Implementation of Profile-based Virtual Machine Placement Framework for Energy Optimization of Data Centers

IFN702
Siqi Zhang
N9077049

Supervisor: Professor Yu-Chu Tian, Zhe Ding

Table of contents

1 Executive Summary 4
2 Introduction 6
3 Environmental scan and review of prior related work 10
3.1 Virtualization 10
3.2 VM placement 11
3.3 Data profiling 11
3.4 Source data 11
4 Project Methodology and Approach 14
4.1 Project overview 14
4.2 Profiling 14
4.3 Tasks & VM Classifications 15
4.4 VM Placement 17
4.5 Evaluation 20
4.6 Summary 20
5 Outcomes or Results 21
5.1 Profiling 21
5.1.1 Large-scale experiment 21
5.1.2 Medium-scale experiment 23
5.1.3 Small-scale experiment 24
5.2 Task & VMs classification 25
5.2.1 Large-scale experiment 26
5.2.2 Medium-scale experiment 27
5.2.3 Small-scale experiment 28
5.3 VM Placement and PM quantity evaluation 30
5.3.1 Large-scale experiment 30
5.3.2 Medium-scale experiment 32
5.3.3 Small scale experiment 34
5.4 Average PM utilization evaluation 35
5.4.1 Large-scale experiment 35
5.4.2 Medium-scale experiment 36
5.4.3 Small-scale experiment 37

5.5 Energy consumption evaluation 38
5.6 Summary 39
6 Discussion 40
6.1 Advantages 40
6.2 Limitations 41
6.3 Recommendations 42
6.4 Summary 42
7 Conclusion 44
Reference 46
Appendix 1 Reflection Discussion 48
Appendix 2 Weekly Report 50

1 Executive Summary

Data center is the storage of cloud computing. With the network data volume increasing, data centers are facing with energy optimization issues. Unfortunately, quite lot data centers have low CPU utilization and some servers are even being idled. This will bring a huge waste in both energy consumption and money expense, as well as carbon dioxide pollution.

Server is one of the high energy consuming area of data center. The structure of server is consisted of applications, virtual machines (VMs) and physical machines (PMs). The nature of data center energy saving is to decrease physical machine quantity. That is, enhance PM utilization as much as possible when placing virtual machines on PMs. As there are few studies on data extraction and VM categories, it is necessary to have data profiling and VM classification for machines to understand records behavior.

This paper will introduce a new profile-based framework for VM placement for energy optimization, mainly used in small or medium enterprise with privacy data centers. The framework consists of three phases: profiling, tasks & VMs classification and VM placement. Profiling is data features analyzing and provide an overview of the whole data structure. Tasks & VMs classification is to classify tasks and assign the tasks to VMs. Through VM placement with an improved FFD algorithm, a VM placement plan can be generated.

Data used for the experiment is derived from Google clusters. The system samples three different scaled data sets to evaluate the framework performance.

Compared with original FFD, there is a significant improvement from the framework in PM quantity saving, average CPU utilization and a 10% energy saving. As tiny tasks are packaged as a whole, the advantage over original FFD is remarkable when there are huge amount of tiny tasks. The framework has strong technical feasibility and scalability for large-scale data set.

Contributions from the author and supervisors: Yu-Chu Tian designed and supervised the whole project; Zhe Ding conducted detailed design of profiling and profile building; and Siqi Zhang wrote Java programs, undertook

experiments, and obtained the profiles.

2 Introduction

Cloud computing, as a kind of distribution computation, has been widely applied in big data manipulation through the Internet. Data center is a common form to store data for cloud computing, which is a repository that contain multiple servers and communication equipment [1,2], commonly run by large companies or government agencies, have facilities of cables, Internet connections and power supplying, cooling systems and security systems [3].

With the network data volume increasing, energy consumption has become a critical issue for data centers. It is reported that from 2006 to 2014, the U.S. data center electricity use has increased from 62 billion kWh to 70 billion kWh and the figure is predicted to keep increasing slightly to 72 billion in 2020 [4]. Such huge amount of energy occupies roughly 2 percent of total U.S. electricity usage [5] and could support the whole New York residents for more than one year [6]. By the standard of $10-$25 per watt [7], the total cost on U.S. data centers will be over $1000 billion.

However, the utilization of data centers is not optimistic. With an only 12 to 18 percentage average server utilization and 30 percent of servers are faced of the risk of being idled [8], the low utilization has caused a huge waste in both energy consumption and money expense. As a consequence, extra expenditure has been spent on cooling device and the worldwide and 50 metrics tons of carbon dioxide pollution have been emitted annually [6].

Infrastructure and severs are two major sections that lead to huge energy consumption. According to Shuja et al, in the Information Communication Technology Data centers, both of infrastructure and servers had consumed more than 1/3 total energy in 2011[1]. Facebook has designed an improved conventional data center power supply chain system, as shown in figure 2-1 [10], the virtualized data center system (VDC) has decreased power loss from uninterruptible power supplies (UPS). This improvement indicates it is feasible to make changes to the internal structure of a data center, such as virtualizing technology.

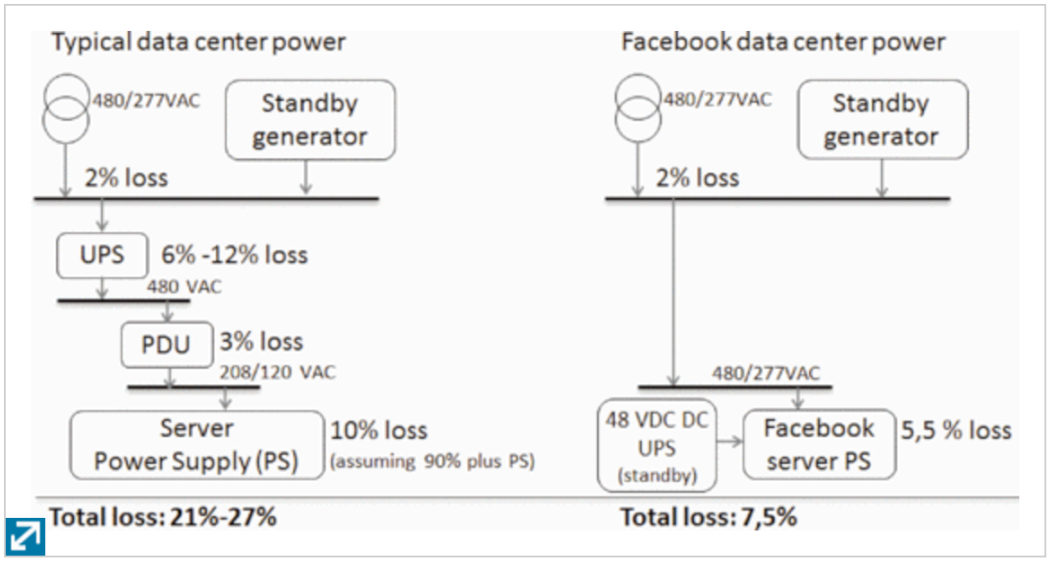

Figure 2-1 Facebook data center design innovation - electrical system [10]

Figure 2-2 shows the structure of sever virtualization. It is a three-layer level with Applications, virtual machines (VMs) and physical machines (PMs).

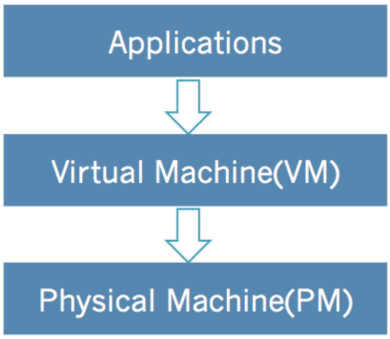

Figure 2-2 Virtualized server structure

In previous work, a graph of the correlation of CPU load and power consumption of VMs (see figure 2-3) had been put forward [11]. With CPU load increasing, the power consumption has a rapid growth from an initial value (160 Watt approx.) and approaches to the peak (215 Watt approx.) slowly. This figure indicates that PMs with higher CPU load will have a better energy efficiency. Enhance CPU load helps to improve VM and PM utilization. With fewer PMs on one data center, the total energy consumption will be decreased.

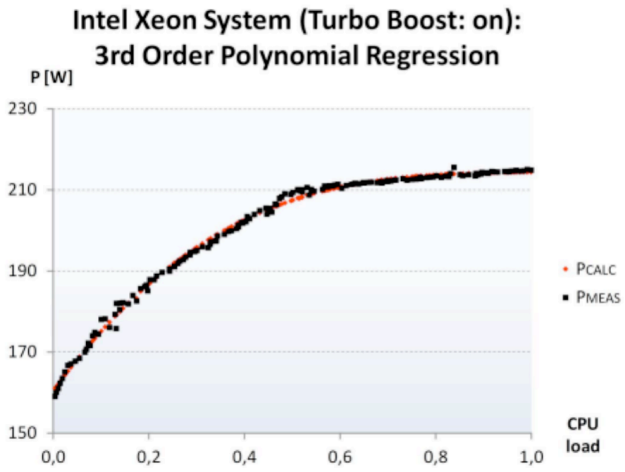

Figure 2-2 Graph of CPU load and Power [11]

Lots of studies have been focused on workload of virtualized data center to optimize server energy consumption, such as the annealing algorithm on physical servers [12], online energy budgeting algorithm eBud to control servers and VMs [13] and workload & energy aware to the clustering-based VM provisioning system [14]. However, these studies are limited to one single layer (usually PMs and VMs) of performance analysis but lack of an overall analyze of the data throughout the whole structure. Actual data is always different from expectation [18]. There should be an extra approach to tell the machine what the data looks like. On the other hand, in VM placement level, most studies concern VM migration, task features has not been addressed so much. Tasks with high lifetime and short lifetime, high CPU request and low CPU request always have different meanings. The allocation strategy should be various. Thus, it is necessary to profile data and classify tasks or VMs ahead of making VM placement plans.

Thus, the purpose of the paper is to develop a methodology includes profiling, VM classification regarding of data features and creating an effective VM placement plan. Through the comparison with ordinal FFD methodology, this approach should have an improvement of PM usage, CPU utilization and energy optimization. The programming language is Java.

The contribution of the paper is a new framework of profile based VM placement with first-fit-decrease (FFD) algorithm. This framework contains three phases:

profiling, Task & VMs classification and VM placement. The system profiles data samples by extracting related features from Google cluster-usage traces. Then classify tasks into different categories, assign tasks to different kinds of VMs and place the VMs to PMs. Through VM placement, a placement plan will be designed to reduce energy consumption. Finally, the performance of the framework will be evaluated by comparing with original FFD, in aspects of PM quantity, average CPU utilization and energy consumption. Computation time is another aspect to be considered, but not as important as the three aspects above. As the data features covered in the framework are not huge, this framework is suit for small or medium enterprise with privacy data centers.

In the following sections, a brief description of virtualized data center and the introduction of source data will be included in prior related work. Project execution or Methodology section will provide a description of the framework. Then follows the outcome of the framework and the evaluation of different scales of data set. Finally, the advantages and limitations of the framework will be analyzed in Discussion section, as well as recommendations for further work.

3 Environmental scan and review of prior related work

3.1 Virtualization

Virtualization is to make a server, storage, network, application etc. available dynamically sharing by IT services. The virtualized objects can be regarded as individual systems. When virtualization is applied on entire machines, the machine becomes a virtual machine (VM), they are systems with completeness hardware. With virtualization, it is convenient to allocate resource and manage the systems without affecting each other, increases effectiveness, flexibility and avoids forging constraints [16, 17]. Virtualization is usually implemented on Windows or Linux system. It has benefits on simplifying the management of IT infrastructures, increasing resource utilization and providing security.

Virtualization can be classified into server virtualization, network virtualization, storage virtualization and application virtualization.

Server virtualization provides the dynamically management of CPU, storage, disk and I/O devices. System can allocate resources to servers by resource priority, in order to decrease the risk of high CPU workload, enhance hardware and software utilization and system efficiency.

Network virtualization enables the virtualized network has the same layer structure, data and services as the physical network. Network virtualization can based on routers or connection devices.

Storage virtualization allows many systems share the same storage service, enable applications and network managing data independently, decrease complexity of aggregating storage resources.

Application virtualization, also known as Desktop virtualization, is to decouple applications from operating system, isolates human-interactive activities (interface, keyboard operation, audio I/O, etc.) with computation logic.

3.2 VM placement

VM placement is the process to map VMs on PMs, selecting which VM should be placed to each PM [20], considering of PM capacity and CPU utilization. It is one of the challenging problems for data centers in cloud infrastructure management.

In VM placement, each arriving task must be set to one VM and each VM must be placed to one PM. Empty VMs can be destroyed or kept for later use. Empty PMs can be switched off [21].

