# Survey on Application Profiling in Recent Industrial Research

Xiao Luo, Xinyu Wang

## 1. Introduction

Numerous dimensions of the resource, namely CPU, memory, networking, etc., need to be managed in modern big data centers. However, many current architectures schedule those resource separately, and even worse, just over-provision, which cause extremely low resource utilization and significant waste. Therefore, we need an efficient resource management methodology with considering multi-dimension resource units and low complexity.

There are some key challenges in multi-dimensional resource management between many servers now[1]:

1) Resource fragmentation. Nowadays, most resource schedulers split resource assignments and schedule operations for each resource type. Separating those tight resources will constrain their ability to pack tasks.

2) Non-Optimized resource assignment in practice. In practice, schedulers usually assign resources by peak requirements. But this method will cause over-provisioning in high-priority tasks and starvation in low-priority tasks. That's because workload's high dynamism in data centers and may change the resource requirement over its time of execution.

3) In modern data centers, there are countless types of fine-grained resources (for example, CPU, memory blocks, storage devices, switches, etc.), resulting in an astounding number of possible resource combinations. As a result, the complexity of jointly optimizing all types of resources in the same framework increases exponentially.

Multi-Resource Schedulable Unit (MRSU)[1] can improve resource utilization by considering multi-dimension resources, high dynamism, and different levels of granularities. The essential logic of MRSU is to pack multi-dimension resources as a unit and assign enough resource units for each tenant. It's a trade-off between scheduling complexity and over-provisioning that it will neither assign resource one dimension by one dimension nor assign resource by peak requirement. However, before we can assign resource units to each tenant, we need to know the definition of a resource unit, known as the definition of MRSU, that is, the available amount of each resource dimension in a resource unit. In [1], the author proposes a weighted fair MRSU algorithm to determine a resource unit according to the

resource requirement of each tenant and application over time. Therefore, the most important issue now is to find a practice and general method to profile applications before we can achieve our final goal: Schedule countless dimension resources efficiently in highly dynamic data centers to improve resource utilization.

In this survey, we will introduce and conclude some recent research and development of application profiling, especially from the industry perspective. And we will conclude existing work and give an insight into what we still need to achieve our final goal.

The rest of the paper is organized as follows. Section 2 gives an overview of application profiling. Section 3 represents and analyzes recent application profiling techniques. Section 4 analyzes how can they be partially deployed for MRSU's application profiling requirement and how far are these techniques from meeting the joint profiling requirement of MRSU.

## 2. Application Profiling Overview

In this section, we represent the overview of recent research on application profiling in big data centers. We introduce the input source of application profiling. We discuss how we can gather necessary input information from those sources. We discuss input information types, focusing on their importance and measure ways. We introduce two major application profiling categories. Finally, we conclude expected application profiling result of recent profiling techniques.

