphically as well as in tabular format. Figure 15 displays the detailed profiling results, which unveil that the CPU utilization is inversely proportional to the number of cores (Tables 16 and 17).

Next, we investigate the correlation among CPU cores, CPU usage, disk size, and disk utilization. We observe that CPU cores have a very strong relation with CPU Usage (−0.5605326), whereas disk usage has a medium correlation with disk size (0.2547766). Therefore, the evidence indicates that *Installation* is a CPU-intensive process.

Figures 16 and 17 portrays the I/O utilization for benchmark installation.

TABLE 16 Mean I/O utilization for install

File size	Disk 1 (35 GB)	Disk 2 (20 GB)	Disk 3 (30 GB)
Small	42%	98%	22%
Medium	18%	65%	78%
Large	98%	85%	40%

TABLE 17 Maximum utilization for install

File size	Disk 1 (35 GB)	Disk 2 (20 GB)	Disk 3 (30 GB)
Small	40%	20%	22%
Medium	20%	16%	25%
Large	50%	29%	39%

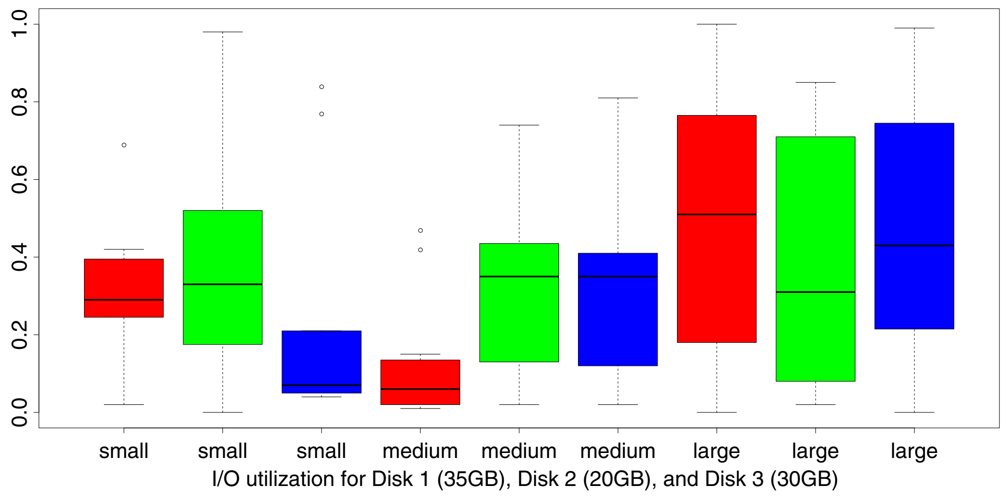


FIGURE 16 I/O utilization of the benchmark for installing small, medium, and large files stored in the heterogeneous disks. The first group of three bars plots the I/O utilization for installing small files on a virtual disk of size 35 GB (red color), 20 GB (blue color), and 30 GB (green color). The second group of three bars represents the I/O utilization for installing medium size files on the same virtual disks consecutively. The third group of three bars represents the I/O utilization for installing large files on the same virtual disks