VM placement is a kind of bin-packaging problem and first-fit decreasing (FFD) algorithm is a common used algorithm in VM placement. With FFD algorithm, VMs and PMs are sorted by size in descending order before the placement. When a VM arrives, the system will check each PM and place the VM on the first PM with enough capacity.

PM quantity, PM energy consumption, resource consumption and launching time are the key aspects that affect VM placement. Combing tasks to one VM is an effective method for saving resource consumption. However, large-sized VMs may restrict launching time [21] and increase PM quantity. Hence, it is important to find a balance among these aspects.

3.3 Data profiling

Data profiling is the process to examine whether the source data describes the values of source data accurately. Profiling improves data quantity, helps machines to understand data structures and dependences, build data quantity rules and discovers hidden relationship between data features [18]. With profiling, it is convenient to decide whether source data is suitable to use.

3.4 Source data

The source data used in the project derived from Google cluster-usage trace logs. It describes the workload of physical machines of each cluster in Linux system

[15]. All data is stored in different kinds of csv files and lines are ordered by timestamp of each event. The csv files include physical machine information, jobs and tasks information and usage (as shown in figure 3-1). In the physical machines, each job consists of several tasks and each task consists of several procedures. Workload is measured every 300 seconds. Tasks run in the current timeslot will be recorded.

351634289	0.02225	0.01062	0.01149	4.48E-05	0.0003109	0.01157	0.0002241	9.54E-07	0.09399
2.899E+09	0.00251	0.007812	0.008728	0.0001993	0.0003166	0.008713	0.001417	0	0.02985
295014299	0.04712	0.011	0.01201	0.0001431	0.0002995	0.01138	0.002163	6.68E-06	0.09631
3.721E+09	0.0343	0.07593	0.07776	6.77E-05	8.30E-05	0.07605	0	9.54E-07	0.06091
3.721E+09	0.03699	0.07605	0.07776	1.72E-05	3.72E-05	0.07605	0	9.54E-07	0.06262
660272	0.01956	0.03314	0.0343	5.91E-05	0.0001307	0.03503	0.0002146	9.54E-07	0.4092
351613851	0.0174	0.03375	0.03503	1.72E-05	6.68E-05	0.0351	4.48E-05	9.54E-07	0.06604
6566576	0.008118	0.03229	0.03333	0.0001183	0.000226	0.0351	0.0005207	9.54E-07	0.06274
765536	0.03857	0.01041	0.01141	0.0001945	0.0005341	0.01067	6.58E-05	2.86E-06	0.1758
1.44E+09	0.04187	0.01039	0.01151	0.000226	0.0005569	0.01115	0.000205	2.86E-06	0.09436

Figure 3-1 Source data of PM machine events table

Physical machine information is stored in machine events table and machine attributes table respectively. These two tables stores fields like timestamp, machine ID, event type, CPU capacity, memory capacity and attribute values.

Jobs and tasks information are stored in job events table and task events table, task constraints table and task resource usage table. These tables store fields like timestamp, job ID, task ID, event type, priority, request for disk space and CPU. Job event table and task events table describe the lifetime and other attributes of jobs and tasks.

Figure 3-1 displays a model of state transitions of jobs and tasks progresses in VMs [15]. After the task is submitted, it will be put into pending queue and waiting for schedule, then starts running. If the task is killed, finished or failed to dead, it can be submitted again and wait for scheduling. The model indicates that tasks for allocation to VMs can be repetitive. The coming system adopts profiling for optimizing allocating the request of repetitive tasks or VMs, and provides task allocations and VM allocations for the given data with First Fit Decrease (FFD) algorithm. The main idea of FFD is firstly order the VMs or tasks by their size.

Then, for each VM or task, search from the first available space and conduct allocation, aiming at reducing the number of migration.

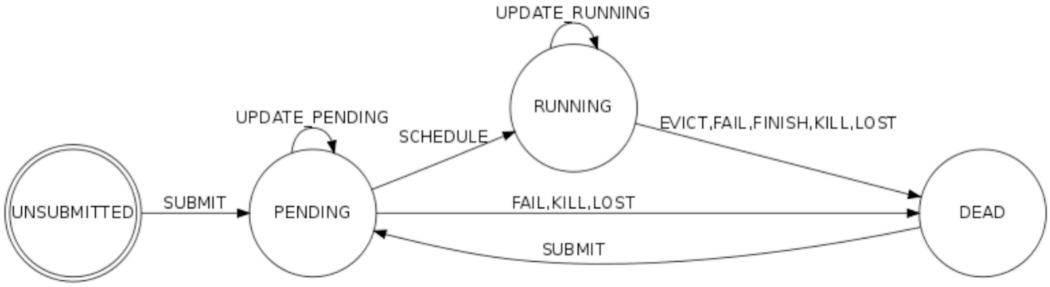

Figure 3-1 State transitions for jobs and tasks [15]

Task event table and task resource table will be used for data profiling. Resource usage is the record of task usage in each measurement period.

Google-cluster trace data is stored on Application Level. It is non-virtualized user data. Meanwhile, the amount of origin data is very huge (around 10 million lines each file). It is essential to extract part of specific lines for analyzing. The detained procedure will be explained in the next section.

4 Project Methodology and Approach

This section provides an overview of the project, discusses the approach of the project, and provides a detailed view of the three phases of the framework and evaluation principle.

4.1 Project overview

This project is written by Java, aiming at developing a framework to profile the data logs for VMs and make VM placement plan. The performance of framework should be evaluated from PM quantity, PM utilization, energy saving and time cost. As the project is depended on FFD algorithm and have improvements on several areas, the results will be compared with original FFD.

The project includes a framework and an evaluation. The structure is shown in figure 4-1.The framework includes three phases: Profiling, tasks & VMs Classification and VM placement, aiming at making VM placement plan with the classified profiling data. Evaluation is used for evaluating the performance of VM placement plan, considering PM quantity and the average CPU utilization of PMs in each timeslot, energy consumption and the computation time.

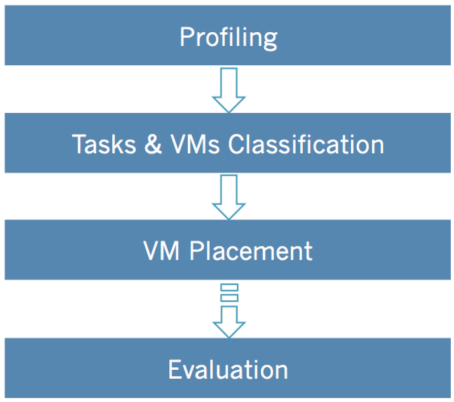

Figure 4-1 project structures

4.2 Profiling

The first phase of the framework is profiling. It is used for optimizing allocating the request of repetitive tasks or VMs. In this phase, the system analyzes 18 task resource usage table files and the related task event table files to make total duration of the tasks in these files around 24 hours, samples data centers into

three scales: Large scale, medium scale and small scale.

The first part of this phase is VM profiling, as well as data centers sampling. Large-scale data set stores the data extracted from origin data files one line in every 200 lines. The line intervals for medium-scale data set and small-scale data set are 400 and 1000. Data attributes used for profiling include start time, end time, job ID, task ID from and CPU request (see table 4-1). The time span of the data is between 50700000000 and 141000000000, about 24 hours from the beginning to the end. Through VM profiling, it is directly to have a view of total CPU request in each time slot. The figures can help to verify sampling correctness and predict PM quantity for VM placement.

Field	Explanations	Source table
Start time	Start time of the task, in the form of million second	Task resource usage
End time	End time of the task, in the form of million second	Task resource usage
Job ID	The ID of the job, numeric	Task resource usage
Task ID	Task index of the job, numeric	Task resource usage
CPU request	Request for CPU cores	Task events

Table 4-1 Details of analyzed data features

The second part is job profiling. Job profiling is to collect job duration and task quantity of each job. Tasks can be considered as repetitive tasks when the span of end time in the previous time slot and the start time in the next time slot is within 300 seconds. If the duration of all tasks in one job covers the whole time slot, the job can be regarded as permanent jobs. Permanent job information will be stored into a file and it is used for task classification in next phase.

4.3 Tasks & VM Classifications

The second phase is tasks & VMs classification. Tasks are the minimum units of the data. Different tasks have different lifetime and different CPU request. Vasudevan et al. has cited a special profile-based task assignment methodology

[9]. The main purpose of this phase is similar to the ideas from the methodology, aiming at classifying tasks into different categories and assigns all the tasks to different sized VMs.

In the pervious phase, a permanent job list is generated. Permanent jobs contain a group of tasks. As the lifetime of the job is long, these kinds of tasks are usually key services with relative high CPU capacity that should be running in the whole procedure without interruptions. They should be placed in stable VMs. Meanwhile, there is a certain amount of tasks with tiny CPU request (such as 6.247E-4) and brief lifetime. It is unrealistic to create VMs with such tiny CPU capability. If each of the tiny tasks is assigned to a larger sized VM, there will be lots space left idled that decreases CPU utilization. Thus, these tasks can be packaged together until the total CPU request reaches the smallest VM CPU capacity. The rest of the tasks are normal tasks. These tasks have medium CPU request and modest lifetime.

Thus, the whole tasks can be divided into three categories:

- Permanent tasks: tasks in permanent jobs
- Tiny tasks: task with CPU request less than 0.015
- Normal tasks: other tasks

There are several kinds VMs with different CPU capacities to place tasks. The minimum CPU capacity is 0.015, as same as the threshold of CPU request of tiny tasks. The highest CPU request of the tasks is 0.4026, so the maximum CPU capacity of VM is set to 0.45. All kinds of VMs are shown in table 4-2. Each task in permanent tasks and normal tasks is assigned to one VM based on its CPU request. Each tiny task package is assigned to one tiny VM.

	Huge VM	Large VM	Medium VM	Normal VM	Small VM	Tiny VM
CPU Capacity	0.45	0.3	0.15	0.1	0.045	0.015

Table 4-2 VMs with different CPU capacity

The procedure of tasks and VMs classification is shown as the figure below:

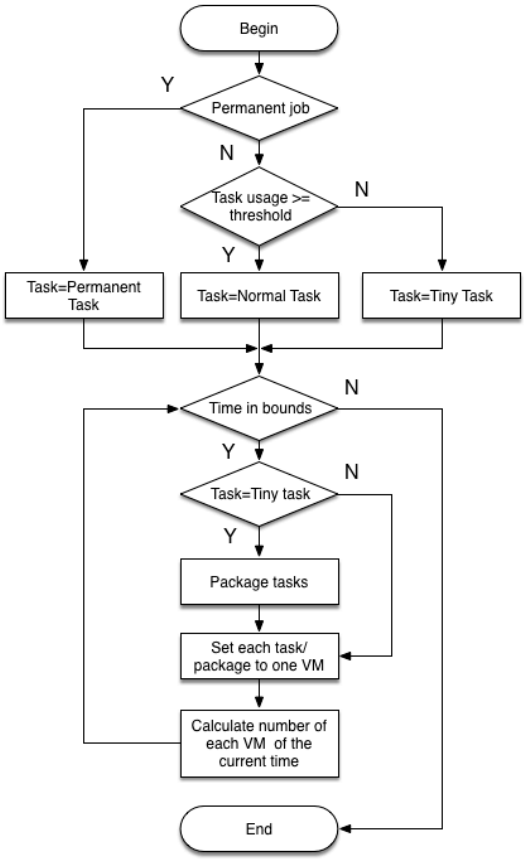

Figure 4-2 Tasks & VMs classification procedure

The process is described as follows:

1. Acquire job ids from the permanent job file and read each line from the tasks files.
2. Tasks with a permanent job id will be regarded as permanent tasks.
3. For the rest of the tasks, if their CPU request is less than the threshold, 0.015, they will be classified as tiny tasks. Otherwise, the tasks are normal tasks.
4. For the tasks in each time slot, package tiny tasks.
5. Assign each of permanent tasks, normal tasks and packages to one VM.
6. Calculate the each VM quantity at the end of each time slot.

4.4 VM Placement

The third phase is VM placement. The aim of this phase is to place the VMs generated from the previous phase to PMs.

Considering of the variable sizes of CPU requests, the PMs in data centers should

have different CPU capacities. The configuration of PMs and PM quantity for the three scaled data set are shown in table 4-3 and table 4-4:

	High Performance PM	Medium Performance PM	Low Performance PM
CPU Capacity	1.5	1.0	0.5
Energy efficiency	High	Medium	Low
Max Power	220	250	280
Min Power	160	160	160

Table 4-3 PM configurations

	High Performance PM	Medium Performance PM	Low Performance PM
Large-scale	10	15	300
Medium-scale	6	9	200
Small-scale	0	2	100

Table 4-4 PM quantities for three scaled data sets

CPU capacities of different PMs are normalized as 1.5, 1.0 and 0.5. The amount of high performance PM and medium performance PM are fixed while the quantity of low performance PMs can be any. High performance PM and medium performance PM are used for placing permanent VMs. The PM energy efficiency is co-related to the performance. According to table 4-3, the min power (160) is acquired from figure 2-2.