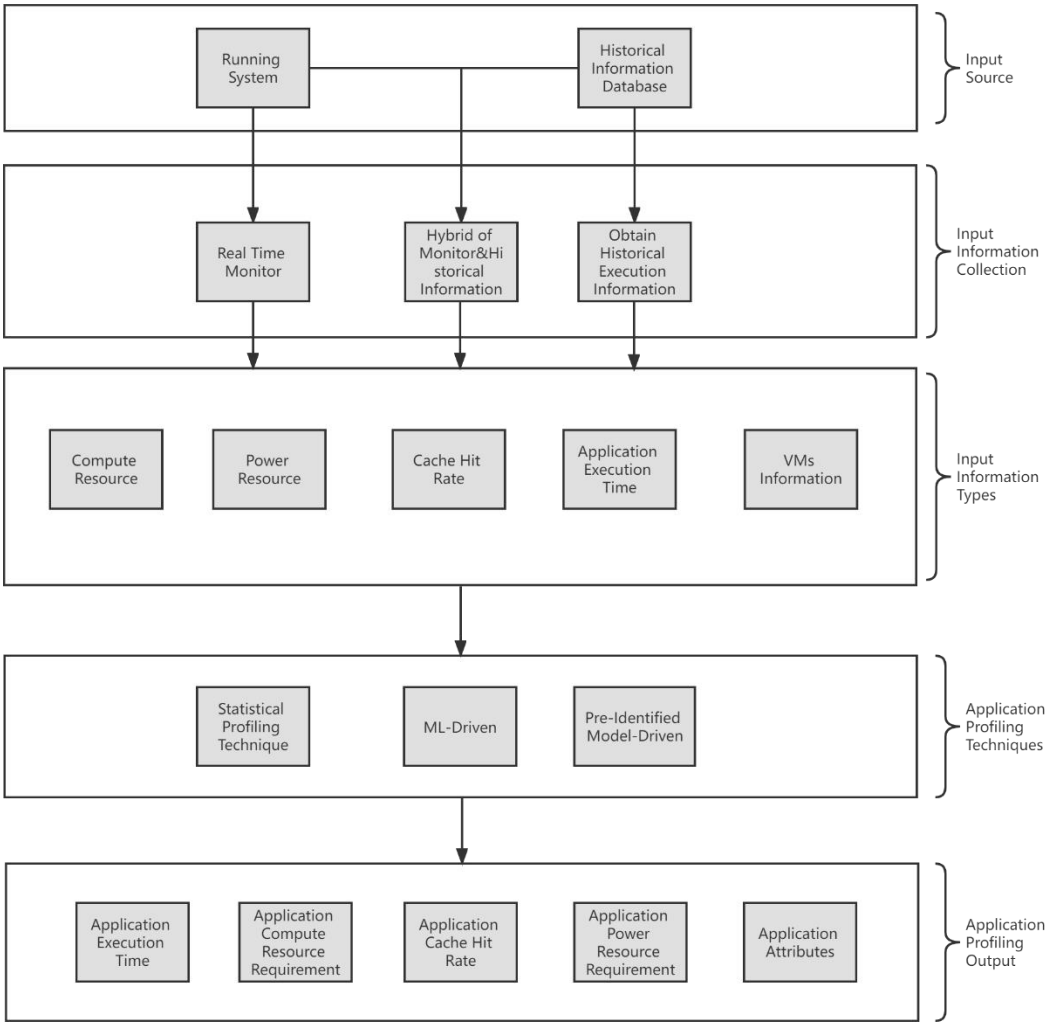


**Fig. 1** Conceptional framework for application profiling in big datacenters. Unidirectional arrows describe the information flows between different modules. The diagram depicts the responsibilities of each module and their major categories.

Recent application profiling techniques are depicted in Fig. 1. And its major modules are described below.

## 2.1 Input Information Source and Collection

Since real time monitoring information can collect application information at present moment, which are usually similar with the next moment, collecting information from running system on real time is necessary. Moreover, many cloud service providers offer comprehensive real time monitor services for their cloud resources. For instance, Google Cloud provide monitoring information about virtual CPU, memory, storage, network etc., from function-level to application level [2]. Therefore, it's easy for many application profiling techniques to get access with necessary input information.

In addition, there are some information outside conversational monitoring metrics can help us to better profile application. For example, [3] analyzes application's criticality by its workload's periodicity information collecting from historical information, that if an application always works on the day and release at night, it cloud be user-facing and therefore, critical one. Moreover, some Machine Learning (ML) based application profiling techniques always utilize application historical execution information to train a model to profile future applications [3-5].

Finally, some application profiling techniques combine real-time monitoring and historical execution information together to profile an application from several dimensions. [3] use historical workload's periodicity to predict future applications' periodicity and use CPU monitoring information to estimate future CPU usage. Moreover, ADARM framework[6] proposes that we can compare prediction generated by historical information and real-time monitoring information to continuously optimize the application profiling model.

## 2.2 Input Information Types and Profiling Output

There are many types of information that need to be considered in application profiling. We briefly categorize them into 5 types, namely compute resources, power resources, cache hit rate, application execution time, and Virtual Machines (VMs) information. We now discuss the quantifiable unit of that information that can enable application profiling.

### 2.2.1 Compute Resource

Typically, compute resources are the collection of Physical Machines (PMs), each comprised of one or more CPU cores, memory, etc., which together provide a complete computational capacity for application[7].

