Analysis of Workloads for Cloud Infrastructure Capacity Planning

Eva Patel and Dharmender Singh Kushwaha

Abstract Workload analysis and characterization are the first steps toward effective cloud infrastructure capacity planning. Identifying workload patterns based on resource utilization not only enables informed decisions about mapping of current request to available capacity, but also serves as a meaningful indicator for future resource requirements. Of paramount concern is the optimal utilization of data center server capacity, i.e., the CPU, I/O, and memory. The compute capacity of modern servers can be further harnessed by optimal utilization of individual CPU cores. A precise CPU core-level usage monitoring and provisioning can lead to cumulative benefits of optimal CPU utilization, efficient VM placement, reduced VM migrations, and energy efficiency through lower power consumption. In this paper, we make a preliminary analysis of usage patterns of CPU cores in the case of CPU- and memory-intensive workloads on an experimental cloud setup in our laboratory. The aim is to make a comparative analysis of the utilization of individual CPU cores with that of aggregated CPU usage to explore the feasibility of incorporating a fine-grained usage detail for resource scheduling and VM provisioning. Initial experiments reveal observable differences between the utilization of individual CPU cores and that reported by aggregate CPU usage. Usage difference ranges from 1 to 29% below and between 4 and 20% above the aggregate. Incorporating such finer details can leverage the vast compute capacity of multicore servers and effective power usage.

Keywords Workload analysis · Workload characterization · Capacity planning
Server capacity planning · Workload intensity · Job inter-arrival patterns
Multicore scheduling

E. Patel (✉) · D. S. Kushwaha
Department of Computer Science and Engineering, MNNIT Allahabad, Allahabad, UP 211004, India
e-mail: evapatel08@gmail.com

D. S. Kushwaha
e-mail: dsk@mnnit.ac.in

L. C. Jain et al. (eds.), *Data and Communication Networks*, Advances in Intelligent Systems and Computing 847, https://doi.org/10.1007/978-981-13-2254-9_4

1 Introduction

Scalability, agility, guaranteed quality of service, the illusion of infinite computational resources, and access to high-end computing infrastructure on a pay-as-per-usage basis, without the painstaking exercise of setting up a private IT infrastructure, have led to the relentless adoption of cloud computing solutions for a range of computing requirements—business intelligence, engineering design, scientific computing, social networking, and content delivery. These services are hosted on data centers that house hundreds of thousands of servers and supporting infrastructure. For instance, Amazon, which made the IT infrastructure available via the Internet in 2006, a technology termed cloud computing [1], has around 450,000 servers spread across seven different data centers around the world [2] that provide the infrastructure services—compute, storage, and networking. Google data center caters to the diverse requests of its cloud users through 900,000 plus servers hosted in data centers in 16 different geographical locations [2]. The envisioned growth in computing and storage requirements, diversity of services, and emergence of innovative server technologies that also entail growing power consumption and increasing carbon footprints calls for adoption of capacity planning of cloud data centers as a continual process.

Data center capacity planning aims to determine a system configuration for given service requests that comply with the service-level agreement (SLA) [3] with minimal over- or under-provisioning of resources. The first step to effective capacity planning is the analysis of resource usage patterns of cloud workloads to identify associated challenges and their implications on resource provisioning and overall system performance. A vast body of literature records design of scheduling algorithms with CPU utilization as the primary performance metric [4–10]. With several CPU cores crammed into a single socket, and multisocket servers with massive compute capacity, the use of individual core usage as decision parameter for VM scheduling is one possible approach to effective utilization of server compute capacity.

Driven by the observation that aggregate CPU usage is not always reflective of the usage of individual CPU cores, we conduct a preliminary analysis of utilization of CPU cores in case of CPU-intensive and memory-intensive workloads. Through this study, the aim is to achieve following objectives:

1. Effective utilization of compute capacity of all the cores of a multicore server by considering individual CPU core capacity rather than the aggregate CPU usage.
2. Identify and segregate cores that are underutilized, to schedule to incoming workload thereby reducing energy wastage.
3. Mitigate situations of overloaded CPU cores to maintain desired performance and sustainable energy consumption levels.

The rest of this paper is organized as follows. Section 2 gives a brief overview of related work on workload analysis and characterization. Section 3 discusses some background concepts and findings from past research that reveal suboptimal use of CPU cores. Experimental setup and results are discussed in Sect. 4. Section 5 concludes with directions for future work.

2 Related Work

Resource requirements of applications deployed on the cloud depend on their processing needs. Consequently, each workload has different challenges which guide the analysis and characterization process.