The placement period is 300 seconds. Data features, VM and PM configurations for these two approaches are the same. The differences between the two approaches are described as follows:

1. In new framework, permanent tasks placement is prior to normal tasks and tiny tasks placement, while there is no task priorities in original FFD.
2. In original FFD, there is no tiny task packaging. One tiny task is assign to one VM.

Placing permanent tasks prior of other tasks leads to permanent tasks being placed to high performance PMs. Meanwhile, tasks in permanent jobs usually have high CPU request, high CPU capacity PMs

The procedure of VM placement of new framework is shown in figure 4-3:

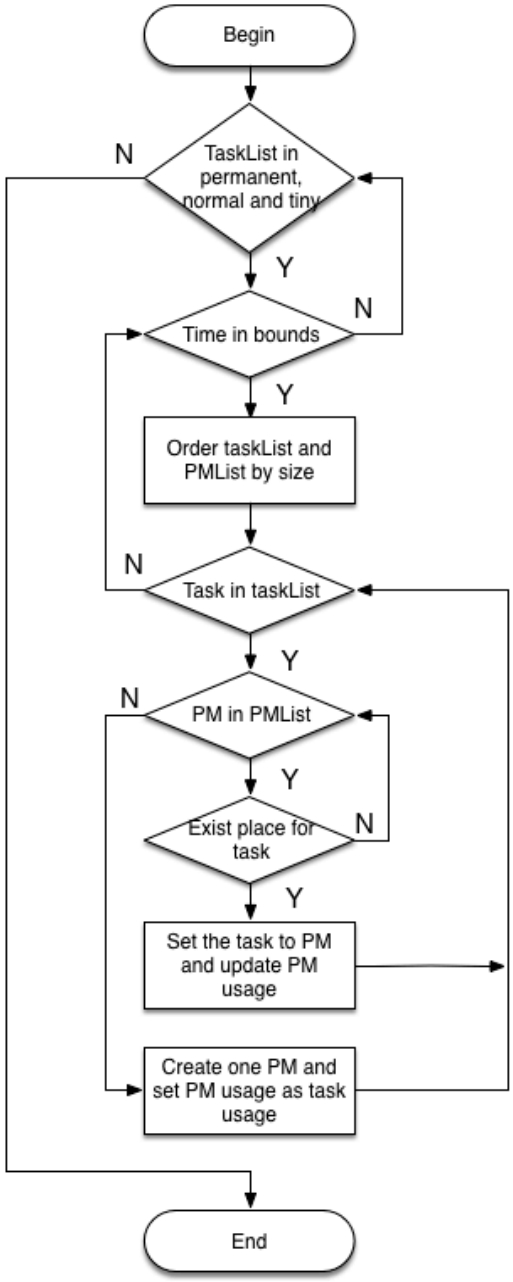

Figure 4-3 Procedure of VM placement

The process is described as follows:

1. Place permanent tasks. For each time slot, order VMs and PMs by size from the largest to the smallest. The size of PM refers to its free CPU capacity.
2. Place VMs to PMs with FFD algorithm. For each VM, check the free capacity of each PM and place the VM to the first-fit PM. If there is no free capacity for all existing PMs, create a new PM and place the VM on the PM.

3. After placing all VMs with permanent tasks, place the rest VMs with normal tasks and tiny tasks.
4. Order VMs and PMs.
5. Place VMs to PMs with the same method.

Finally, a VM placement plan can be generated. According to the information of VM placement, the PM quantity used in each time slot can be acquired directly. Utilization of each PM can be calculated by total usage dividing total capability. Then calculates an average value as the average PM utilization. These data will be used for evaluation.

4.5 Evaluation

The final step is to evaluate the performance of the plans. The evaluation standard is planned to include two aspects: feasibility and scalability. Feasibility refers to whether the framework could have an obvious improvement of energy consumption saving and PM quantity saving, as well as a short computation time. The energy consumption is described as a sum of CPU power of each PM in each time slot. Computation time is not as important as the other aspects, as long as in an acceptable range.

Scalability refers to whether the system could work well with various scaled data set. More nearer the relationship between data set scale and processing time to linear, the stronger scalability the framework will have.

At last, diagrams of the PM quantity and PM utilization in every time slot and power consumption will be generated. Through the comparison between the two approaches, the one with fewer PMs, higher PM utilization and less time cost will be the higher effective approach.

4.6 Summary

This section describes the approaches of each phase of the profile-based framework, provides a clear structure of profiling, tasks classification and VM placement, emphasize the improvement from original FFD. Then describes the steps of evaluation, states several key features to be analyzed in the next section.

5 Outcomes or Results

This section includes data features analyzing, VM placement plan analyzing and framework performance evaluation of the three scaled data center. The program is written in Java (JDK 1.8.0.91), and run in a desktop with Intel Core2 Q6700 3.4GHz &32GB DDR4 2666 RAM.

5.1 Profiling

In profiling phase, the system generates figures of the distribution of total CPU request and job numbers of the whole time slot. These figures are used for predicting data features, and help to verify the results of tasks & VMs classification and VM placement plan. In all figures, time slot is represented in the unit of second.

5.1.1 Large-scale experiment

Figure 5-1 shows the distribution of timely peak CPU usage of origin data set and sampled data set. Data of peak CPU usage of original data sets is in blue line and that of sampled data sets is in orange line. Both of the two lines are in the same trend, which indicates the sampled data is suitable to use.

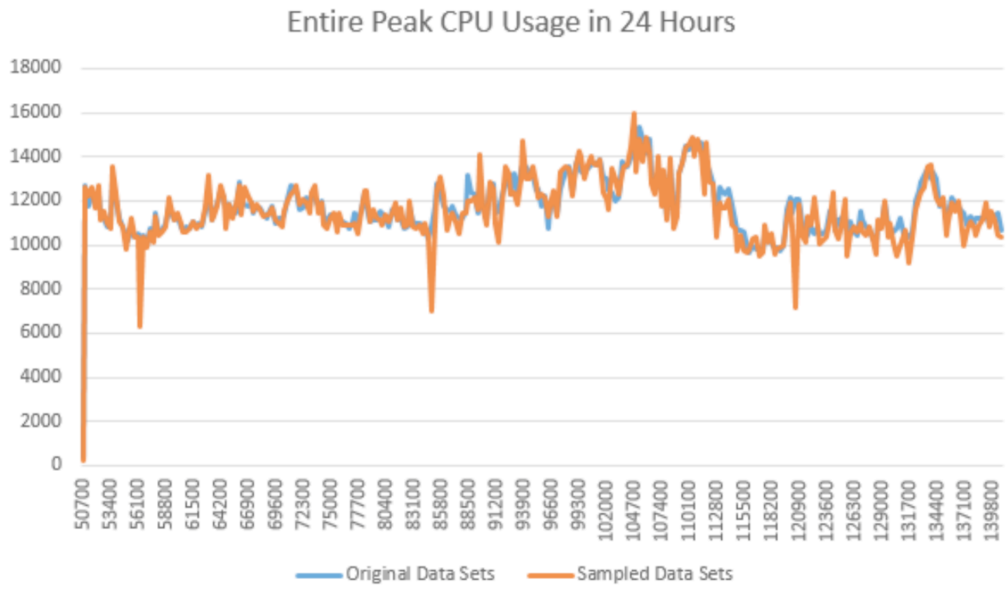

Figure 5-1 Entire peak CPU Usage in 24 hours

Figures 5-2 shows the timely total CPU request. It can be seen from the figure the during the whole time span, most of CPU requests fall in 20 to 40 and the total CPU requests in few time slots are extremely high. According to table 4-4, with the quantity of three kinds of PMs and total CPU request, a rough number of PMs to be used in each time slot can be predicted.

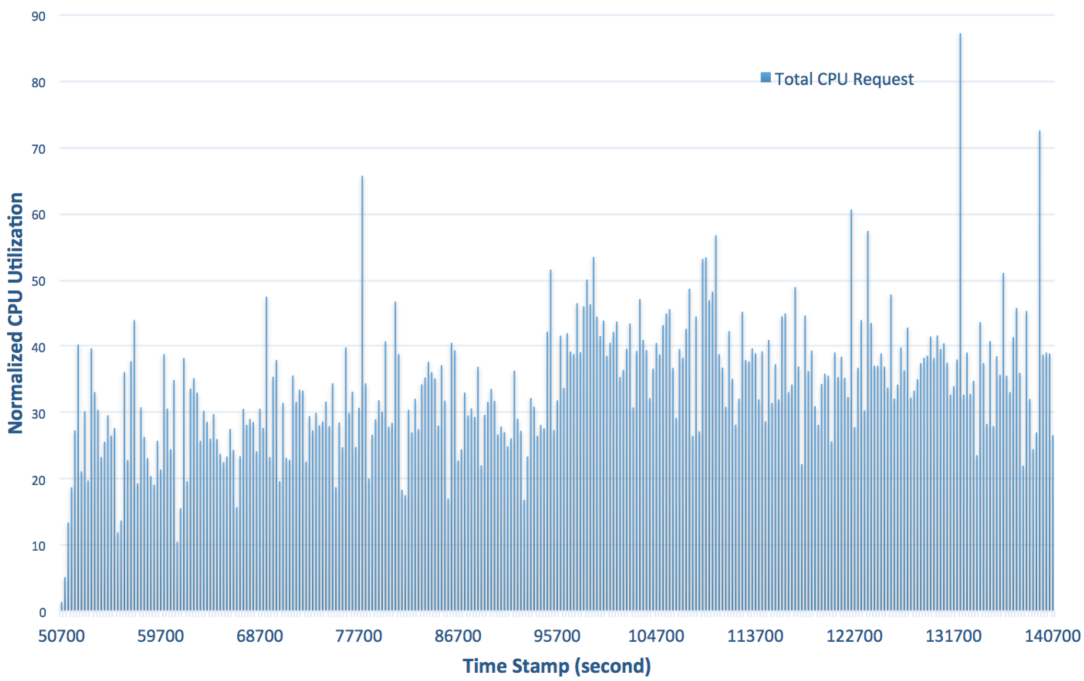

Figure 5-2 Timely CPU request of large-scale data

Figure 5-3 shows number of jobs and permanent job distribution in each time slot. Jobs are in blue lines and number of permanent jobs is in orange line. Job numbers range between 344 to 460 in 24 hours. As analyzed in this phase, the total number of permanent jobs is 101. From figure 5-3, it can be seen that all permanent jobs run in each time slot. The number of permanent job is kept stable.

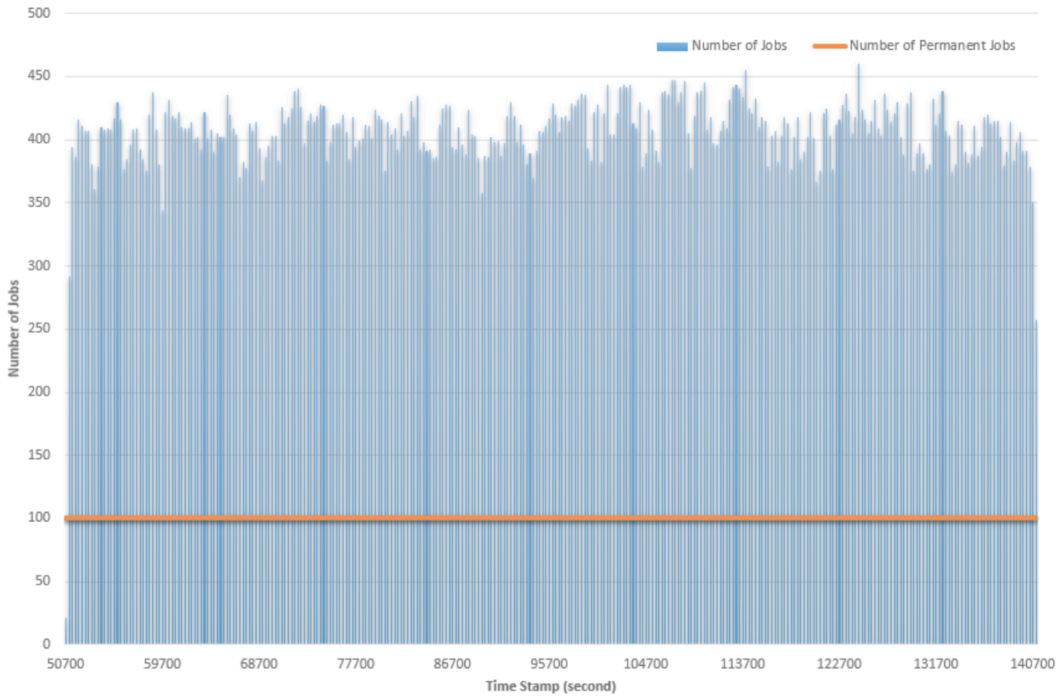

Figure 5-3 Total job number and permanent job number in large-scale data center

5.1.2 Medium-scale experiment

The data of total CPU request and total job number of medium scale data center is displayed in figure 5-4 and figure 5-5. Total CPU request ranges between 21 and 37 in 24 hours. The number of permanent jobs is as same as that in large-scale data set.