CPU resource information is the most common input in application profiling. Many application profiling techniques take CPU cores number or CPU utilization as the input compute resource unit[3, 5, 8, 9]. Moreover, with the development of system virtualization technology, many physical machines deploy virtualization software to host some VMs that run different operating systems, and many VMs use virtual CPU core (core) techniques to share the same compute resource. Therefore, tremendous application profiling techniques take VMs number or vcore number as the compute resource unit[3, 5, 10-12].

Moreover, memory is another necessary resource need to consider, which is usually taken as input information together with CPU in application profiling. Some application profiling techniques take memory usage information as input to provide more accurate profile for application[4, 5, 9, 10].

### 2.2.2 Power Resource

In the past, cloud service providers like Microsoft Azure, provision power for each server based on either its nameplate maximum power or its peak draw while running a power benchmark, which leads to massive power under-utilization[3]. However, datacenters in United States consumed an estimated 70 billion kWh, representing approximately 1.8% of the total U.S. electricity consumption[13].

Therefore, some application profiling techniques take power as input to estimate future power budgets to avoid unnecessary costs. Moreover, profiling application's power can also enable the balanced distribution of critical VMs, which can decrease influence when the power system is down.

### 2.2.3 Cache Hit Rate

With the exponentially growth of Internet users and content, information retrieving cost, especially flash or disk I/O cost, is extremely significant in the system performance. For instance, Google now processes over 40,000 search queries every second on average, which translates to over 3.5 billion searches per day[14], retrieving results from an index of many billions of documents.

At this scale, some application profiling techniques try to take cache hit rate between application and PMs as input information, therefore, to significantly decrease storage I/O cost[15].

### 2.2.4 Application Execution Time

Most Internet-based applications nowadays utilize remote cloud compute resources provided by cloud datacenters. Typically, cloud service users have some Service Level Agreements (SLAs) with cloud service providers, especially execution time SLA. Cloud service providers usually choose over-provision strategy to meet execution time SLA, which cause under-utilization of resources.

Therefore, some application profiling techniques take application execution time as input information with other information together to minimize resource costs under the limitation of execution time SLA[4, 8].

### 2.2.5 Information about VMs hosting application

When provide Infrastructure as a Service (IaaS) to clients, cloud service providers usually have limited information about application running on VMs due to privacy SLA. Therefore, some application profiling techniques use information about VM itself and VM-to-VM traffic

to estimate roles and functionality of application running on VMs, which can be used in application profiling techniques to infer applications'[12].

## 2.3 Application Profiling Techniques Taxonomy

Big datacenters host numerous different applications from Internet. Since it is well known that the workload and resource demand of different applications varies dynamically over time, there are many application profiling techniques try to profile application on real time. Those techniques can broadly be categorized into 3 types. We will discuss these different techniques below.

### 2.3.1 Statistical Profiling Techniques

For most scenario, engineers only consider single or several dimensions of application performance metrics, especially CPU and memory. The essential function of statistical profiling techniques is to take specific dimensions as input and provide corresponding dimensions as output by some statistical formulation. This type of techniques has the strongest interpretability and high universality, but they are hard to adjust complex profiling requirements and have limited accuracy.

### 2.3.2 Pre-Identified Model

With necessary knowledge of application context and logic, some application profiling techniques utilize some application-specific characteristics to design a profiling model with combining many dimensions as input and get expected output. In [8], the author profile Hadoop[16] application. [8] separates profiling procedure into 3 stages as Map Task Stage, Map Stage and Reduce Stage, with taking historical execution time, CPU cores number and tasks number from each stage and predict execution of corresponding stage. Pre-Identified Model can draw on application-specific information to better profile the application and provide more information. But it is lack of university and really need deeply understanding of application to propose an accurate model.