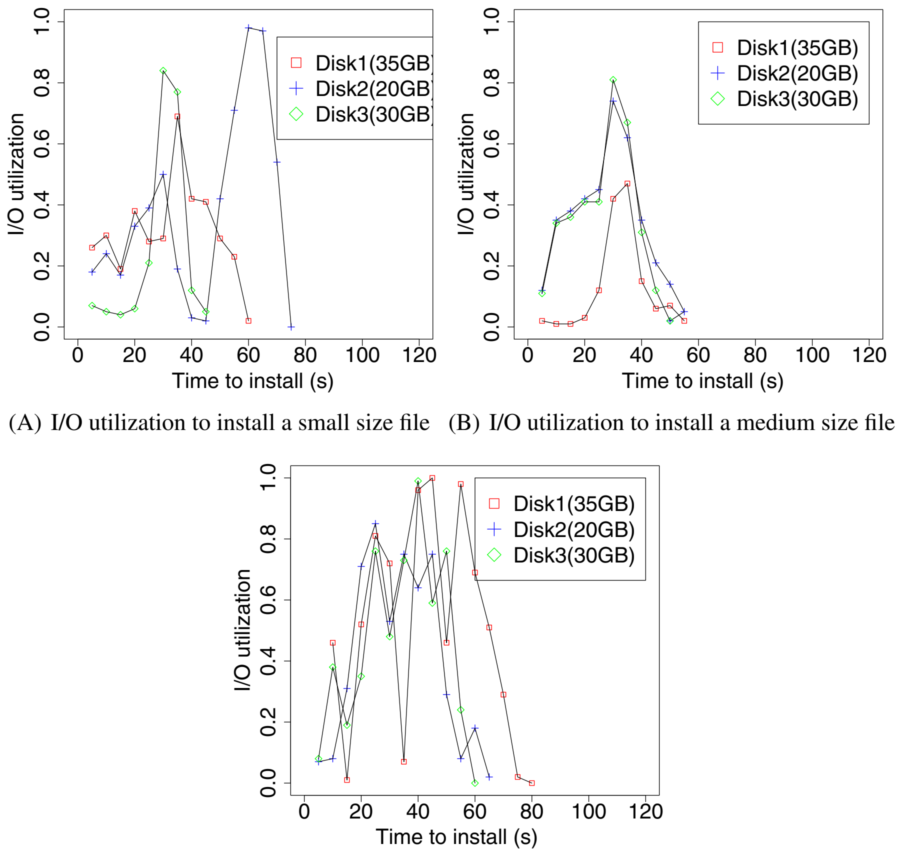
We plot in Figure 18 the time taken to install small, medium, and large programs with heterogeneous virtual machines. The average and maximum times are listed in the tabular format in Table 18.

Now we build an I/O-utilization model for the install benchmark. Equation (10) is a linear model that leverages disk size (i.e., S_{disk}) and main memory size (i.e., S_{memory}) to deduce disk usage of a virtual machine running independently on a cloud. Nevertheless, S_{disk} and S_{memory} are two insignificant parameters to predict disk usage. The model has an Rsquared value of 0.05375, meaning that the model has a weak fit. Only CPU usage U_{CPU} is significant in this model.

$$\begin{aligned} U_{disk} = \; & 0.341039 \times U_{CPU} - 0.044867 \times S_{memory} \\ & + -0.006595 \times S_{disk}. \end{aligned} \tag{10}$$

We move forward to construct another model projecting the CPU utilization of virtual machines for the *install* benchmark. Equation (11) is a linear model to deduce CPU utilization for virtual machines concurrently running on a cloud. In this model, all the parameters, including CPU cores(N_{CPU}), disk size(S_{Disk}), and disk utilization (U_{Disk}), are the most significant parameters. The model has an R squared value of 0.3808, indicating that the model has a goodness of fit and strong significance.

$$\begin{aligned} U_{CPU} = & -0.046087 \times N_{CPU} + 0.197513 \times U_{Disk} \\ & + 0.009046 \times S_{Disk}. \end{aligned} \tag{11}$$



(A) I/O utilization to install a small size file (B) I/O utilization to install a medium size file

(C) I/O utilization to install a large size file

FIGURE 17 I/O Utilization for installation of a small, medium, and large size file

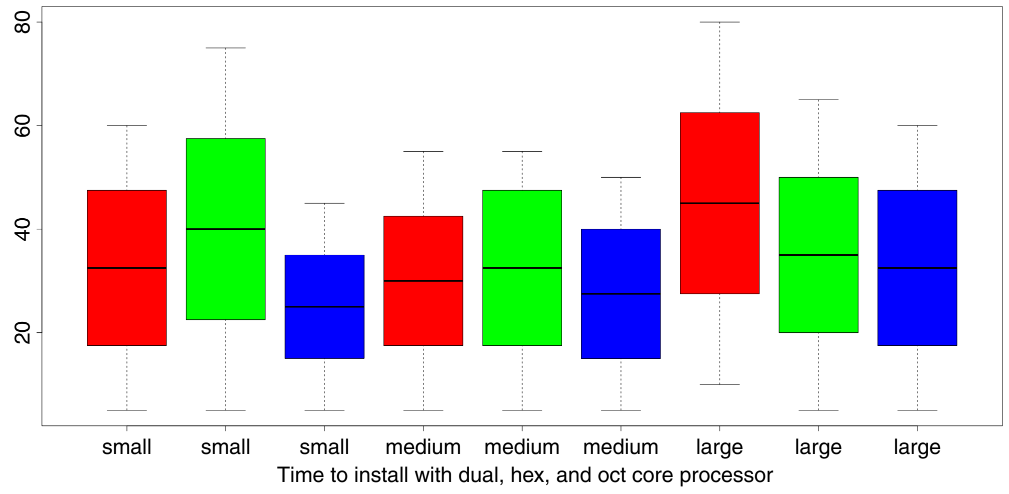


FIGURE 18 The image orchestrates the time taken to install files of small, medium, and large size by heterogeneous processors. The first group of three bar plots represents the time to install small-size files by a dual-, hex-, and oct- core processor. The second group of three bar plots represents the time to install medium size files by the same heterogeneous processors. The third group of three bar plots represents the time to install large-size files by the same group of processors