Based on behavioral characteristics, Mahambre et al. [4] identify five cloud workload patterns, i.e., periodicity, threshold, relationship, variability, and image similarity, for capacity planning decisions such as migration, re-provisioning, load balancing, and initial placement of the workloads. Moreno et al. [5] present a comprehensive analysis of workload characteristics to study heterogeneity due to user behavior and task resource usage. Their study shows that users are responsible the most for introducing heterogeneity in the cloud as compared to task diversity which is a consequence of diverse service requests due to the illusion of infinite resources to the users. Authors in [6] characterize the interactive behavior of Web applications in terms of number of instructions executed per second, and CPU, memory and disk utilization. Peng et al. [7] develop a classification scheme based on computational needs of the workloads for compute-intensive, I/O-intensive, and network-intensive applications. Zhang et al. [8] consider both the heterogeneity of machine hardware and workload diversity for dynamic capacity provisioning. Cloud infrastructure is heterogeneous due to the presence of machines from multiple generations, heterogeneous processor architectures and speed, and different memory and disk capacities. Singh and Chana [9] characterize workloads based on QoS requirements for different workload types such as scientific computing, online transaction processing, performance testing, and storage and backup services. Authors in [10] classify cloud workloads based on their functional characteristics, into six categories—scientific processing, Big Data application, OLTP, caching, streaming, and Web serving. The aim is to customize the resource requirements with the actual utilization of a specific workload for effective cloud monitoring.

From our study of the above work and the conclusions drawn by Lozi et al. [11], we infer that compute capacity of data center servers can be harnessed more effectively by considering per-core CPU usage rather than the aggregated utilization.

3 Related Concepts

3.1 Capacity Planning

Capacity planning is the process of determining computing infrastructure (hardware, software, and connection interface) and the floor area to house these components to cater to services for a future time period [12]. Data center capacity planning involves server capacity planning as well as network capacity planning. This work focusses on *server capacity planning* in which an IT department determines the amount of server hardware resources required to provide the desired levels of service for a given

workload mix for the least cost [12]. The central objective is to provision resources in compliance with SLA.

The notion of capacity planning as perceived by cloud actors depends on their corresponding role. National Institute of Standards and Technology (NIST) [13] defines five cloud actors: consumer, provider, auditor, broker, and carrier. Of interest for server capacity planning are the roles of cloud consumer and cloud provider. The main aim of a cloud provider is to provision resources in a way that maximizes returns on investment, whereas a cloud user is concerned with quality of service that would justify rental cost. These benefits can be fully realized through optimal utilization of compute capacity of server.

3.2 Workload Analysis and Characterization

Workload analysis involves a detailed investigation of workload features of interest, to identify their behavioral characteristics, intra- and inter-dependencies, and impact on system performance. *Workload characterization* is the process of precisely describing the system's global workload in terms of its main components. Workload analysis measures cloud services along different dimensions including infrastructure capacity planning, energy efficiency, performance, reliability, and security. Two fundamental issues with workload analysis specific to infrastructure capacity planning [3] are:

1. Workload arrival pattern
2. Workload intensity

Workload arrival pattern measures the rate at which jobs arrive for service. *Workload intensity* refers to the amount of work done over a unit of time. Job arrival patterns are highly unpredictable and exhibit different behavioral patterns—diurnal, seasonal, and flash crowd [14]. Additionally, workloads may have either steady resource demands or may manifest temporal patterns such as periodic, bursting, growing, and on-and-off. An effective server capacity planning scheme should ensure resource availability on demand and provisioning for required time duration.

Job Arrival Patterns as Non-homogeneous Poisson Process. Non-homogeneous Poisson process is one way to model real-world job arrival patterns. Formally, given the occurrence of events at a constant rate λ, over a period T, the job inter-arrival time can be modeled as a Poisson process with exponential distribution which is governed by Eq. (1) below:

$$P(N(t) = k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t}, \quad k = 0, 1, 2, \ldots \tag{1}$$

where the random variable $N(t)$ denotes the number of events that occur in t time units, P is the probability of occurrence of k events, and λ, known as the rate parameter, measures the number of events that occur per unit of time. Variability in job arrival

can be captured using non-homogeneous Poisson process in which the rate parameter λ changes with time.

3.3 Multicore Scheduling

Multicore processors are the consequence of unsustainable power consumption and heat dissipation resulting from increased CPU clock frequency for performance gains [15]. The layout of a typical quad-core processor chip is shown in Fig. 1. Each core has a separate Level 1 (L1) cache for data and instructions and a Level 2 (L2) cache that holds both instruction and data, which are private. L2 cache is either united or distributed. L2 cache is usually physically distributed with a united Level 3 (L3) cache. Some multicores have an off-chip Level 4 (L4) cache. Two central issues when scheduling on multicore servers are contentions due to access to shared resources and efficient utilization of individual cores. The vast processing capability of modern multicore processors can be tapped, by keeping all the cores busy to the maximum. This has been achieved by adapting the operating system scheduler for single-core systems to that of scheduling large number of cores.