### 2.3.3 ML Model Driven

With the progressively mature of ML application and development of computation capacity, more and more scientists and engineers utilize ML-based approaches for prediction. The essential function of ML for application profiling is to utilize multiple dimensions of information in the past to predict specific dimensions, known as profiling output. This type of techniques does not need deeply understanding of application and can evolve on its own with constantly input. [4] tries some ML models and choose one with best accuracy. Moreover, its model keeps learning from the execution in the cluster, becoming more

accurate without considerable overhead. However, to train an accurate ML model, we indeed need a large quantity of historical information to train the ML model and need to consume a lot of compute resources.

# 3. Application Profiling Techniques

As the prerequisite of resource management in big datacenters, application profiling techniques have been studied by countless researchers either from academic or industry. In this section, we discuss those techniques within the scope described in Sect. 2.3.

**Table 1** Comparing works in application profiling

| Publication | | Techniques Type | | | Profiling Result | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Reference | Year | SPT | PIMD | MLD | AET | CR | PR | AT | OD |
| Verma, Pedrosa [9] | 2015 | × | | | | | | | × |
| Gandhi, Thota [8] | 2016 | | × | | × | | | | |
| Gandhi, Dube [11] | 2017 | | × | | × | | | | |
| Hernández, Perez [4] | 2018 | | | × | × | | | | |
| Archer, Aydin [15] | 2019 | × | | | | | | | × |
| Chen, Goetsch [10] | 2019 | | × | | × | | | | |
| Balaji, Kakovitch [5] | 2021 | | | × | | × | | | × |
| Bashir, Deng [17] | 2021 | × | | | | × | | | |
| Kumbhare, Azimi [3] | 2021 | × | | × | | | × | × | |
| Pang, Panda [12] | 2022 | | × | | | | | × | |
| Yadwadkar, Hariharan [18] | 2017 | | | × | | × | | | |
| Vasudevan, Tian [19] | 2017 | | | × | × | | | | |
| Baughman, Chard [20] | 2019 | × | | | | | × | | |
| Mao, Akgul [21] | 2022 | × | | | | × | | | |

Most application profiling techniques profile applications' execution time and compute resource, and a few for other dimensions. All 3 technique types are adopted evenly.

*SPT* Statistical Profiling Techniques, *PIMD* Pre-Identified Model Driven, *MLD* Machine Learning Driven, *AET* Application Execution Time, *CR* Compute Resource, *PR* Power Resource, *AT* Application Attributes, *OD* Other Dimensions namely VMs arrival pattern, cache hit rate and application priorities.

## 3.1 Statistical Profiling Techniques

As mentioned in Sect. 2.3.1, the basic logic of statistical profiling techniques is predicting some applications' information by historical information with the same dimensions, and without considering applications' characteristics. Archer, Aydin [15] want to profile cache hit rate between a request to a specific replica worker machine. They use a priori probability of cache hit rate based on historical cache hit information to estimate incoming applications' cache hit rate to every replica and choose the one with highest cache hit rate.

Sometimes we can combine multiple statistical methods to get better results. Bashir, Deng [17] mentions 3 statistical methods to predict task resource usage, especially compute resources usage: use the sum of tasks' resource requirement to multiply with a fixed ratio; use sum of each task' resource utilization; use combination of mean total usage and its standard deviation. The author combines these 3 methods and take maximum predictive resource usage as profiling results, which works better than individually adopting those methods.

In [20], the scenario is scientific workloads. For those non-parallel applications, the system will record the start and end time. For those parallel applications, deploy and monitor the applications in different types, to get the CPU, memory demands. Also, in this paper, the final output considers the amount of data, to get the final configuration of VMs.

In [21], this paper mainly considers the latency requirement in different scenario of vehicular fog computing. In this paper, the system linearly increases the number of the same type of application until the latency requirement cannot be satisfied. In this way, the system will get the maximum number of applications which can be run in the same time, and the mean CPU usage can also be calculated.

## 3.2 Pre-Identified Model Driven Profiling Techniques

As mentioned in Sect. 2.3.2, the basic logic of pre-identified model driven profiling techniques is designing a pre-identified model with considering application-specific characteristics, and input the applications' real time monitoring or historical information into the model and get the profiling information of current or future applications.