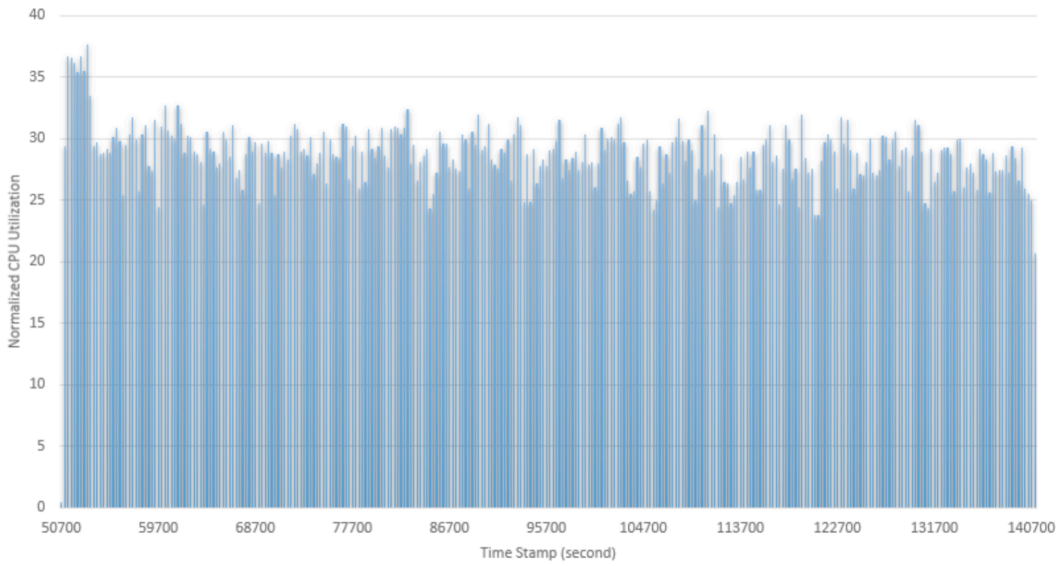

Figure 5-4 Total CPU request in medium scale data center

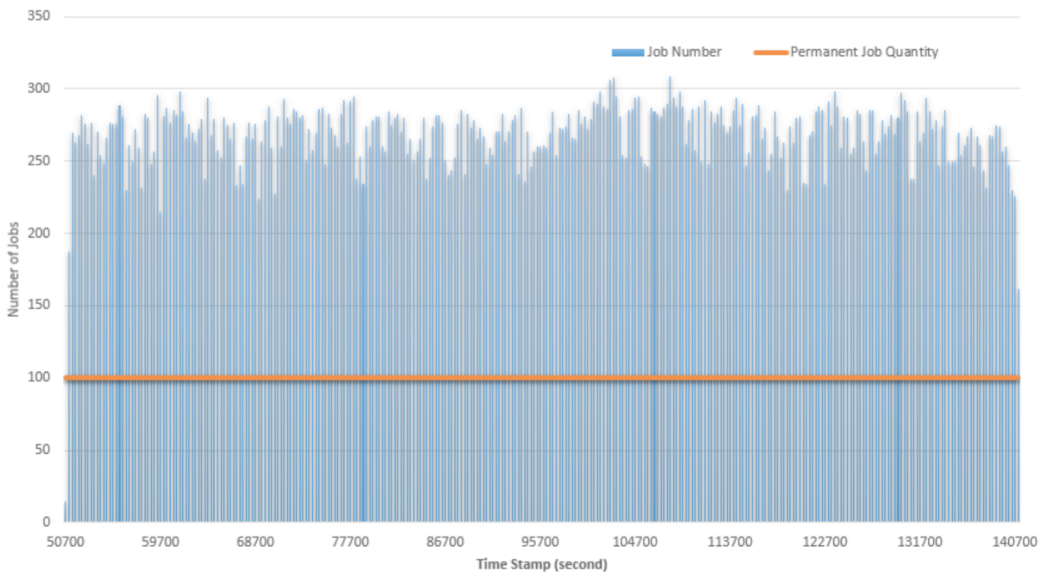

Figure 5-5 Total job number and permanent job number in medium scale data center

5.1.3 Small-scale experiment

The data of total CPU request and total job number of small-scale data center is displayed in figure 5-6 and figure 5-7. Total CPU request ranges between 11 and 21 in 24 hours. The number of permanent jobs is as same as that in large-scale data set and medium-scale data set.

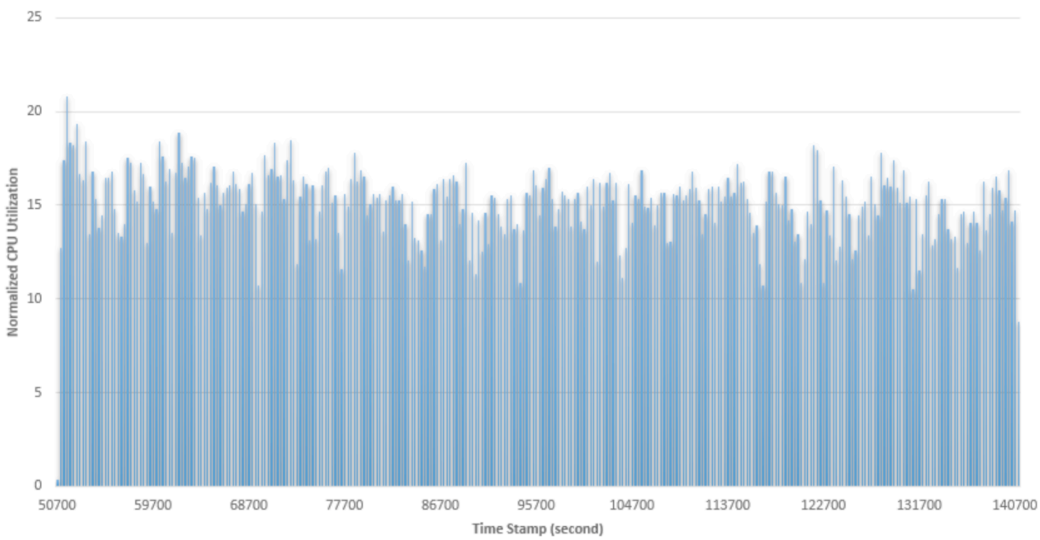

Figure 5-6 Total CPU request in small-scale data center

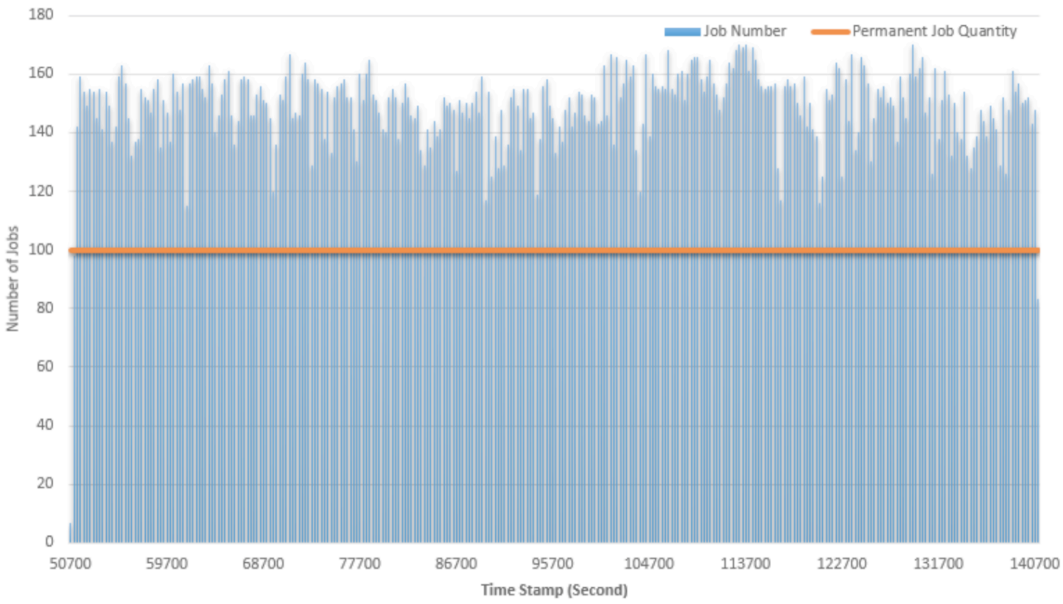

Figure 5-7 Total job number and permanent job number in small-scale data center

5.2 Task & VMs classification

In this phase, the system classifies tasks into three categories: permanent, normal and tiny in each time slot. Each permanent task and normal task is assigned to one VM while tiny tasks are packaged and each package is assigned to one VM. Finally, the classified VM numbers of three data sets are acquired.

According to table 4-2, there are five scales of VMs with different CPU capacities: 0.45, 0.3, 0.15, 0.1, 0.045 and 0.015. Each task is assigned to the VM with the nearest CPU capacity to its CPU request. Tasks have a high CPU request that above 0.3 are the fewest and tasks have the CPU request below 0.015 are the most, then follows by the tasks with CPU request between 0.1 and 0.015. Results for tiny task packaging are explained below in different scale data set experiments.

The pseudo code for this tasks and VMs classification is described in Algorithm 1.

Algorithm 1 Separation & assignment algorithm.

```
procedure CLASSIFICATION
    for all tasks do
        if Related job is in permanent jobList then
            task = permanent task
        else
            if task usage ≥ tiny threshold then
                task = normal task
            else
                task = tiny task
            end if
        end if
        for all time slots do
            for all permanent and normal tasks do
                Assign separate VMs to each of them
            end for
            for all tiny tasks do
                if this task repeats regularly then
                    Pack task
                    Calculate package number
                end if
            end for
        end for
    end for
end procedure
```

Since the difference between the new framework and original FFD is mainly depended on packaging tiny tasks or not, this paper will only analyze tiny tasks in the next sections.

5.2.1 Large-scale experiment

There are 95331 tasks altogether tasks in large-scale data set in 24 hours. Figure 5-8 shows the number of tiny tasks in each time slot. There are obvious more tiny tasks between time slot 92700 and time slot 111600 than the number of tiny tasks in other time slot. The number of tiny tasks ahead of that time span (time slot 50700 to 92700) is about 400, while the tiny task quantity after that time span (time slot 111600 to the end) is a litter fewer, less than 200. Figure 5-9 shows the number of packages converted from the tiny tasks in figure 5-8. After packaging, the VM numbers has reduced to 10% (Assuming that each tiny task belongs to one VM before packaging). Similarly, in time slot 92700 to 111600, there are significantly more packages than the other time slot.