TABLE 18 Time to install

File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
Small	60 s	70 s	45 s
Medium	52 s	57 s	52 s
Large	79 s	70 s	62 s

4.6 | Uninstallation

To evaluate whether *uninstallation* is CPU- or I/O- intensive, we perform multiple experiments running the benchmark in the same setup. The experimental results unveil that uninstalling a file is CPU- as well as I/O- intensive. The correlation between the CPU cores and CPU usage is −0.62297, whereas the correlation between I/O configuration and the CPU utilization is 0.3240454. Again the correlation between CPU cores and the disk usage is −0.238729 and that between I/o configuration and I/O usage is 0.42864. Instead of plotting all the diagrams, we only illustrate significant ones from the perspectives of I/O and CPU behaviors of the virtual machines running on the cloud.

Figure 19 depicts the CPU utilization for the virtual processors uninstalling three files of various sizes. Figure 19 demonstrates that increasing the number of cores boosts performance. The figure offers evidence for a strong correlation between the number of CPU cores and CPU usage. Tables 19 and 20 provides the average and maximum CPU utilization for the virtual processors.

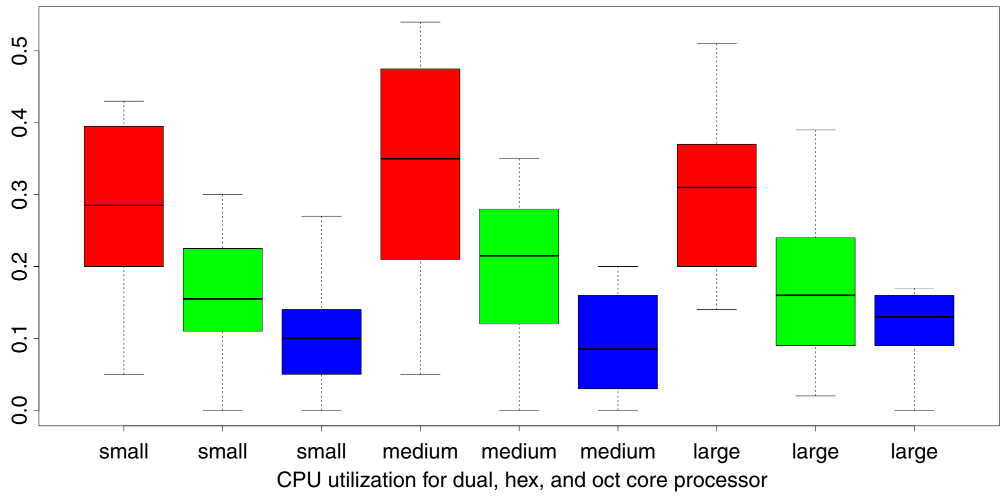


FIGURE 19 CPU utilization of the benchmark uninstall for the heterogeneous CPUs, in which small, medium, and large files are uninstalled. The first group of three bar plots represents the CPU utilization for uninstalling small files by a dual-, hex-, and oct- core processor. The second group of three bar plots represents the CPU utilization for uninstalling medium size files by the same group of heterogeneous processors. The third group of three bar plots represents the CPU utilization for uninstalling large-size files by the same group of heterogeneous processors

TABLE 19 Maximum CPU utilization for uninstall

File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
Small	45%	32%	28%
Medium	55%	35%	22%
Large	52%	40%	15%

TABLE 20 Average CPU utilization for uninstall

File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
Small	29%	15%	9%
Medium	32%	22%	7%
Large	30%	18%	12%