The growing requirement for big data processing and analytics driven by the progressively development of Internet has spawned an array of frameworks tailored for this requirement, such as Hadoop[16], Spark[22], etc. Gandhi, Thota [8] proposes an application profiling techniques to determine resources needed by a given Hadoop job within meeting its execution time SLA. They mention that data processing jobs are often composed of multiple stages and each stage requires different resources allocations. Therefore, they separate profiling procedure into 3 stages as Map Task Stage, Map Stage and Reduce Stage. For each stage, they use different statistical formulations defined by Hadoop jobs' characteristics to predict corresponding execution time, which can help them to schedule and provision

resource for jobs to meet execution time SLA. In addition, since Hadoop applications performance depends on many parameters such as CPU cores number, tasks number, etc., their model considers execution information with historical data together to calculate predictive execution time for incoming applications.

Moreover, some application profiling techniques consider about parallelism in cloud environment. Gandhi, Dube [11] proposes that adding more compute resources to the existing VMs works better for parallel work, whereas adding more VMs is more effective for high load sequential work. Therefore, they separate their model into 2 parts as sequential one and parallel one. For sequential part, they design the model based on sequential RUBiS[23] workloads, which are comprised of one front-end VM to receive requests, one database VM to access storage and many application tier VMs to process applications. They use a queueing-network model[24] with considering service time and workloads request rate to estimate mean response time for application tier VMs. For parallel part, they design the model based on parallel LINPACK[25] workloads with considering parallel portion of application, virtual CPUs (vCPUs) number and service time to estimate mean response time. Finally, they combine sequential part and parallel part as a complete model. Sequential part considers VMs' external information to add more VMs and parallel part considers VMs' internal information to add more compute resources, when estimated mean response time exceeds service time SLA.

Some application profiling techniques also consider about privacy SLA. Pang, Panda [12] proposes that cloud services providers always have limited visibility into cloud service functionality because of privacy SLA. And we need to profile applications' functionality within limited information for better resource management from cloud services providers' perspective. They creatively utilize traffic information between VMs, VMs' placement and provision information to infer their functionality by a model based on matrix theory. The essential logic of the model is that VMs with the same functionality will have similar linear dependance nature in their traffic matrix. With this information, we can assign similar resources to VMs with the same functionality and profile a VM with information from other VMs with the same functionality.

The output of a pre-identified model is not necessary a concreate metrics of applications, like CPU cores number, application response time, etc. Applications' profiles can be abstracted as a cost about representing your specific scenarios. Borg system[9], which admits, schedules, starts, restarts, and monitors the full range of applications that Google runs, utilizes E-PVM[26] strategy spread tasks across machines, or Best-Fit strategy to fill machines with tasks as tightly as possible. For these two different strategies, they use different formulations to represent costs of placing a new task in a machine. And based on different profiles, Borg's schedulers can result in different distributions of tasks.

Unlike the models mentioned above that consider specific scenarios, Chen, Goetsch [10] propose the d-Simplexes model, which takes any dimension of information that related to execution time as input and get predictive execution time as output. They build a model based on Delaunay Triangulation from computational geometry, that input $\mathrm{R}^{\mathrm{d}}$ space with

n-dimension vectors into the model and output $n+1_{th}$ dimension value. They also adopt Latin Hypercube Sampling, which can utilize fewer points to represent inputting features, to decrease computation complexity of the model.

## 3.3 ML Driven Model

Unlike pre-identified models in Sect. 3.2, ML driven models don't need deep understanding of application to design a model. What you need to do is to collect a large quantity of data from numerous dimensions that related to your expected output dimension. Hernández, Perez [4] considers many input dimensions such as executors (compute resource unit) number, tasks number, memory, number of applications that access storage, CPU waiting time, etc. as input to predict applications execution time. They evaluate 7 different ML models with historical data from multiple dimensions and chose one model with minimum Mean Absolute Error (MAE) between predictive and true values.

In addition to execution time, Balaji, Kakovitch [5] use imitation learning to predict future resource usage of a running application by application historical information, especially CPU usage. Their goal is finding a better physical machines to place incoming VMs. They utilize historical information to predict future resource usage of existing VMs and predict typical distribution of future VMs' arrival patterns. Based on this knowledge, their approach explores every placement combination to minimize average resource usage.