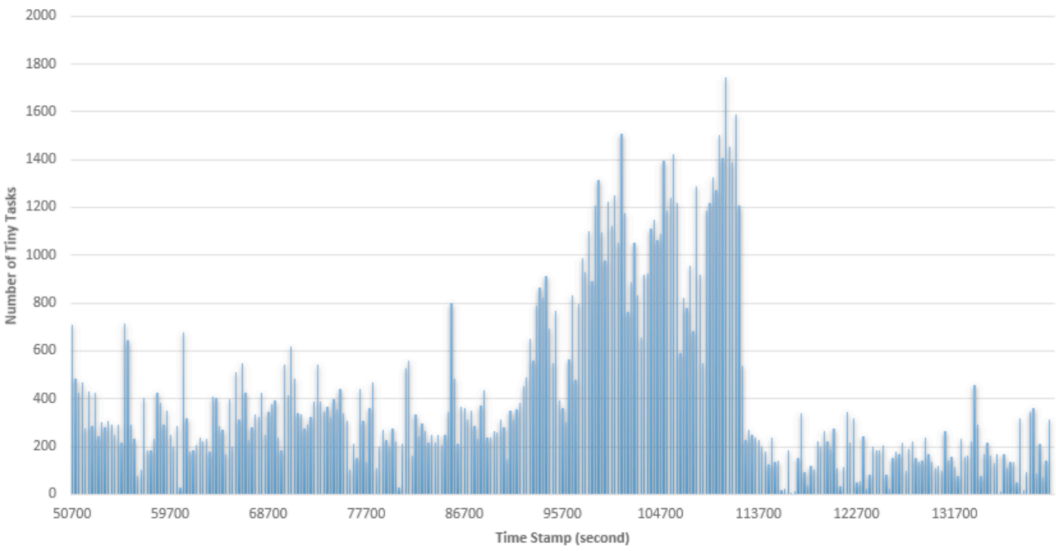

Figure 5-8 Large-scale tiny tasks packing result for number of tiny tasks

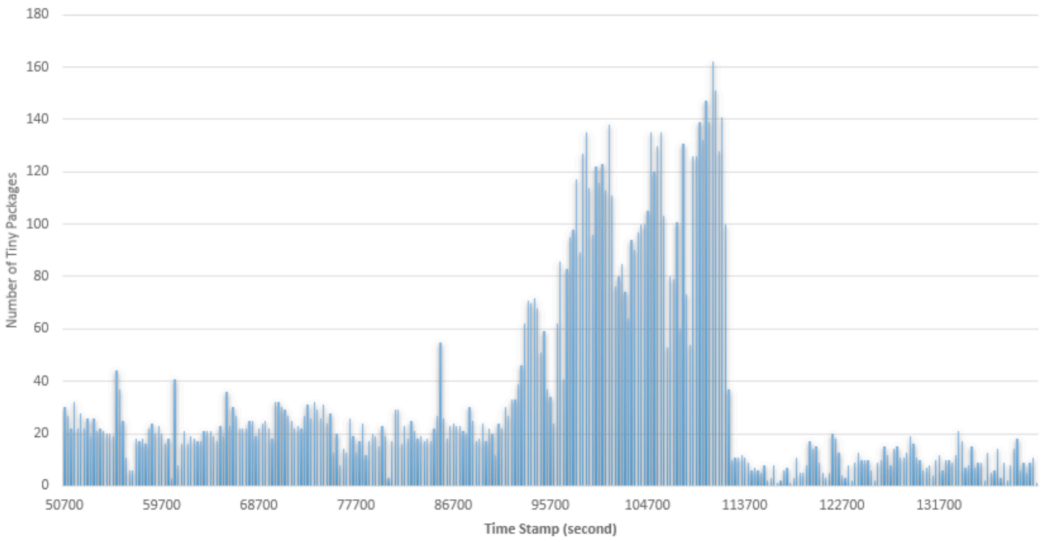

Figure 5-9 Large-scale tiny tasks packing result for number of tiny packages.

5.2.2 Medium-scale experiment

There are 47484 tasks altogether tasks in medium-scale data set in 24 hours. Figure 5-10 shows the number of tiny tasks in each time slot and figure 5-11 shows the number of packages converted from the tiny tasks in figure 5-10. Similar to large-scale experiment, there is a significant raise of tiny tasks number and tiny packages in time slot 95700 to time slot 111300 of the two figures separately.

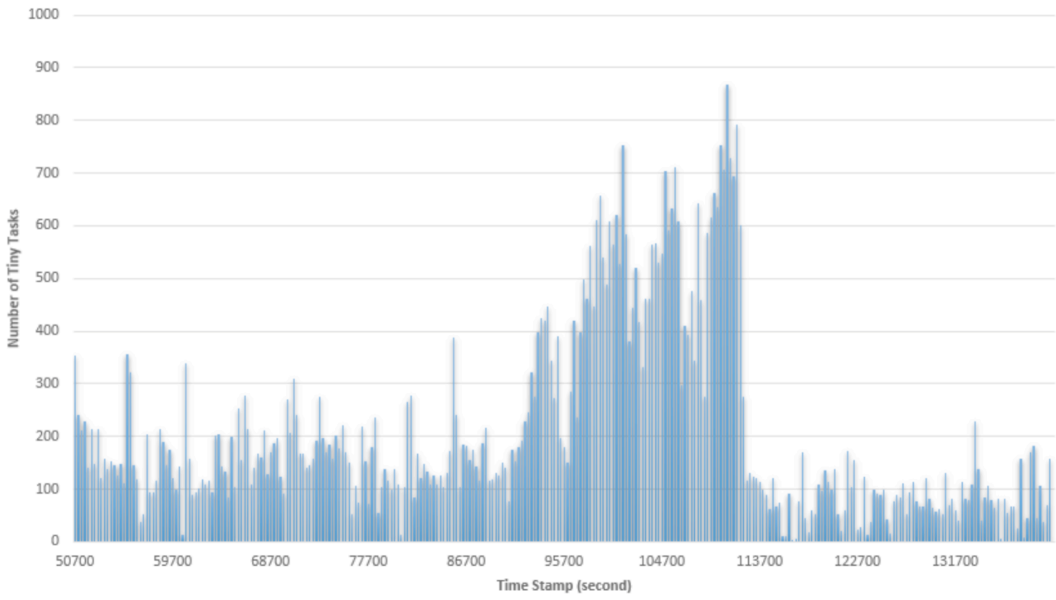

Figure 5-10 Medium-scale tiny tasks packing result for number of tiny tasks

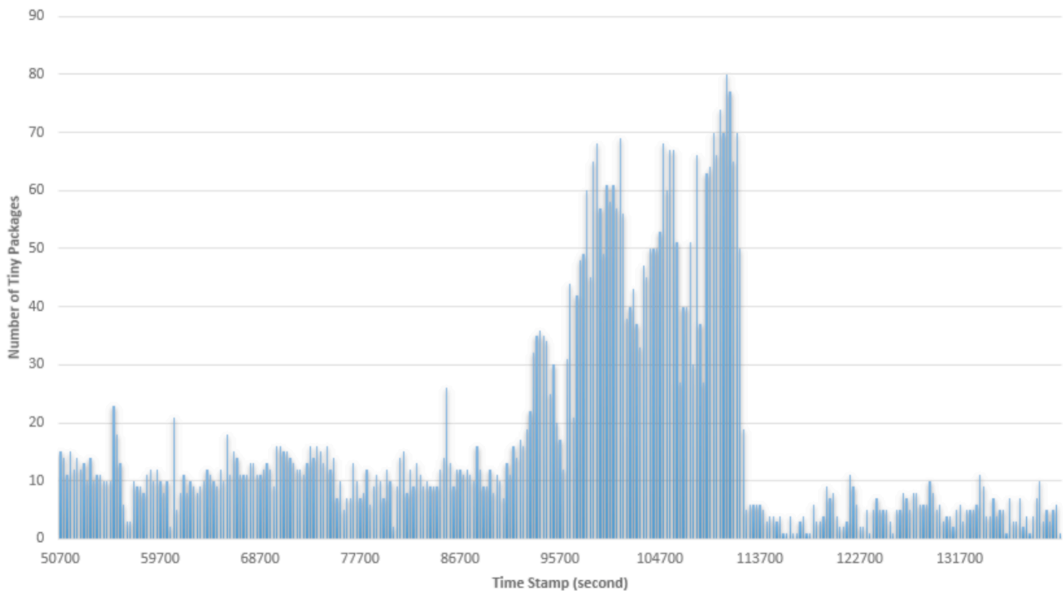

Figure 5-11 Medium-scale tiny tasks packing result for number of tiny packages

5.2.3 Small-scale experiment

There are 18994 tasks altogether tasks in medium-scale data set in 24 hours. Figure 5-12 shows the number of tiny tasks in each time slot and figure 5-13 shows the number of packages converted from the tiny tasks in figure 5-12. Similar to large-scale experiment and medium-scale experiment, there is a

significant raise of tiny tasks number and tiny packages in time slot 92700 to time slot 111600 of the two figures separately.

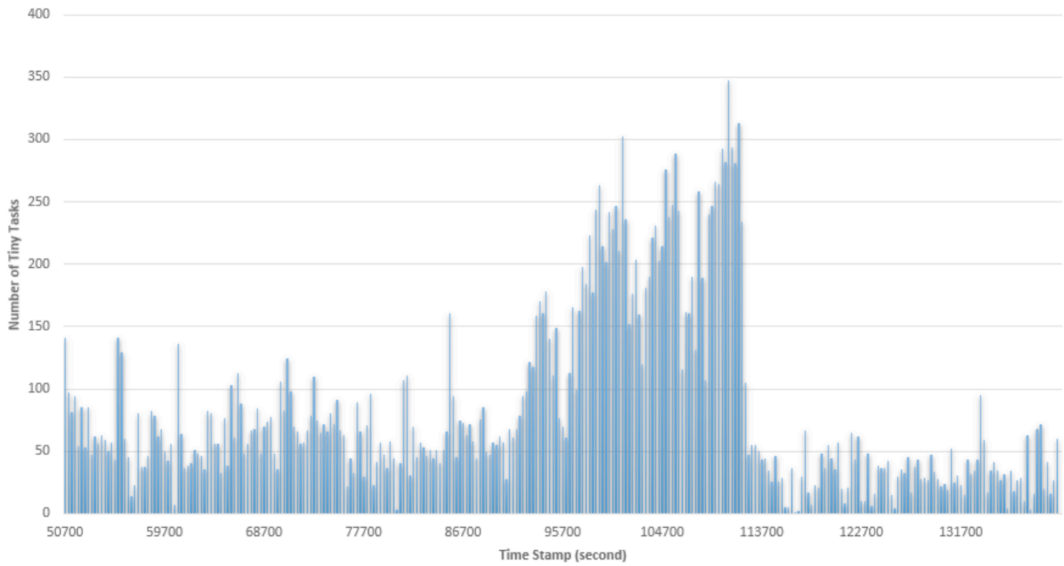

Figure 5-12 Small-scale tiny tasks packing result for number of tiny tasks

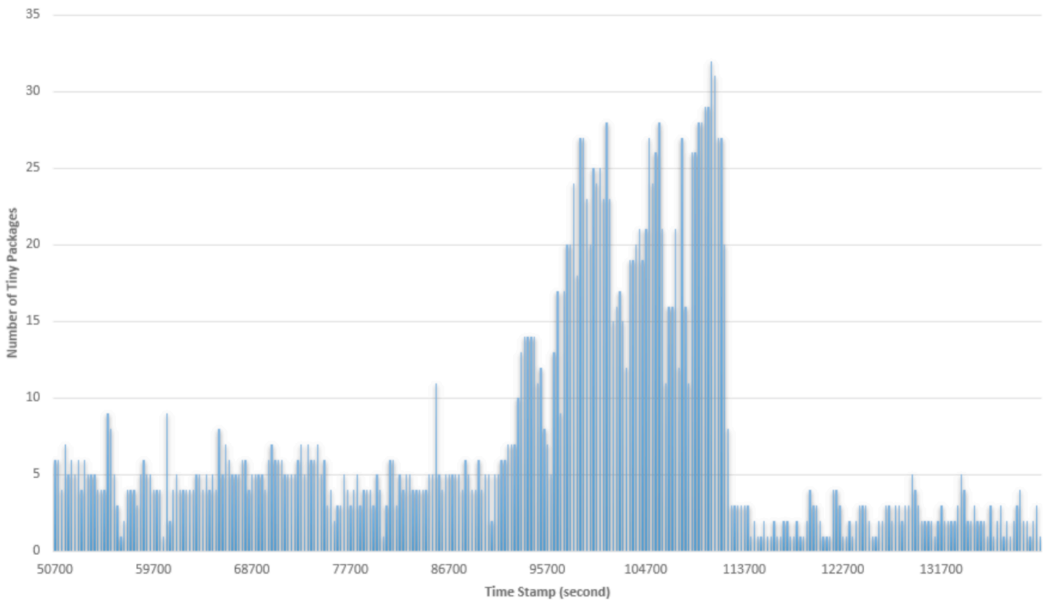

Figure 5-13 Small-scale tiny tasks packing result for number of tiny packages

For original FFD placement to be explained later, there is no tasks classification and all tasks are assigned to one proper VM based on its CPU request. The results of classification are stored into different csv files by task type. Each line refers to one time slot, representing how many VMs are set in this time slot.

5.3 VM Placement and PM quantity evaluation

In this phase, through placing the VMs generated from tasks & VMs classification, the number of PMs in each time slot and CPU usage has been generated. With the data, the average CPU utilization can be calculated. In addition, energy consumption is calculated by power formula. The performance of the framework and original FFD are generated in the same figures for comparison. Results of three data sets are explained separately.