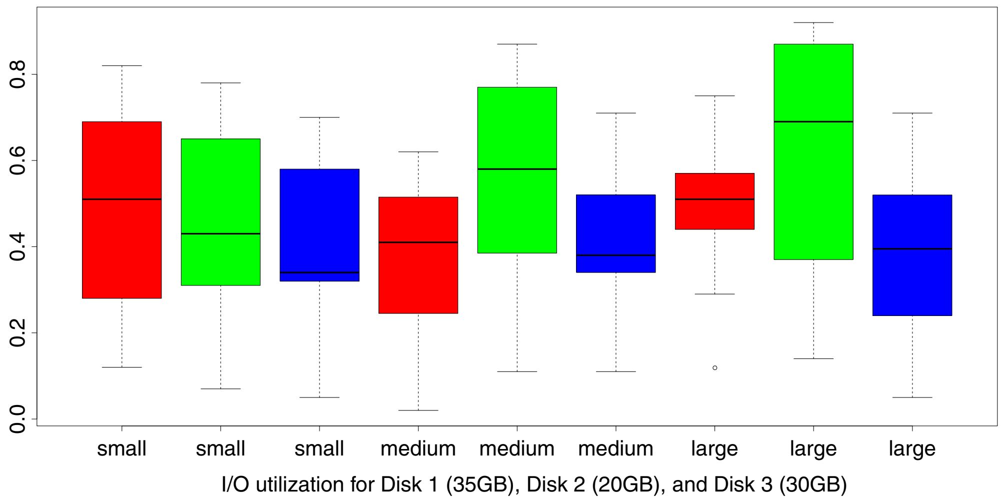


FIGURE 20 I/O utilization of the uninstall benchmark for uninstalling small, medium, and large files stored in the heterogeneous disks. The first group of three bars plots the I/O utilization for uninstalling small files on a virtual disk of size 35 GB (red color), 20 GB (blue color), and 30 GB (green color). The second group of three bars represents the I/O utilization for uninstalling medium size files on the same virtual disks consecutively. The third group of three bars represents the I/O utilization for uninstalling large files on the same virtual disks

TABLE 21 Mean I/O utilization for uninstall

File size	Disk 1 (35 GB)	Disk 2 (20 GB)	Disk 3 (30 GB)
Small	50%	42%	38%
Medium	40%	57%	40%
Large	50%	60%	43%

Figure 20 reveals the I/O utilization for the virtual disks uninstalling three small-, medium-, and large-sized files. The results imply that disk size, as well as the number of CPU cores, play a vital role in the benchmark's I/O behaviors. Tables 21 and 22 summarizes the average and maximum I/O utilization of the virtual disks.

Next, we plot the time taken to uninstall small, medium, and large files stored on the heterogeneous Virtual machines. We present the average and maximum time taken to uninstall the files in the tabular format (see also Table 23). Figure 21 represents the required time to install files via these VMs.

TABLE 22 Maximum I/O utilization for uninstall

File size	Disk 1 (35 GB)	Disk 2 (20 GB)	Disk 3 (30 GB)
Small	82%	78%	75%
Medium	65%	82%	70%
Large	75%	90%	72%

TABLE 23 Time to uninstall

File size	CPU 1 (2 core)	CPU 2 (6 core)	CPU 3 (8 core)
Small	60 s	59 s	52 s
Medium	55 s	53 s	50 s
Large	80 s	68 s	52 s