The Linux Scheduler. Completely Fair Scheduler (CFS), introduced in October 2007, is the default process scheduler in Linux since kernel version 2.6.23 and supersedes the $O(n)$ scheduler and $O(1)$ scheduler. $O(n)$ scheduler was used in kernel versions from 2.4 to 2.6 and was replaced by $O(1)$ scheduler in 2001, due to non-scalability issues. The $O(1)$ scheduler maintains a constant scheduling time irrespective of the number of jobs in the system and uses average sleep time of a process to distinguish between interactive and non-interactive jobs. However, this

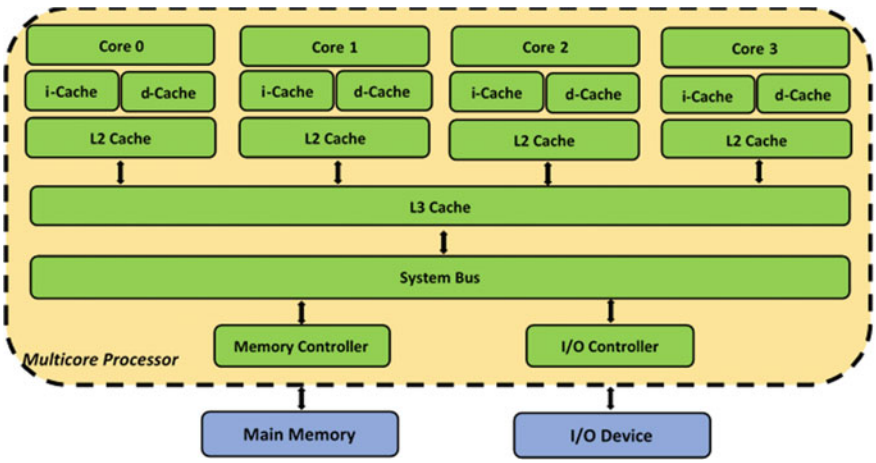

Fig. 1 A multicore chip

heuristic makes the scheduler complex with incidental cases of non-interactive jobs being identified as interactive jobs. CFS uses the concept of execution time and alleviates such miscalculations by organizing the jobs as red–black tree. The job with lowest execution time is the leftmost node of the tree and is picked up by CFS when invoked to schedule a process. CFS is a work-conserving scheduler which implies that the cores are prevented from being idle if there are processes ready for execution. However, experimental studies conducted by Lozi et al. [11] reveal that the Linux scheduler leaves cores idle while threads are waiting in run queues. This is revealed by performance degradation.

4 Results and Discussion

To make a comparative analysis of individual core usage with that of aggregate CPU utilization, we monitor per-core CPU utilization for workload types given below:

- A *CPU-intensive workload* is one that involves massive computation such as in financial modeling and scientific applications.
- *Memory-intensive workloads* involve high memory activity such as pushing of large volumes of data into and out of memory and frequent and long durations of memory read and write operations. Such workloads need not necessarily bloat memory usage and involve minimal CPU processing.

4.1 Experimental Cloud Setup

Usage of CPU cores is collected on a cloud setup in the laboratory. The experimental cloud consists of a controller node, a Network File System (NFS) server node, and two compute nodes. The configuration and software environment details are given in Table 1.

Table 1 Node configuration

Node	Configuration	Environment
Server and compute	Intel core i7 4790 3.60 GHz quad-core CPU, 16-GB 1600 MHz DDR3 RAM	Linux Kernel Version 4.4.0-116-generic, QEMU-KVM Version 2.5.0
Controller	Intel core i7 4770 3.40 GHz quad-core CPU, 16-GB 1600 MHz DDR3 RAM	Linux Kernel Version 4.4.0-116-generic

4.2 Design of Experiments

Applications used to generate different workload types are listed in Table 2, and different test cases are given in Table 3. Experiments are conducted on VMs running on a single host that uses full virtualization with Kernel-based Virtual Machine (KVM). The core virtualization infrastructure is implemented as a loadable kernel module (kvm.io). The modules, kvm-intel.ko and kvm-amd.ko, correspond to processors specific to Intel and AMD, respectively. Virtualization with KVM leverages basic mechanisms of CPU scheduling, memory management, and I/O access, provided by the Linux kernel.