The pseudo code of VM placement is described in Algorithm 2:

Algorithm 2 VM placement algorithm as FFD.

```
procedure FFD
    for all taskList in permanent, normal, and tiny VMs do
        for all t=time slot do
            Order VMs and PMList by size
            for all VM in taskList do
                for all PM in PMList do
                    if PMUsage + TaskUsage ≤ maximum PM capacity then
                        PMUsage = PMUsage + VMUsage
                    end if
                end for
                if This VM has not been placed yet then
                    Create a new PM
                    PMUsage = VMUsage
                end if
            end for
        end for
    end for
end procedure
```

5.3.1 Large-scale experiment

Figure 5-14 shows the number of PMs in each time slot of large-scale data set. PM quantity of original FFD are in orange lines and PM quantity of new framework are in blue lines. During time slot 92700 and 111600, there are more PMs in original FFD than the new framework. In comparison, there is no remarkable difference between the PM quantities of these two methods in other time slots. The highest PM quantity is above 200 and the lowest is 6, which is similar to the predication in profiling phase. In most of the time, the PM quantity is no more than 100.

According to figure 5-2, the timely CPU requests are about 30 to 40 in the first half of time span. If all CPU utilizations are 100%, there should be 20 to 40 PMs at least. The actual PM quantity of that time span is about 40 to 100. It is impossible to make all CPU utilization as 100% and most VM utilizations are also less than 100% when assigning tasks to VMs. Hence, the result is acceptable.

The trend of timely PM quantity of the new framework is similar to trend of total CPU request (as shown in figure 5-2), which indicates there are no significant errors in VM placement.

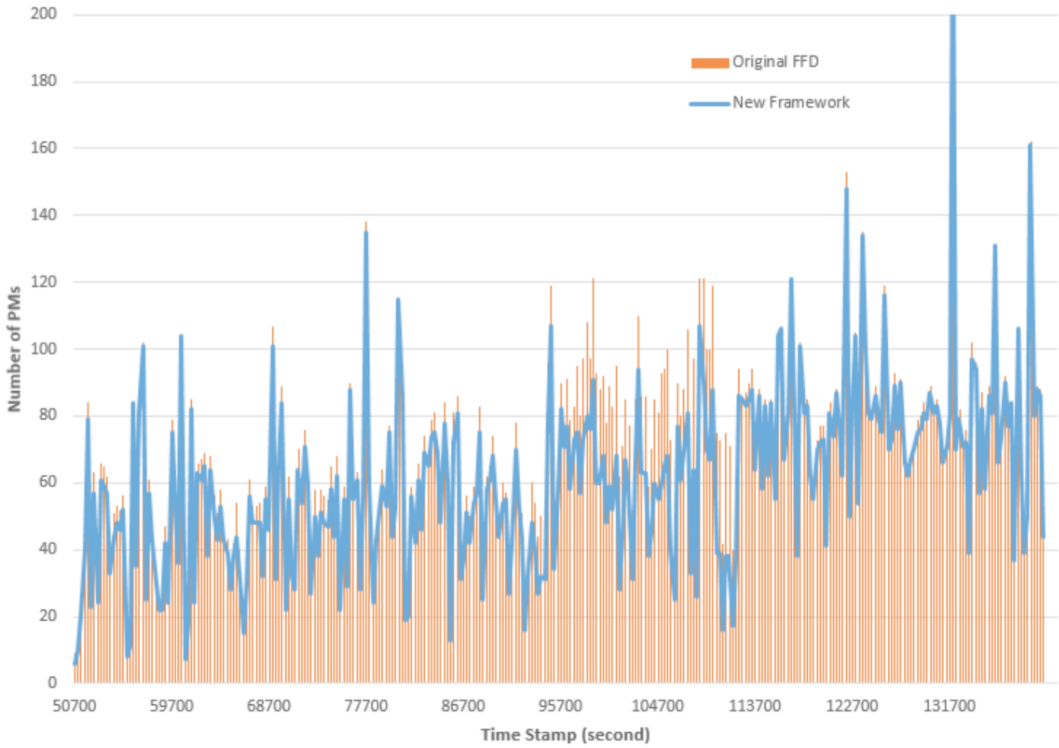

Figure 5-14 Large-scale VM placement experiment result: timely PM quantity

Figure 5-15 shows PM quantity overlap. In the figure, Y-axis is the PM quality gap, which represents the difference between PM quantity of original FFD and new framework. Numbers in y-axis below zero indicates the new framework uses less PMs. The new framework saves a great number of PMs in from time slot 92700 to time slot 111600 and the highest figure is nearly 40. According to figure 5-8 and figure 5-9, there are apparently more tiny tasks (packages) in that time zone. On the other hand, as shown in figure 5-2 and figure 5-3 the total amount of CPU request and the total amount of jobs have not exceeded the other time zone so

much. Thus, the reason for the significant saved PM amounts is due to the huge number of tiny tasks. More narrowly, tasks packaging is an effective way to decrease PM quantity.

In the other time slots, though the amounts of timely saved PMs are not as high as that area, the new framework could still save several PMs at most times. Meanwhile, the saved PM quantity ahead of the time zone with huge tiny tasks amount is slightly more than that of time slot after it. The trend is just as same as the trend of tiny tasks quantity. Again, this phenomenon indicates PM numbers saving has a positive correlation with tiny tasks amount.

The figures are flat occasionally and only in two time slots, the PM numbers of new framework exceed that of original FFD.

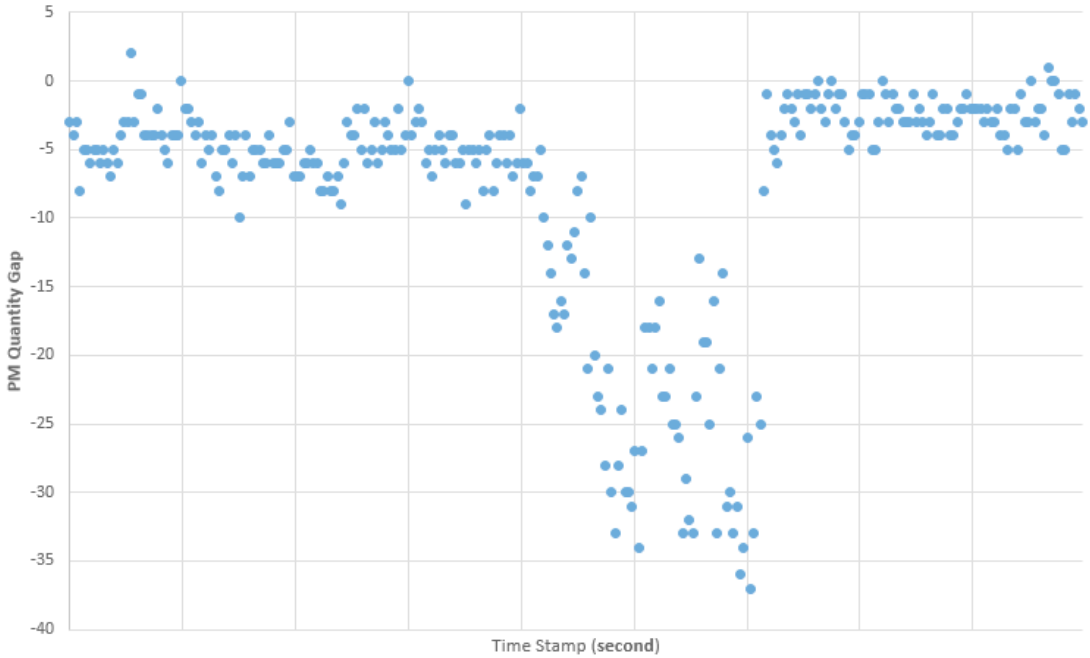

Figure 5-15 Large-scale VM placement experimental result: PM number overlap

5.3.2 Medium-scale experiment

Figure 5-16 shows the number of PMs in each time slot of medium-scale data set. Similarly, during the time span with huge amount of tiny tasks, there are more PMs in original FFD than the new framework and there is no remarkable difference between the PM quantity of these two methods in other time slots. The PM quantity of new framework range from 4 to 78 and most PM quantity

ranges between 4 and 40.

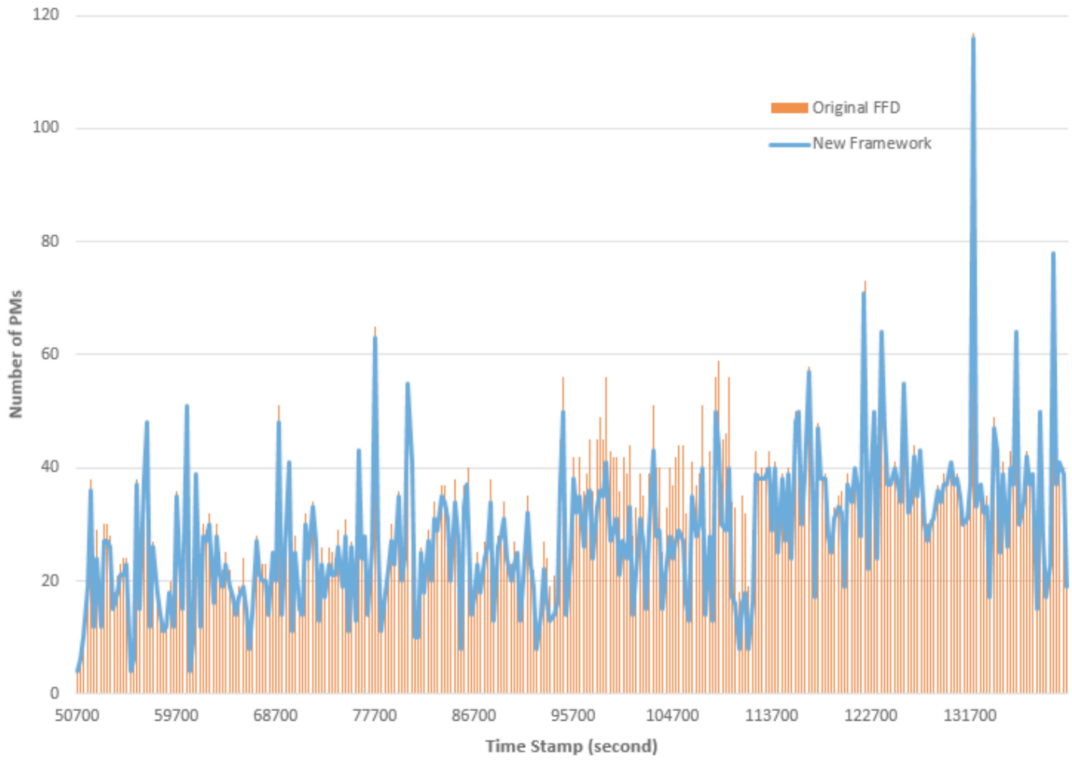

Figure 5-16 Medium-scale VM placement experimental result: timely PM quantity

Figure 5-17 shows PM quantity overlap of medium-scale data set. Similarly, the new framework saves a great number of PMs during the time span with huge amount of tiny tasks and saves nearly 20 PMs at most in one time slot. In the other time slots, the new framework can usually save 1 to 5 PMs. Compare to large-scale experiment, the percentage of PM quantity saving is a little lower.

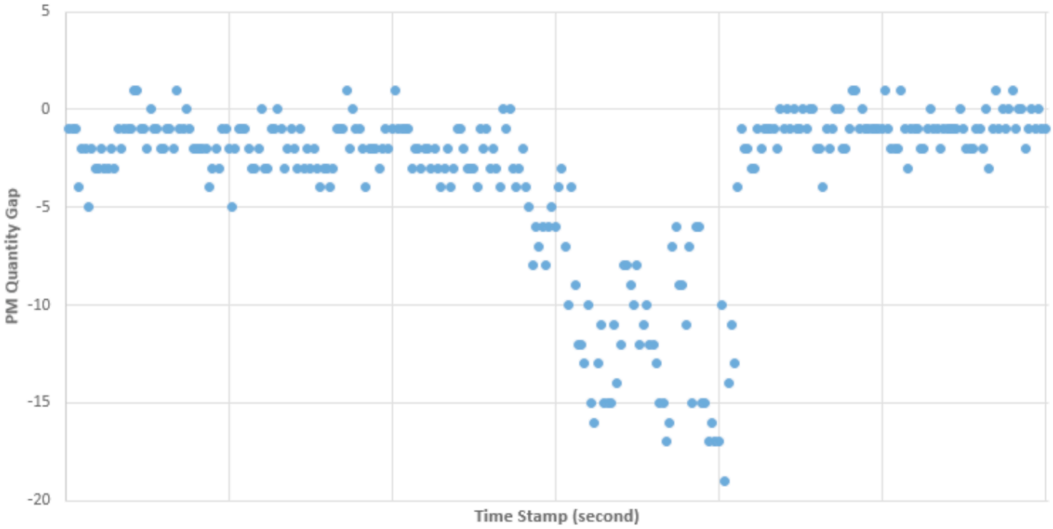

Figure 5-17 Medium-scale VM placement experiment result: PM quantity overlap

5.3.3 Small scale experiment

Figure 5-18 shows the number of PMs in each time slot of small-scale data set. Similarly, during the time span with huge amount of tiny tasks, there are more PMs in original FFD than the new framework and there is no remarkable difference between the PM quantity of these two methods in other time slots. The PM quantity of new framework range from 4 to 38 and most PM quantity ranges between 4 and 20.