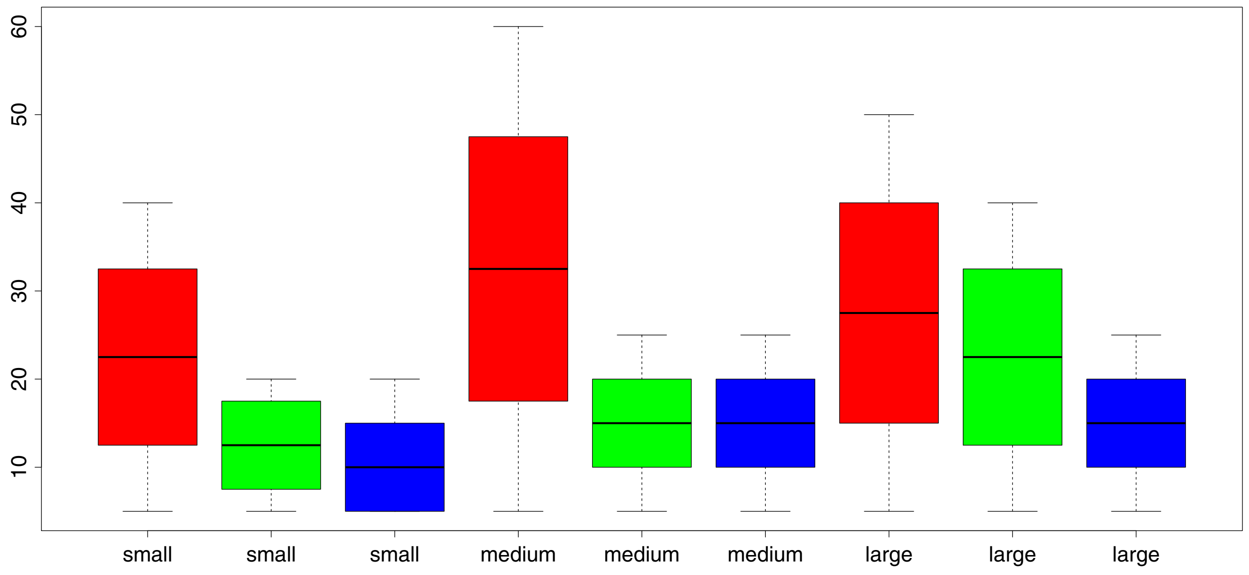


FIGURE 21 The image orchestrates the time taken to uninstall files of the small, medium, and large size by heterogeneous processors. The first group of three bar plots represents the time to uninstall small-size files by a dual-, hex-, and oct- core processor. The second group of three bar plots represents the time to uninstall medium size files by the same heterogeneous processors. The third group of three bar plots represents the time to uninstall large-size files by the same group of processors

We design two models (see Equations (12) and (13)) for the CPU and I/O utilization of the *uninstallation* benchmark. Equation (12) elaborates that CPU cores and disk usage are two significant parameters, whereas the disk size of the disk is insignificant. Equation (13) indicates that the model has two significant parameters, namely, the number of cores and disk utilization.

$$\begin{aligned} U_{disk} = {} & 0.020465 \times N_{CPU} - 0.878699 \times U_{CPU} \\ & + -0.011085 \times S_{disk}, \end{aligned} \tag{12}$$

$$\begin{aligned} U_{CPU} = {} & -0.035990 \times N_{CPU} + 0.305639 \times U_{Disk} \\ & + 0.003026 \times S_{Disk}. \end{aligned} \tag{13}$$

5 | CASE STUDIES

While conducting our experiments, we come across two intriguing case studies. In the first case study (see Section 5.1), we shed light on the performance discrepancy between split disks and monolithic disks. The second case study (see Section 5.2) is focused on the significance of memory usage in the performance models. The detailed study is described in the following subsections:

5.1 | Split disk and monolithic disk

There is two way of configuring virtual disks—split disks and monolithic disks. To dig into the comparison of the two disk configurations, we set up two equal-sized virtual disks, one of which is a split disk and another is a monolithic one. An interesting observation is the split disk outperforms the monolithic counterpart in terms of I/O processing time. This finding is universally consistent across all the tested benchmarks under various settings.

A split disk is superior to a monolithic disk thanks to virtual segmentation, where multiple segments assigned to different tasks can concurrently running to expand the I/O throughput. A split disk leverages the multiple segments to lower the disk utilization. Unlike split disks, monolithic disks stay away from the advantage of concurrent segments. Figure 22 portrays the performance comparisons between the split and monolithic virtual disks.

It is worth noting that for some benchmarks (e.g., *install*), the performance gap between the split and monolithic disks is diminished due to the nature of CPU-intensive workload. Therefore, we experiment with split disks to provide real-world experience to the power managers. Also, split disks expedite the I/O operations with concurrency leading to a lesser experiment time.