To emulate real-world service request scenario, we generate variable job arrival times using a non-homogeneous Poisson process. Workloads are executed through a Bash script according to the generated job arrival times. Resource utilization log is collected for 1 hour at 3 seconds interval using System Activity Report (sar) system monitoring tool for both the guest and the host. Usage statistics extracted from the collected log are analyzed using R programming environment.

Baseline Resource Usage. To obtain CPU utilization for running basic system services and virtualization overheads, we run two VMs with no other workload on the guest or the host. A plot of the aggregated CPU usage is shown in Fig. 2, and usage statistics are given in Table 4. Spikes in the graph indicate some high CPU activity for a short duration. Mean value indicates that only 1–3% of the core usage accounts for running operating system as well as virtualization software.

CPU-Intensive Workload on Single VM. Per-CPU core usage of host and guest machines when running CPU-intensive workload on a single machine is depicted in Fig. 3. Summary statistics for host and guest are given in Tables 5 and 6, respectively.

Table 2 Workload generation

Workload type	Real-world application
CPU intensive	Prime factorization, Fibonacci series generation, factorial calculation
Memory intensive	Swap two integers, array copy

Table 3 Test cases for usage monitoring

Test case	Measurement objective
Baseline resource usage	Overhead due to basic system services and virtualization
CPU-intensive workload on single VM	Individual CPU core usage
Memory-intensive workload on single VM	Individual CPU core usage
CPU-intensive workload on one VM, memory-intensive workload on second VM	CPU core usage for workloads with complementary computational requirements
CPU-intensive workloads on three VMs	CPU core usage for workloads having similar computational requirements

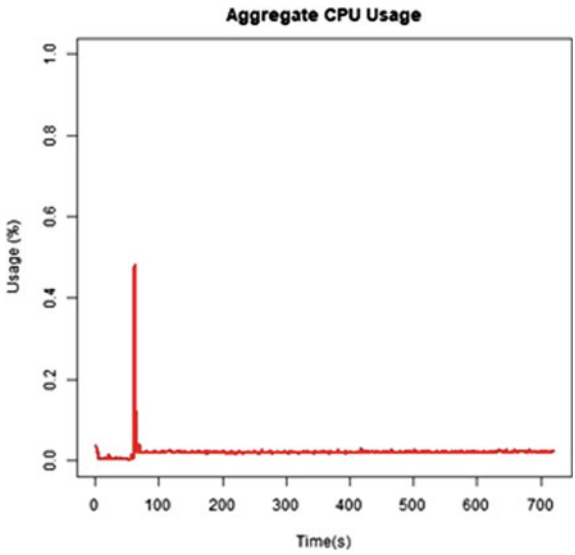

Fig. 2 Baseline aggregate host usage

Table 4 Summary statistics of baseline host CPU core usage

CPU usage	Min.	Median	Mean	Max.	Standard deviation
Aggregate	0.15	2.05	2.15	48.34	2.53
Core 0	0.00	2.40	2.35	39.28	2.55
Core 1	0.00	2.20	1.99	64.87	3.13
Core 2	0.00	2.40	1.80	80.49	3.91
Core 3	0.00	1.00	2.24	44.42	2.51

Table 5 Summary statistics for host CPU core usage with CPU-intensive workload

CPU usage	Min.	Median	Mean	Max.	Standard deviation
Aggregate	1.96	75.48	70.60	80.53	13.01
Core 0	0.00	04.00	41.57	100.00	44.97
Core 1	0.34	100.00	80.73	100.00	33.48
Core 2	0.00	100.00	90.54	100.00	20.21
Core 3	0.34	100.00	68.12	100.00	41.44

Mean utilization of the cores lies between 41 and 90%, whereas aggregate usage is 70% with core 2 being highest utilized and core 0 being the least utilized than what is given by the aggregate usage. High values for standard deviation show the widely varying usage levels of individual cores. Core usage by the VM as given in Table 8 reveals near equal usage of all the vCPUs for executing the workload.