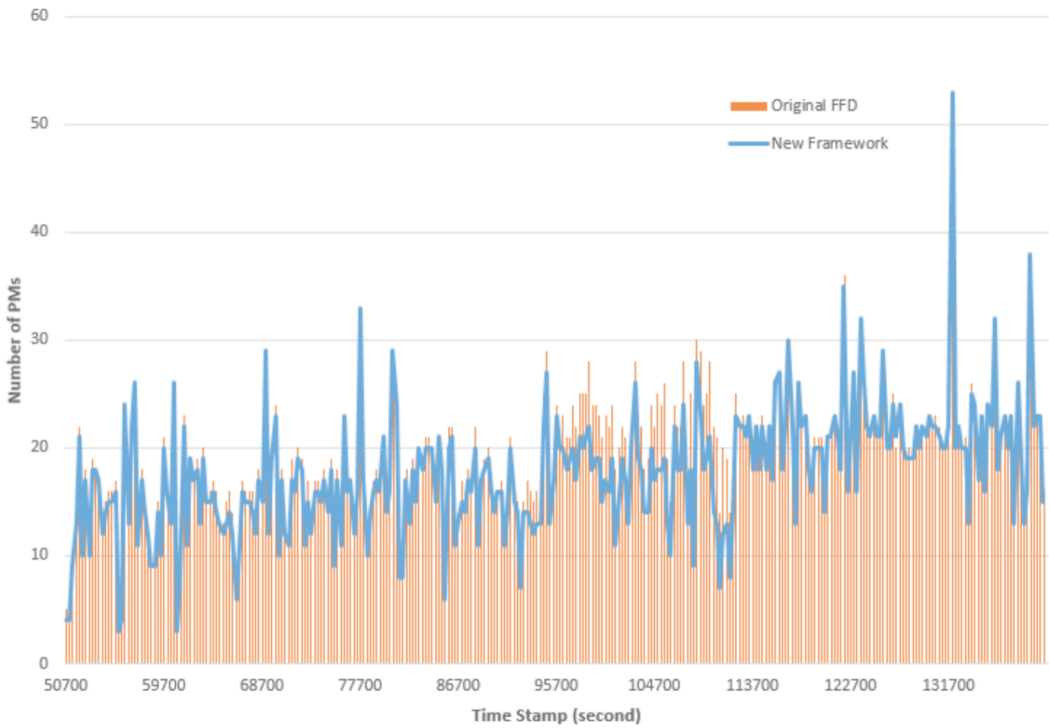

Figure 5-18 Small-scale VM placement experimental result: timely PM quantity

Figure 5-19 shows PM quantity overlap of small-scale data set. Similarly, the new framework saves a great number of PMs during the time span with huge amount of tiny tasks and saves 8 PMs at most in one time slot. In the other time slots, the framework can only save 1 or 2 PMs.

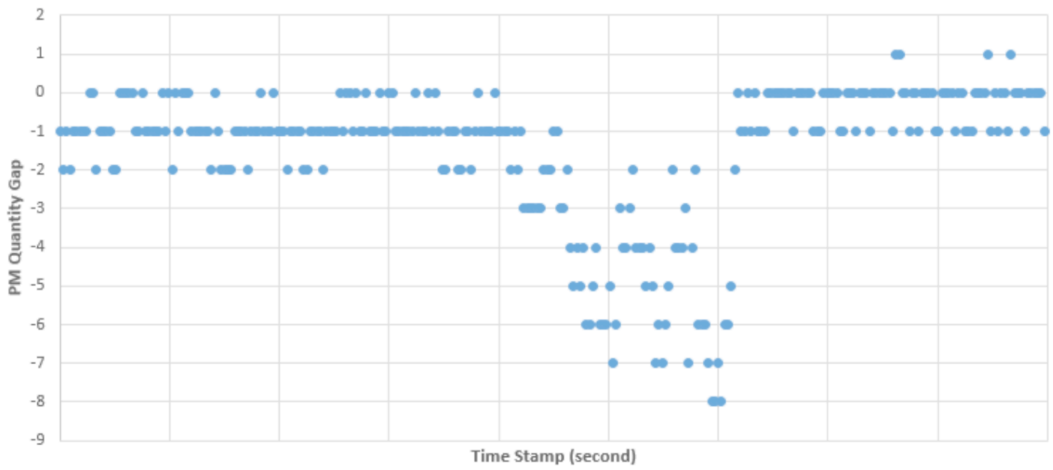

Figure 5-19 Small-scale VM placement experimental result: PM quantity overlap

5.4 Average PM utilization evaluation

5.4.1 Large-scale experiment

Figure 5-20 shows the average CPU utilization proportion of the new framework and original FFD of large-scale data set. The average CPU utilization indicates an average of all PMs utilization in each time slot. Figures of the new framework are in blue lines and figures of original FFD are in orange lines. In comparison, in most of the time slots, the new framework has a higher average CPU utilization than original FFD.

However, in some time slots, the new framework has a lot less average CPU utilization than original FFD has. That is because in those time slots, according to figure 5-15, the new framework applies more PMs than original FFD. If one extra PM contains very few VMs, the CPU utilization will be quite low that decrease the average CPU utilization. Though the lowest average CPU utilization of the new framework is below 0.9, the percentage of such situation is less than 1%. In

comparison, there is no such obvious the average CPU utilization difference of the two methods during the time span with huge amount of tiny tasks. It can be seen that though original FFD uses far more PMs than new framework, there are no PMs with especially low CPU utilization.

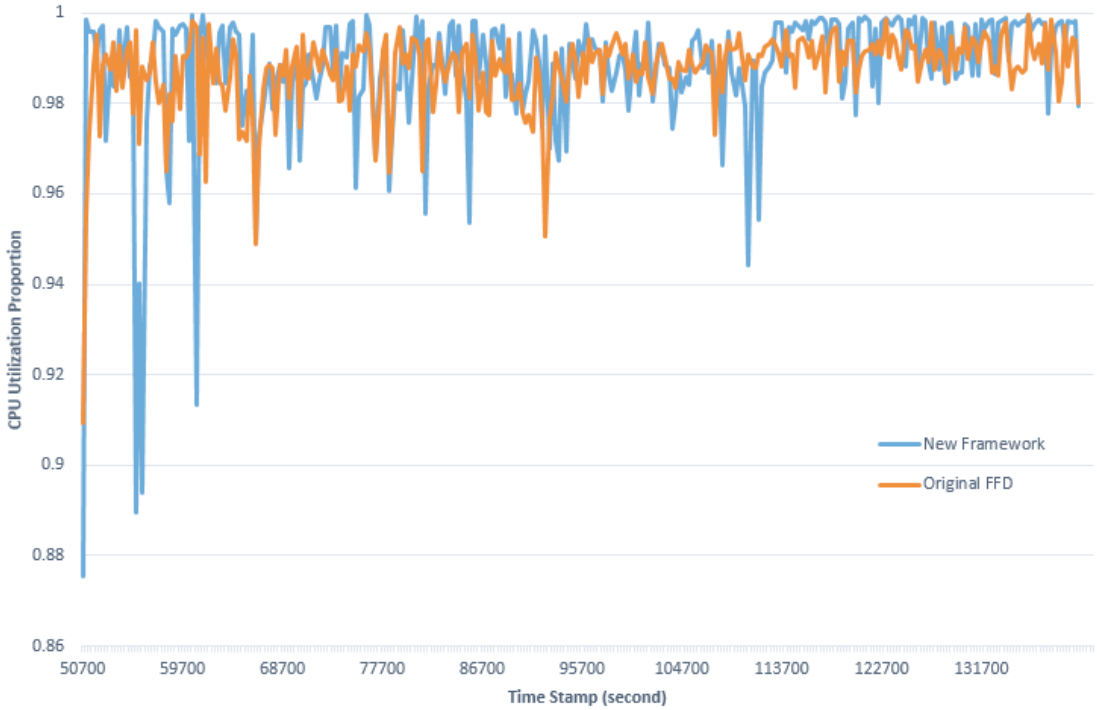

Figure 5-20 Large-scale VM placement experimental result: average PM utilization

5.4.2 Medium-scale experiment

Figure 5-21 shows the average CPU utilization proportion of the new framework and original FFD of medium-scale data set. The performance is in a similar trend as that of large-scale experiment. In the most situations, the new framework preforms well in CPU utilization and the difference between original FFD has increased significantly than that of large-scale experiment.

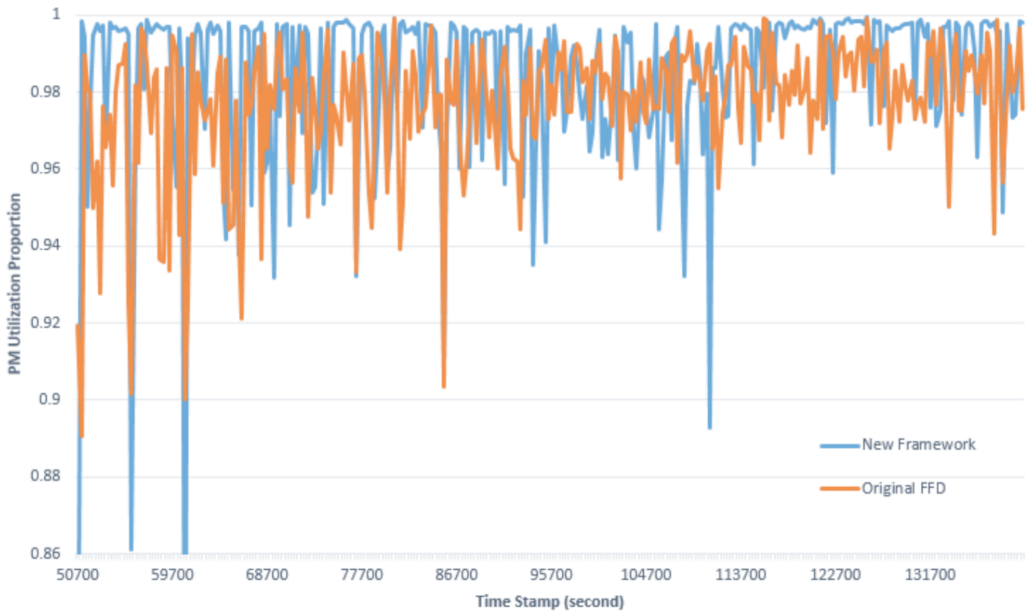

Figure 5-21 Medium-scale VM placement experimental Result: average PM utilization

5.4.3 Small-scale experiment

Figure 5-22 shows the average CPU utilization proportion of the new framework and original FFD of small-scale data set. The performance is in a similar trend as that of large-scale experiment and medium-scale experiment. The new framework preforms well in CPU utilization and the difference between original FFD has also increased significantly than that of large-scale experiment.

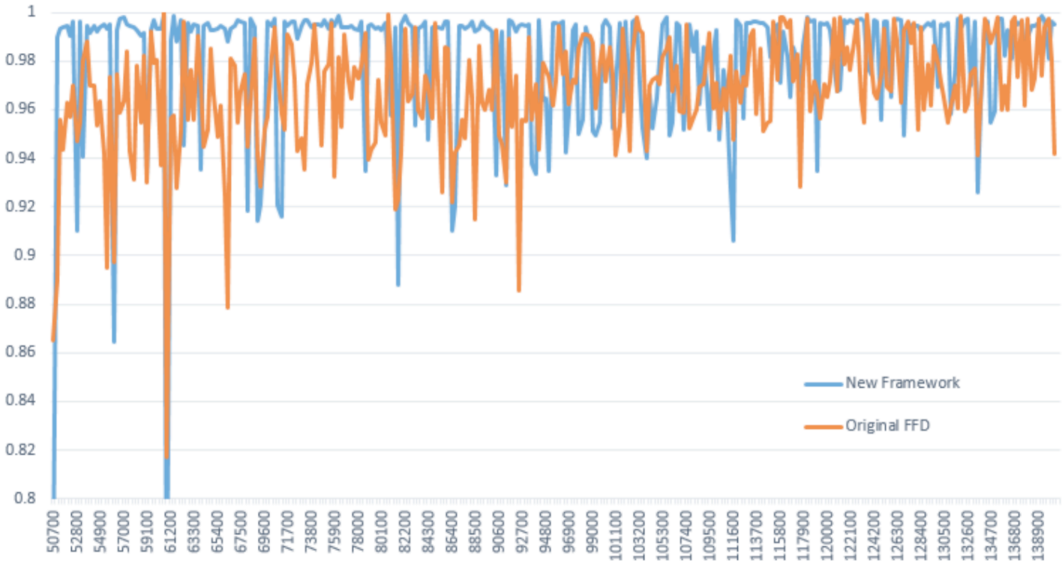

Figure 5-22 Small-scale VM placement experiment result: average PM utilization

5.5 Energy consumption evaluation

Table 5-1 displays the energy cost of the three data sets with new framework and original FFD. The energy cost is calculated with formula in section 4. Percentage gap shows how much percentage of energy has saved from new framework. According to the figures, in large-scale and medium-scale experiments, the new framework has saved 12% energy. In small-scale experiment, the new framework has saved 8%. It is a significant improvement in energy optimization.