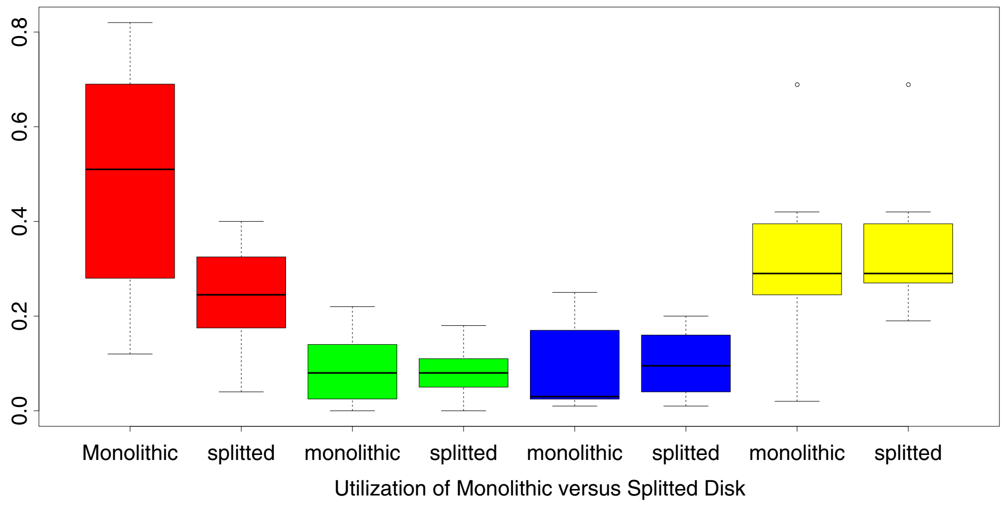


FIGURE 22 The I/O utilization for a monolithic disk and a split disk under the same disk size for *uninstall*, *zip*, *unzip*, and *install*. The red, green, blue, and yellow box plots represents the benchmarks uninstall, zip, unzip, and install

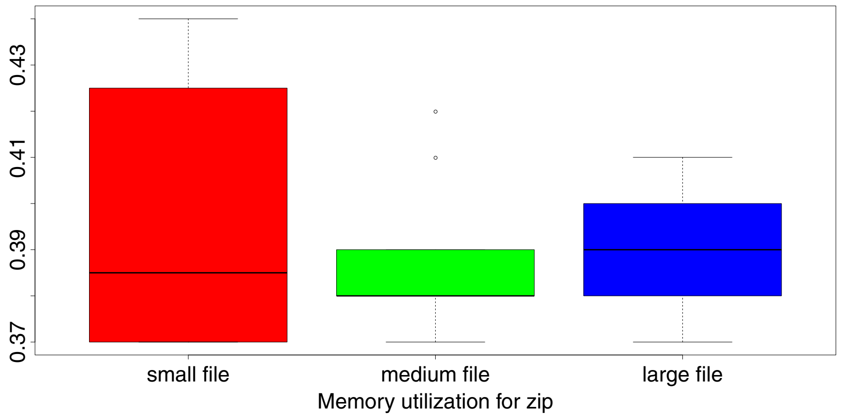


FIGURE 23 The memory utilization to compress small, medium, and large size file. The red, green, and blue box plots represents the small, medium, and large size file

5.2 | Memory utilization

Though memory is an important parameter, since we were keener to project CPU and I/O usage in this article, we did not experiment much with the size of the memory. In this study, we figured that the memory configuration is an insignificant parameter to build a utilization model. Though memory plays a significant role for some benchmarks like *installation*, it has a weak correlation with I/O or CPU usage. For example, in the case of the *zip* benchmark, the correlation between memory and I/O utilization is 0.0011097 and that with CPU utilization is 0.000965. Figure 23 displays the memory usage for the *zip* benchmark. For a majority of the tested benchmarks, the result trends are quite similar. Therefore, we conclude that the benchmarks evaluated in this study impose a nonmemory-intensive workload. Therefore, we decide not to make the memory usage model in this study. Hopefully, in the future, we will further work with memory-intensive benchmark applications to model the memory usage for those specific benchmarks.

6 | CONCLUSIONS AND FUTURE SCOPE

Building linear models for I/O- and CPU-intensive applications have been an interesting journey for us. Through this study, we learned various dynamics of cloud platforms, virtual resource usage, and modeling resource utilization in the context of heterogeneous virtual machines. Starting with the effect of cache coherence on resource usage to determining the resource usage for a set of virtual machines, it is time-consuming to build predictive models of resource utilization for a set of benchmarks running on virtual machines. The strength of our research sparks a strong foundation in the realm of predicting models for resource utilization in the cloud infrastructure.

Future research directions are threefold. First, we plan to progress our horizon of research into taking full benefits of split disks to optimize the overall performance of I/O-intensive applications running on virtual machines. Second, we intend to expand our exposure toward novel machine learning models (e.g., *random forest* and *decision tree*) to enhance the accuracy and efficiency of the models constructed in this study. Third, we will explore a handful of memory-intensive applications on virtual machines from the perspective of performance modeling.