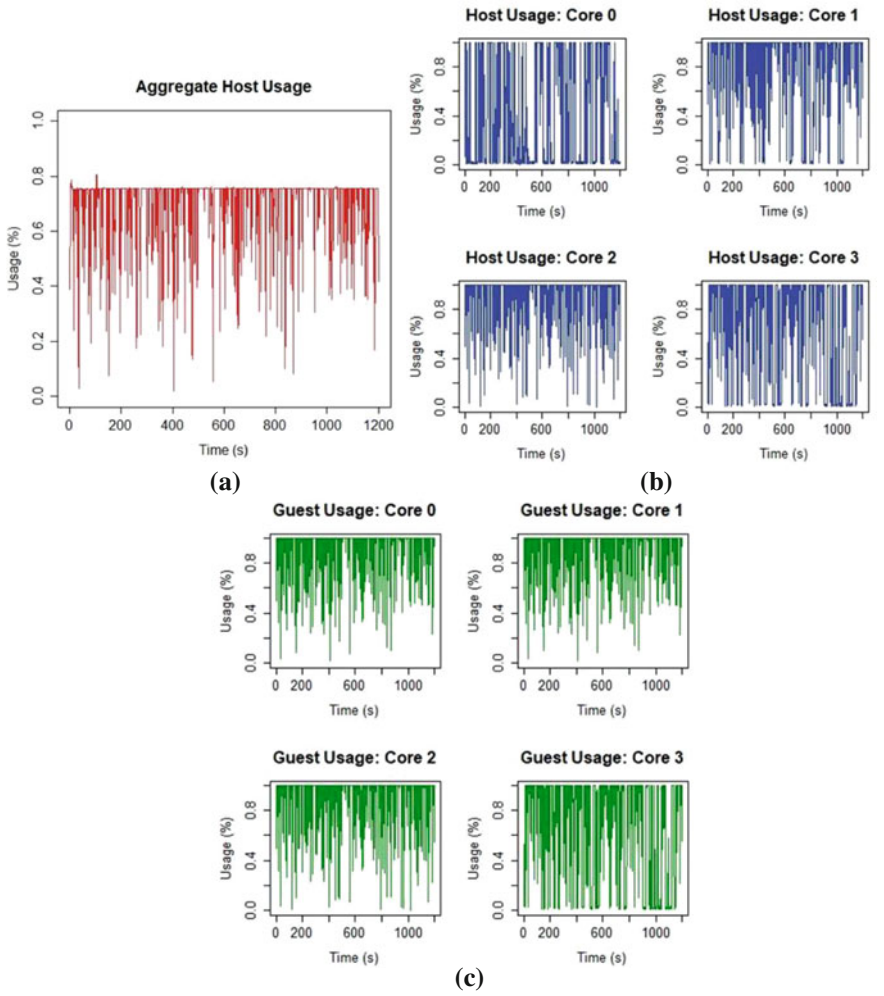

Fig. 3 CPU utilization for CPU-intensive workload. **a** Host aggregate, **b** per-core host, and **c** per-core guest

Memory-Intensive Workloads. Figure 4 shows a plot of per-core CPU usage for host and guest, running memory-intensive workloads on a single VM. The plots reveal that CPU activity is low with memory-intensive workloads as compared to CPU-intensive workloads, as the case should be. The cores either are mostly idle or have very low utilization as the median values in Table 7 reveal.

In this case, utilization of core 1 is 2% lesser than the aggregate and utilization of core 2 is 5% higher as given in Table 6.

Memory- and CPU-Intensive Workloads. The usage plots for this case are shown in Fig. 5, and Table 8 contains the usage statistics. While mean aggregated CPU

Table 6 Summary statistics for guest CPU core usage with CPU-intensive workload

CPU usage	Min.	Median	Mean	Max.	Standard deviation
Aggregate	2.11	100.00	92.11	100.00	17.28
Core 0	3.01	100.00	92.22	100.00	17.06
Core 1	1.67	100.00	92.09	100.00	17.32
Core 2	1.67	100.00	92.11	100.00	17.27
Core 3	1.33	100.00	92.08	100.00	17.34