Sometimes combining ML models with human's experiences may work well. Kumbhare, Azimi [3] separate VMs as user-facing and non-user-facing, and points that user-facing VMs are usually critical. Since human usually work on day and relax at night, they label VMs by their periodicity and use labelled data to train the ML model to profile VMs' priorities (critical VM has high priority). Therefore, they can balance the distribution of critical VMs, that can lower power during an event without affecting critical VMs.

The method used in [18] also leverage the ML algorithm to do the application profiling. When the application come, the decision tree will keep bringing up the question to split the resource demand of the application into different types (the test will be done on experiment VMs), and the system will record the split point and schedule them into a fingerprint. The fingerprints can be used to match different types of VMs.

Paper [19] assumed that the application run-time is calculated using a hyper-gamma distribution. This model is proposed by Lublinand Feitelson. [27] The model is hyper-gamma mixture model, so we can use a ML algorithm, the EM algorithm to calculate the parameter in the situation of knowing the type and number of functions. When we get the function, we can know the required CPU and memory of each application.

# 4. From Recent Techniques to MRSU

With MRSU, the application profiles are combined across applications and across tenants

with considering multiple resources dimensions so that a joint profiles is utilized to derive the datacenter-wide MRSU[1]. However, recent application profiling can only partially meet this requirement. In this section, we discuss what recent techniques can do for MRSU and how far are these techniques from meeting the MRSU requirements.

**Multiple Dimensions.** Most applications profiling techniques only consider about single or limited dimensions, especially compute resources. But for deriving a datacenter wide MRSU, we need an application profiling technique to profile applications from multiple dimensions, at least from CPU, memory, networking, and storage dimensions. Although Chen, Goetsch [10] considers multiple dimensions in its model, it can only have one dimension output. Balaji, Kakovitch [5] mentions that its ML model can predict multiple dimensions of resource usage, the author only tests its model in the compute resource dimension.

In fact, those methods have covered multiple dimensions, such as compute resource, power resource, application periodicity, application priority, application execution time, etc., each application profiling technique only consider single dimension. What MRSU need is to consider those dimensions together in a more complete application profiling framework.

**Universality.** Another key issue of recent techniques is that most of them can only work in a specific scenario, such as Hadoop jobs, specific VM topologies, only consider single dimension as bottleneck, etc. However, there are countless different types of applications and resource dimensions in a big datacenter. As a datacenter-wide resource unit, MRSU need an application profiling technique to profile every application with different contexts and characteristics. Some statistical and ML driven profiling techniques can be generalized to consider more universal scenarios, but their covered scope is still limited.

**Dynamism.** Dynamism is the most prominent nature of modern big datacenters. Some statistical methods and pre-identified models are designed by priori information and human's experiences. Although they can profile applications with dynamically changed input, application types or the structure of application itself changed dynamically in big datacenters. Therefore, MRSU need a more generalized profiling technique without specific assumptions to adjust this big datacenters' nature.

**Omission Factors.** In fact, there are more other factors are not considered that can influence the performance of resource management such as network resource, datacenter's location, storage resources[15], mobility, tenant, time sequence, etc.. Those factors are necessary component of a well-defined MRSU in big datacenters, that can be used in jointly profiling to enable resource allocation and scheduling optimization in ADARM[6].

**Relation Model.** Since most recent techniques can only apply in a specific scenario, they treat all applications as individual entities. However, a modern big datacenter always hosts countless types of applications and many applications inside it usually has relation with other applications, which indicates that they are interdependent. Therefore, future application profiling techniques need to consider their interdependent relation for more optimized resource allocations and scheduling.

# References


[1] Gutierrez-Estevez, D.M. and M. Luo. *Multi-resource schedulable unit for adaptive application-driven unified resource management in data centers*. in *2015 International Telecommunication Networks and Applications Conference (ITNAC)*. 2015. IEEE.

[2] Google. *Google Cloud*. [cited 2022; Available from: https://cloud.google.com/monitoring/docs?hl=en.