DATA AVAILABILITY STATEMENT

Data available on request due to privacy/ethical restrictions

ORCID

Tathagata Bhattacharya https://orcid.org/0000-0003-3925-7009
Xiaopu Peng https://orcid.org/0000-0003-4995-7746

REFERENCES

1. Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr Comput Pract Exper*. 2012;24(13):1397-1420.
2. Dikaiakos MD, Katsaros D, Mehra P, Pallis G, Vakali A. Cloud computing: distributed internet computing for IT and scientific research. *IEEE Internet Comput*. 2009;13(5):1–2.
3. Nagpal AR, Shukla H, Grobman I, Babu SBR, Ramesh A. Storage-aware dynamic placement of virtual machines; 2018. US Patent App. 15/352,495.
4. Mishra M, Das A, Kulkarni P, Sahoo A. Dynamic resource management using virtual machine migrations. *IEEE Commun Mag*. 2012;50(9):34-40.
5. Kansal A, Zhao F, Liu J, Kothari N, Bhattacharya AA. Virtual machine power metering and provisioning. Proceedings of the 1st ACM symposium on Cloud computing; 2010:39-50; ACM, New York, NY.
6. Shaw R, Howley E, Barrett E. An energy efficient anti-correlated virtual machine placement algorithm using resource usage predictions. *Simul Model Pract Theory*. 2019;93:322–242.
7. Rahmanian AA, Ghobaei-Arani M, Tofighy S. A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment. *Futur Gener Comput Syst*. 2018;79:54-71.
8. Gul B, Khan IA, Mustafa S, et al. CPU and RAM energy-based SLA-aware workload consolidation techniques for clouds. *IEEE Access*. 2020;8:62990-63003.
9. Baskaran N, Eswari R. CPU-memory aware VM consolidation for cloud data centers. *Scalable Comput Pract Exper*. 2020;21(2):159-172.
10. Kerr A, Diamos G, Yalamanchili S. Modeling GPU-CPU workloads and systems. Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units 2010:31-42; ACM, New York, NY.
11. Huber N, van Quast M, Hauck M, Kounev S. Evaluating and modeling virtualization performance overhead for cloud environments. *CLOSER*. 2011;11:563-573.
12. Feitelson DG. Memory usage in the LANL CM-5 workload. Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing; 1997:78-94; Springer, Berlin, Heidelberg.
13. Yang CT, Wan TY. Implementation of an energy saving cloud infrastructure with virtual machine power usage monitoring and live migration on OpenStack. *Computing*. 2020;102(6):1547-1566.
14. MuthuPandi K, Somasundaram K. Design of energy efficient datacenter with optimized virtual infrastructure. AIP Conference Proceedings; Vol. 2271, 2020:030014; AIP Publishing LLC.
15. Bui KT, Ho HD, Pham TV, Tran HC. Virtual machines migration game approach for multi-tier application in infrastructure as a service cloud computing. *IET Netw*. 2020;9(6):326-337.
16. Duan H, Chen C, Min G, Wu Y. Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems. *Futur Gener Comput Syst*. 2017;74:142-150.
17. Xiao Z, Song W, Chen Q, others . Dynamic resource allocation using virtual machines for cloud computing environment.. *IEEE Trans Parallel Distrib Syst* 2013; 24(6): 1107–1117.
18. Li X, Qian Z, Lu S, Wu J. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. *Math Comput Model*. 2013;58(5-6):1222-1235.
19. Nguyen TH, Di Francesco M, Yla-Jaaski A. Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers. *IEEE Trans Serv Comput*. 2017;13(1):186–199.
20. Khosravi A, Buyya R. Short-term prediction model to maximize renewable energy usage in cloud data centers. *Sustainable Cloud and Energy Services*. Springer; 2018:203-218.
21. Shafer J. I/O virtualization bottlenecks in cloud computing today. Proceedings of the 2nd Conference on I/O Virtualization; 2010:5; USENIX Association.
22. Jung J, Park J, Kim S, Heo M, Hong J. A virtual CPU scheduling model for I/O performance in paravirtualized environments. *RACS '17*. ACM; 2017:281-286.
23. Hao M, Zhang W, Zhang Y, Snir M, Yang LT. Automatic generation of benchmarks for I/O-intensive parallel applications. *J Parallel Distrib Comput*. 2019;124:1-13.
24. Nawaz H, Juve G, Da Silva RF, Deelman E. Performance analysis of an I/O-intensive workflow executing on Google cloud and Amazon web services. Proceedings of the 2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW); 2016:535-544; IEEE.
25. Qin X, Jiang H, Zhu Y, Swanson DR. Improving the performance of I/O-intensive applications on clusters of workstations. *Clust Comput*. 2006;9(3):297-311.
26. Kurazumi S, Tsumura T, Saito S, Matsuo H. Dynamic processing slots scheduling for I/O intensive jobs of Hadoop MapReduce. Proceedings of the 2012 Third International Conference on Networking and Computing; 2012:288-292; IEEE.