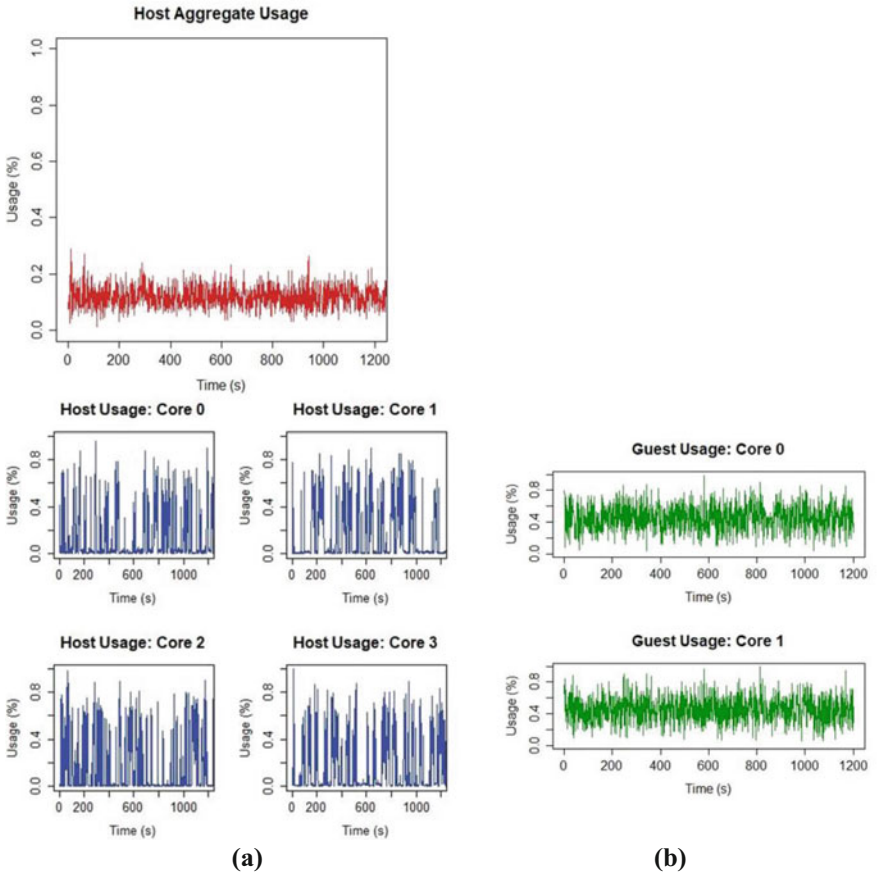

Fig. 4 Memory-intensive workloads **a** Aggregate CPU usage, **b** per-CPU core host usage, and **c** per-core guest usage

usage is close to 21%, core 0 and core 1 have higher mean usage. Such a usage detail can be detrimental when scheduling pCPUs to vCPUs in a virtualized environment. Also, since computational requirements of CPU- and memory-intensive workloads

Table 7 Summary statistics of host and guest CPU core usage for memory-intensive workload

CPU usage	Min.	Median	Mean	Max.	Standard deviation
Host usage					
Aggregate	1.40	11.59	11.84	28.85	03.76
Core 0	0.00	1.97	13.135	96.20	20.81
Core 1	0.00	0.60	08.389	98.00	21.59
Core 2	0.00	1.60	15.485	100.00	22.10
Core 3	0.00	0.200	10.446	77.110	08.22
VM usage					
Aggregate	5.51	45.05	45.76	100.00	14.74
Core 0	4.19	45.80	45.92	100.00	16.33
Core 1	1.60	45.20	45.59	100.00	16.23

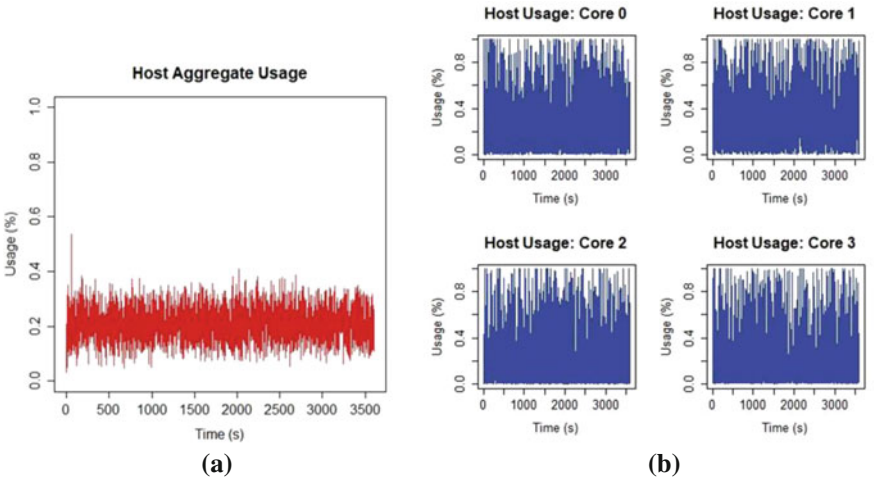

Fig. 5 Compute- and memory-intensive workloads: **a** host aggregate usage and **b** host per-core utilization

are complementary, the workloads do not contend for compute capacity of cores as can be seen from Fig. 6.

Compute-Intensive Workloads on three VMs. To study CPU core usage with workloads having similar computational requirements, we run two VMs, each with compute-intensive workloads, on a single host. Figure 7 contains the usage plots, and utilization values are given in Table 9.

From Table 9, we make similar observations as in the previous cases that in the case of multiple VMs running workloads in terms of similar resource requirements, core utilization is not uniform. Mean utilization of Core 1 is higher than that reported by the aggregate usage while that of Core 3 is below the aggregate value.