[3] Kumbhare, A.G., et al. *Prediction-Based Power Oversubscription in Cloud Platforms*. in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 2021.

[4] Hernández, Á.B., et al., *Using machine learning to optimize parallelism in big data applications.* Future Generation Computer Systems, 2018. **86**: p. 1076-1092.

[5] Balaji, B., C. Kakovitch, and B. Narayanaswamy, *FirePlace: Placing Firecraker Virtual Machines with Hindsight Imitation*. Proceedings of Machine Learning and Systems, 2021. **3**: p. 652-663.

[6] Luo, M., L. Li, and W. Chou. *ADARM: an application-driven adaptive resource management framework for data centers*. in *2017 IEEE International Conference on AI & Mobile Services (AIMS)*. 2017. IEEE.

[7] Jennings, B. and R. Stadler, *Resource Management in Clouds: Survey and Research Challenges.* Journal of Network and Systems Management, 2014. **23**(3): p. 567-619.

[8] Gandhi, A., et al., *Autoscaling for Hadoop Clusters*, in *2016 IEEE International Conference on Cloud Engineering (IC2E)*. 2016. p. 109-118.

[9] Verma, A., et al., *Large-scale cluster management at Google with Borg*, in *Proceedings of the Tenth European Conference on Computer Systems*. 2015. p. 1-17.

[10] Chen, Y., et al., *d-Simplexed: Adaptive Delaunay Triangulation for Performance Modeling and Prediction on Big Data Analytics*. IEEE Transactions on Big Data, 2019: p. 1-1.

[11] Gandhi, A., et al., *Model-driven optimal resource scaling in cloud.* Software & Systems Modeling, 2017. **17**(2): p. 509-526.

[12] Pang, W., et al. *CloudCluster: Unearthing the Functional Structure of a Cloud Service*. in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 2022.

[13] Shehabi, A., et al., *United states data center energy usage report.* 2016.

[14] *Google Search Statistics*. [cited 2022; Available from: https://www.internetlivestats.com/google-search-statistics/.

[15] Archer, A., et al., *Cache-aware load balancing of data center applications.* Proceedings of the VLDB Endowment, 2019. **12**(6): p. 709-723.

[16] APACHE. *Hadoop*. Available from: https://hadoop.apache.org/.

[17] Bashir, N., et al., *Take it to the Limit: Peak Prediction-driven Resource Overcommitment in Datacenters*, in *Proceedings of the Sixteenth European Conference on Computer Systems*. 2021. p. 556-573.

[18] Yadwadkar, N.J., et al. *Selecting the best vm across multiple public clouds: A data-driven performance modeling approach*. in *Proceedings of the 2017 Symposium on Cloud Computing*. 2017.

[19] Vasudevan, M., et al., *Profile-based application assignment for greener and more energy-efficient data centers*. Future generation computer systems, 2017. **67**: p. 94-108.

[20] Baughman, M., et al. *Profiling and predicting application performance on the cloud*. in *11th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*. 2018.

[21] Mao, W., et al., *Data-driven capacity planning for vehicular fog computing.* IEEE Internet of Things Journal, 2022.

[22] APACHE. *Spark*. [cited 2022; Available from: https://spark.apache.org/.

[23] *RUBiS: Rice University Bidding System*. . Available from: http://rubis.ow2.org.

[24] Walrand, J., *An introduction to queueing networks*. 1988, Englewood Cliffs, N.J: Prentice Hall. xii, 384 pages : illustrations.

[25] Corp., I. *Intel Math Kernel Library - LINPACK11.1 Update 2*. Available from: https://software.intel.com/en-us/articles/intel-math-kernel-library-linpack-download.

[26] Amir, Y., et al., *An opportunity cost approach for job assignment in a scalable computing cluster.* IEEE Transactions on parallel and distributed Systems, 2000. **11**(7): p. 760-768.

[27] Lublin, U. and D.G. Feitelson, *The workload on parallel supercomputers: modeling the characteristics of rigid jobs.* Journal of Parallel and Distributed Computing, 2003. **63**(11): p. 1105-1122.