27. Segall RS, Cook JS, Niu G. Overview of big data-intensive storage and its technologies for cloud and fog computing. *Int J Fog Comput (IJFC)*. 2019;2(1):74-113.
28. Gordon A, Amit N, Har'El N, et al. ELI: bare-metal performance for I/O virtualization. *ACM SIGPLAN Not*. 2012;47(4):411-422.
29. Pu X, Liu L, Mei Y, Sivathanu S, Koh Y, Pu C. Understanding performance interference of i/o workload in virtualized cloud environments. Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing; 2010:51-58.
30. Zhang X, Davis K, Jiang S. QoS support for end users of I/O-intensive applications using shared storage systems. Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis; 2011:18; ACM.
31. Zakarya M, Gillam L, Khan AA, Rahman IU. Perficientcloudsim: a tool to simulate large-scale computation in heterogeneous clouds. *J Supercomput*. 2020;1-55.
32. de Silva RF, Casanova H, Orgerie AC, Tanaka R, Deelman E, Suter F. Characterizing, modeling, and accurately simulating power and energy consumption of I/O-intensive scientific workflows. *J Comput Sci*. 2020;101157.
33. Baghban H, Chou J, Hsu CH, Chung YC. Modeling of resource granularity and utilization with virtual machine splitting. Proceedings of the 2016 IEEE 14th International Conference on Dependable, Autonomic and Secure Computing, 14th International Conference on Pervasive Intelligence and Computing, 2nd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech); 2016:769-772; IEEE.
34. Bohra AEH, Chaudhary V. VMeter: power modelling for virtualized clouds. Proceedings of the 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Ph.D forum (ipdpsw); 2010:1-8; IEEE.
35. Yaseen FA, Alhumaima RS, Al-Zubaedi W, Al-Raweshidy HS. Modelling the power cost and trade-off of live migration the virtual machines in cloud-radio access networks. Proceedings of the 2017 9th Computer Science and Electronic Engineering; 2017:122-127; IEEE.
36. Alzamil IAM. *Energy-Aware Profiling and Prediction Modelling of Virtual Machines in Cloud Computing Environments*. PhD thesis. University of Leeds; 2017.
37. Briongos S, Malagón P, Risco JL, Moya JM. Building accurate models to determine the current CPU utilization of a host within a virtual machine allocated on it. Proceedings of the Summer Simulation Multi-Conference; 2017:1-12.
38. Liaqat M, Naveed A, Ali RL, Shuja J, Ko KM. Characterizing dynamic load balancing in cloud environments using virtual machine deployment models. *IEEE Access*. 2019;7:145767-145776.
39. Medel V, Rana O, Bañares JÁ, Arronategui U. Modelling performance & resource management in kubernetes. Proceedings of the 9th International Conference on Utility and Cloud Computing; 2016:257-262.
40. Lettieri G, Maffione V, Rizzo L. A study of I/O performance of virtual machines. *Comput J*. 2018;61(6):808-831.
41. de Silva RF, Orgerie AC, Casanova H, Tanaka R, Deelman E, Suter F. Accurately simulating energy consumption of I/O-intensive scientific workflows. Proceedings of the International Conference on Computational Science; 2019:138-152.
42. Elsakhawy M, Bauer M. An investigation into the usage-trends of Canada's research computing clouds. Proceedings of the 2019 IEEE International Conference on Smart Cloud; 2019:1-6; IEEE.
43. Jin C, Bai X, Yang C, Mao W, Xu X. A review of power consumption models of servers in data centers. *Appl Energy*. 2020;265:114806.
44. Chou YL, Yang JM, Wu CH. An energy-aware scheduling algorithm under maximum power consumption constraints. *J Manuf Syst*. 2020;57:182-197.
45. Liang Y, Hu Z, Li K. Power consumption model based on feature selection and deep learning in cloud computing scenarios. *IET Commun*. 2020.
46. Jung W, Zhao H, Sun M, Zhou G. IoT botnet detection via power consumption modeling. *Smart Health*. 2020;15:100103.

How to cite this article: Bhattacharya T, Peng X, Mao J, et al. Performance modeling for I/O-intensive applications on virtual machines. *Concurrency Computat Pract Exper*. 2022;34(10):e6823. doi: 10.1002/cpe.6823