Table 8 Summary statistics of host CPU Core usage for memory- and CPU-intensive workloads

CPU usage	Min.	Median	Mean	Max.	Standard deviation
Aggregate	3.23	20.62	20.95	53.79	6.23
Core 0	0.00	17.14	24.63	100.00	22.34
Core 1	0.00	18.30	25.02	100.00	24.93
Core 2	0.00	25.27	19.95	100.00	24.43
Core 3	0.00	13.21	19.17	100.00	21.94

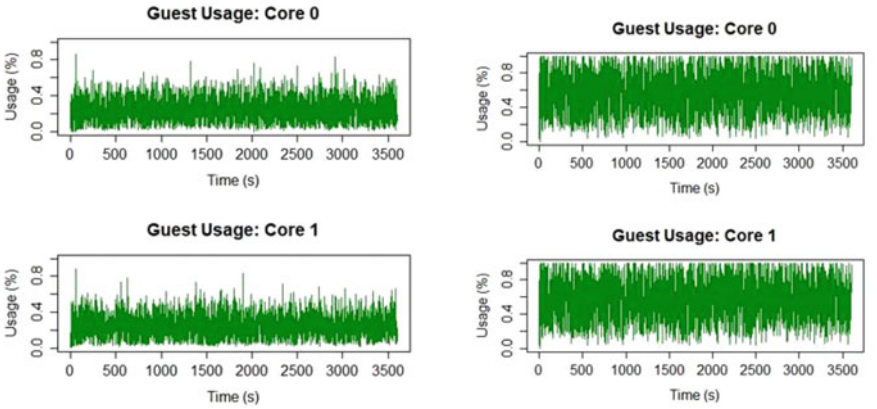

Fig. 6 Guest per-core utilization. **a** Memory-intensive workload and **b** CPU-intensive workload

Table 9 Summary statistics of host with CPU-intensive workloads on three VMs

CPU usage	Min.	Median	Mean	Max.	Standard deviation
Aggregate	53.36	73.85	75.08	100.00	10.51
Core 0	07.12	94.67	74.70	100.00	25.18
Core 1	07.38	72.65	82.02	100.00	26.16
Core 2	08.45	96.67	75.52	100.00	23.61
Core 3	10.03	94.64	67.95	100.00	22.37

Table 10 summarizes the percentage difference in utilization of individual cores from that of the aggregated usage. With compute-intensive workloads on single VM (Case II), a core usage of 29% below and 20% above the aggregate usage is recorded. When running only memory-intensive workloads (Case III), individual core usage is 3% above and below the aggregate usage. Similarly, in case of CPU- and memory-intensive workloads (Case IV) and CPU-intensive workloads on multiple VMs (Case V), the core usage varies between 1% below and 4% above the aggregate usage for Case IV and 7% above and below the aggregate usage in Case V. Knowledge

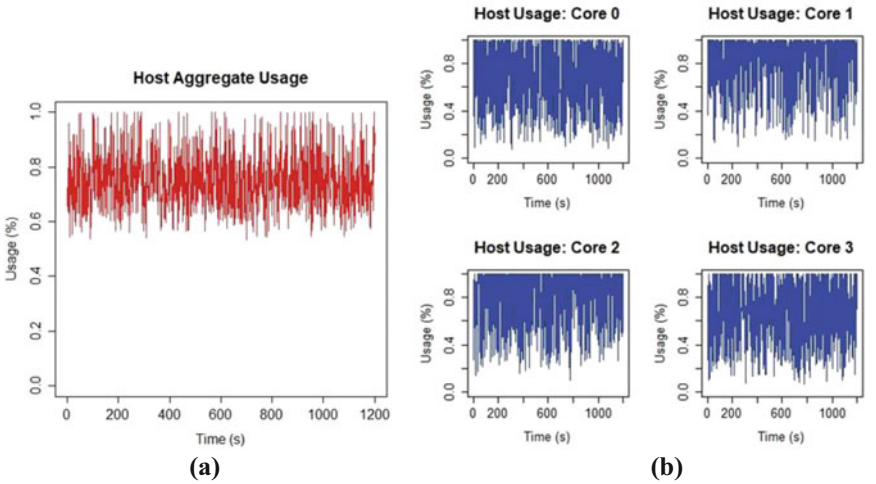

Fig. 7 CPU-intensive workloads on three VMs. **a** Host aggregate usage and **b** per-core host usage

Table 10 Usage comparison of CPU cores with aggregate usage

	Aggregate	Core 0	Core 1	Core 2	Core 3
Case II	70.60	41.57	80.73	90.54	68.12
Case III	11.84	13.14	8.40	15.48	10.45
Case IV	20.95	24.63	25.02	19.95	19.17
Case V	75.08	74.70	82.02	75.52	67.95

of individual core usage can be instrumental in effective scheduling and resource utilization.

5 Conclusion and Future Scope

In this work, we conducted an empirical study of utilization of CPU cores of a multicore server to analyze core-level CPU usage to verify our observations that aggregated CPU usage is not reflective of utilization at the core level. We collected CPU usage for five test cases—baseline usage and CPU core usage in case of CPU-intensive workload on single VM, memory-intensive workload on single VM, CPU-intensive workload on one VM and memory-intensive workload on the other, and CPU-intensive workloads on two VMs. We observe that basic system services and virtualization software incur some overhead in terms of CPU usage, as the case should be. The remaining four cases clearly bring out differences between aggregated CPU usage and individual cores which ranges from 1 to 29% below and between 4 and 20% above the aggregate usage. A precise knowledge of individual core usage will help in

identifying cores that are underutilized and making informed scheduling decisions. Additionally, it will also be a vital input for reducing the load of overloaded cores and devising policies to prevent aggravating of the overload situation.

As future research, we would study core usage for I/O- and network-intensive workloads. A model based on the usage measurements of the four workload types—CPU, memory, I/O, and network intensive—would then be developed for workload characterization and prediction at the level of individual CPU cores. We would delve deeper into the varied usage levels of the CPU cores to study their criticality and impact on overall performance on different workload types. Scheduling policies that consider usage of CPU cores and account for inferences derived from the investigations would be developed, the impact of such a fine-grained scheduling on overall system performance would be analyzed, and approaches to mitigate performance loss would be investigated.

References

1. Varia, J., Mathew, S.: Overview of Amazon Web Services. Amaz. Web Serv. (2014)
2. https://creative-brackets.com/business/interesting-facts-statistics-largest-data-centers-world/. Last Accessed 2018/06/10
3. Feitelson, D.G.: Workload modeling for computer systems performance evaluation. Cambridge University Press, Cambridge (2015)
4. Mahambre, S., Kulkarni, P., Bellur, U., Chafle, G., Deshpande, D.: Workload characterization for capacity planning and performance management in IaaS cloud. In: 2012 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 1–7 (2012)
5. Moreno, I.S., Garraghan, P., Townend, P., Xu, J.: Analysis, modeling and simulation of workload patterns in a large-scale utility cloud. IEEE Trans. Cloud Comput. **2**, 208–221 (2014)
6. Magalhães, D., Calheiros, R.N., Buyya, R., Gomes, D.G.: Workload modeling for resource usage analysis and simulation in cloud computing. Comput. Electr. Eng. **47**, 69–81 (2015)
7. Peng, J., Chen, J., Zhi, X., Qiu, M., Xie, X.: Research on application classification method in cloud computing environment. J. Supercomput. **73**, 3488–3507 (2017)
8. Zhang, Q., Zhani, M.F., Boutaba, R., Hellerstein, J.L.: Dynamic heterogeneity-aware resource provisioning in the cloud. IEEE Trans. Cloud Comput. **2**, 14–28 (2014)
9. Singh, S., Chana, I.: QRSF: QoS-aware resource scheduling framework in cloud computing. J. Supercomput. **71**, 241–292 (2015)
10. Orzechowski, P., Proficz, J., Krawczyk, H., Szymański, J.: Categorization of cloud workload types with clustering. In: Proceedings of the International Conference on Signal, Networks, Computing, and Systems, pp. 303–313 (2017)
11. Lozi, J.P., Lepers, B., Funston, J., Gaud, F., Quéma, V., Fedorova, A.: The Linux scheduler: a decade of wasted cores. In: Proceedings of the Eleventh European Conference on Computer Systems, p. 1 (2016)
12. Capacity Planning—Discipline for Data center decisions, https://www.teamquest.com/files/6814/2049/9759/tqeb01.pdf. Last Accessed 2018/06/10
13. Sokol, A.W., Hogan, M.D.: NIST Cloud Computing Standards Roadmap (2013)
14. Calzarossa, M.C., Della Vedova, M.L., Massari, L., Petcu, D., Tabash, M.I.M., Tessera, D.: Workloads in the clouds. In: Principles of Performance and Reliability Modeling and Evaluation, pp. 525–550. Springer, Berlin (2016)
15. Vajda, A.: Programming many-core chips. Springer Science & Business Media, Berlin (2011)