	New framework	Original FFD	Percentage gap
Large-scale	1.391*10^9	1.585*10^9	12.240%
Medium-scale	6.011*10^8	6.833*10^8	12.030%
Small-scale	3.632*10^8	3.947*10^8	7.981%

Table 5-1 Entire energy cost of the experiments (kWh)

Figure 5-23 displays the relationship between execution time and the number of processed tasks. X-axis is tasks (records) quantity and y-axis is execution time. Execution time covers tasks and VM classification and VM placement. Results of the new framework and original FFD are represented in two lines. Each line is formed from the task quantity and execution time in three scaled data sets experiments. The relationship among the results of those experiments of original FFD itself is linear, while that of new framework is linear approximately. This chart indicates that the new framework has strong scalability.

In large-scale data set, the new framework processed 95531 pieces of data only in 16 seconds, which is quick enough. Furthermore, though in all of the three experiments, the new framework has slightly more execution time than original FFD and the execution time gap is predicted to be increased one second every 70,000 pieces of extra data, comparing to the great gaps of energy consumption, the execution time difference is acceptable.

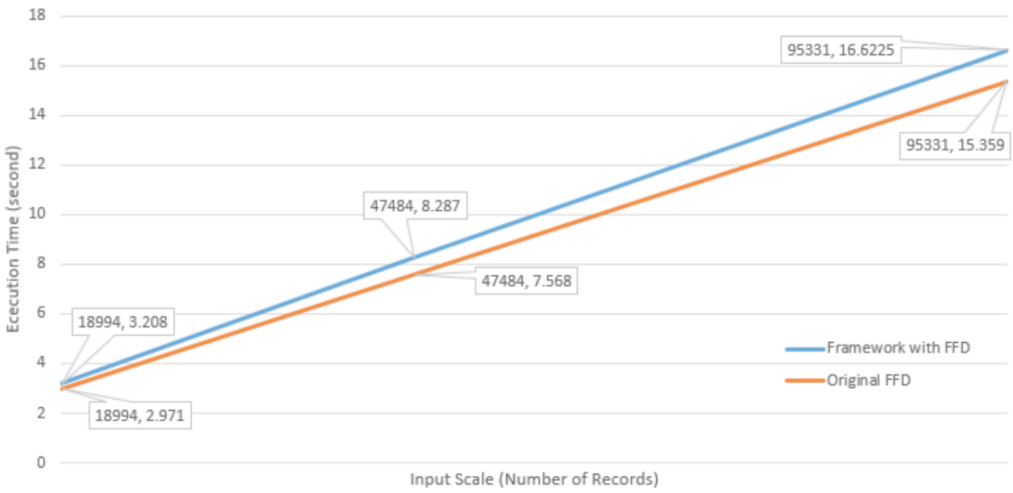

Figure 5-23 Relationship between execution time and task amount

5.6 Summary

This section explains the outcome with figures and algorithms in three phases. It can be seen that the new framework has an overwhelming advantage over original FFD in PM quantity saving, average CPU utilization and energy consumption, and an acceptable computation time.

6 Discussion

This section brings a detailed analysis on the advantages and limitations of the framework performance and provides recommendations for further work.

6.1 Advantages

Based on the analysis in the previous section, the new framework has advantages in PM quantity saving, average CPU utilization enhancement and energy saving over original FFD. There is a remarkable improvement especially in larger data set. This project has achieved energy optimization in all the predicted aspects.

Firstly, the quantity of saved PMs is quite considerable. Generally, one task is assigned to one VM. As the CPU capability of VMs cannot be as tiny as the CPU requests from tiny task, there will be definitely a big waste of CPU. Thus, task packaging is an effective approach to enhance VM utilization and save PM quantity, especially when tiny tasks quantity is extremely huge. Permanent VMs placing ahead of other VMs also contributes to PM quantity saving.

Secondly, the average CPU utilization enhancement is significant. There are quite a lot PMs have nearly full CPU utilization while original FFD does not. Because of original FFD uses more PMs, with the same amount of total CPU request, it is natural that the CPU utilization of original FFD is lower. Accordingly, with more PM quantity gaps, the difference of average CPU utilization is more obvious.

Thirdly, based on the saved PM quantity and average CPU utilization enhancement, the new framework produces an approximated 10-percentage energy optimization. If $10 per watt is cost for per kWh [17], as mentioned in Introduction section, the saved energy almost equals to 10 billion dollars, as well as tens of metric tons carbon pollution. Besides, The linear-like chart in figure 5-22 demonstrates that the framework is able to fit for a larger-scaled data set, which indicates a strong scalability of the model.

Finally, through comparing the trend of peak CPU usage between source data and profiled data, predicted PM quantity from total CPU request and actual PM

quantity, it has approved that the results of data profiling and VM placement plan are acceptable. The framework is feasible in data center energy optimization.

According to the results above, this framework has a strong feasibility and scalability, achieved energy optimization particularly well.

6.2 Limitations

Computation efficiency is the first limitation for the new framework. According to figure 5-12, the new framework does have advantages over original FFD in execution time. It could probably because of the new framework have an extra time costing of task packaging. Though there are more VMs in original FFD, the new framework did VM placement loop twice. As the essence of the two approaches are all FFD, it is hard to have a remarkable improvement in computation time unless the changing the algorithm.

Accuracy is the second limitation. FFD algorithm can only return the first fit object, but has limitations in searching for the most appropriate one. In other words, FFD cannot ensure there are as many PMs as possible with high utilization after each placement. Except that, there could be a small percentage of data error. Because the sample data is extracted every 200 lines or more, it cannot provide every piece of data is typical.

Limited data feature is another limitation. At the current stage, task attributes used for VM placement only contain CPU request and lifetime. In reality, far more attributes like RAM request and disk usage could affect the result of VM placement. Take Google cluster trace for an example, there are almost 30 columns (attributes) in task resource usage table and task event table and only 5 columns have been extracted for analysis. Too few feature variables may affect performance when the system being applied in a complex environment.

Besides, the samples used in experiment are partly extracted from the source data files. Some sample data may be unrepresentative due to large intervals. This could lead to a small proportion of errors in experiment results.

6.3 Recommendations

In further study, this system performance can be improved in data attributes analysis, accuracy and computation efficiency.

The first aspect is adding more data attributes into analysis. Many features could affect energy consumption, such as CPU usage and disk usage. The analysis is planned to start with such basic tasks attributes, then followed by jobs and PM attributes. Besides, tasks may have relationship between each other such as priority and dependence. Tasks should be placed ordered by priorities and tasks with dependence should be placed on the same PM. Except that, tasks could be repetitive, which means one task could appear more than one time. At the current stage, repetitive tasks have not been considered separately in the system. In the future, machines should be able to store the task features to avoid repetitive analysis when encounter the same task.

The second aspect is the accuracy of the result. As mentioned in limitations, FFD is not an effective algorithm in searching for the best result. Best-fit decreasing (BFD) algorithm and Generation algorithm (GA) will be used. With BFD, each VM can be set to the best-fit PM. GA can simulate the generation of data to help machines learning data features in order to improve the accuracy.

The third aspect is computation efficiency. On one hand, Java is not a high effective programming language in data manipulation. Since Java is a high-level language, source code needs to be explained to binary pseudo code on virtual machine (JVM) rather than converting to machine code directly. Java also requires abundant memory in file reading that could slow down processing speed. Thus, it is better to apply a low-level compiled language such as C language. On the other hand, computer performance also affects computation efficiency. A higher performance machine will be used for further experiments.

6.4 Summary

The new framework has achieved energy optimization in aspect of PM quantity saving, average CPU utilization and energy saving in the comparison with

original FFD. In further study, BFD and GA algorithms will be applied for enhancing the accuracy instead of FFD. More data features need to be analyzed to enhance the system feasibility with a low-level compiled language.

7 Conclusion

Data centers have huge energy consumption. It is important to increase PM utilization and PM quantity for data center energy saving. As there are few studies on data extraction and VM categories, it is necessary to have data profiling and VM classification for machines to understand records behavior.

This paper introduced a new profile-based framework for VM placement for energy optimization, consisting of three phases: profiling, tasks & VMs classification and VM placement. The basic algorithm for VM placement is improved FFD, the enhancements lie in task packaging, placement sequence and VM PM sorting. The framework is suit for small or medium enterprise with privacy data centers.

Three sampled data sets in different scales are used for performance evaluation. The data is extracted from Google cluster-usage traces logs. Data features of start time, end time, job ID, task ID and CPU request are used for building VM placement plan. The project produces a group of different sizes of VMs and PMs for task classification and VM placement. The measurement time of each placement is 300 seconds.

In aspect of PM quantity, normally, the framework can save less than 5 PMs in each time slot of large-scaled data set. But when there are huge amount of tiny tasks, the saved PM quantity achieves more than 30.

In aspect of average PM utilization, in majority of the time slots, the new framework achieved an average PM utilization up to 0.99%, which is a few percentage points higher than original FFD. There is a slightly increase of PM utilization difference in the timespan with huge amount of tiny tasks, but does not as significant as PM quantity difference.

In energy consumption, the framework saved around 10% power than original FFD, which is an significant improvement in energy consumption. It processed 10 thousands data in 16 seconds. Though the computation time is slightly slower that original FFD, it is still acceptable.

It can be seen that the new framework has an overwhelming advantage over original FFD in PM quantity saving, average CPU utilization and energy consumption, and an acceptable computation time. The framework has strong technical feasibility and scalability for large-scale data set. However, the data features for analyzing are few and the data extraction method cannot ensure all data is typical. The performance of framework may not have enough accuracy. In further study, more data features will be take into experiment with a low-level programming language to enhance computation efficiency. In a word, the new framework has achieved energy optimization particularly well in all of the previous three aspects.

Reference

[1] J, Shuja., K, Bilal., S, A, Madani., & S, U, Khan. (2014). *Data center energy efficient resource scheduling. 17(4).* 1265-1277. doi:10.1007/s10586-014-0365-0

[2] M, D, Hill. (2009). *The Datacenter as a Computer*. doi: 10.2200/S00193ED1V01Y200905CAC006

[3] M, Arregoces & M, Portolani. (2003). *Data Center Fundamentals.* Retrieved from https://books.google.com.au/books?id=DRIryrLoxKkC

[4] A, Shehabi., S, Smith., D, Sartor., R, Brown., & M, Herrlin. (2016). *United States Data Center Energy Usage Report.* Retrieved from http://eta.lbl.gov/sites/all/files/lbnl-1005775_v2.pdf

[5]P, Delforge. (2016). The Power of Efficiency to Cut Data Center Energy Waste. Retrieved from https://www.nrdc.org/experts/pierre-delforge/power-efficiency-cut-data-center-energy-waste

[6] P, Delforge. (2014). New Study: America's Data Centers Consuming -- and Wasting -- Growing Amounts of Energy. Retrieved from https://www.nrdc.org/experts/pierre-delforge/new-study-americas-data-centers-consuming-and-wasting-growing-amounts-energy

[7] S, Pelley, D, Mesiner, P, Zandevakili., & T, F, Wenisch. (2010). *Power Routing: Dynamic Power Provisioning in the Data Center.* ACM.

[8] Data Center Efficiency Assessment. (2014). Retrieved from https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IB.pdf

[9] M, Vasudevan., Y, Tian., M, Tang., & E, Kozan. (2014). *Profiling : an application assignment approach for green data centers. 67.* 94-108.

[10] G, Zhabelova., A, Yavarian., & V, Vyatkin. (2015). Data center energy efficiency and power quality: An alternative approach with solid state transformer. doi: 10.1109/IECON.2015.7392279

[11] D, Versick., I, WaBmann., & D, Tavangarian. (2013). *Power Consumption Estimation of CPU and Peripheral Components in Virtual Machines* .13(3), 17-25. doi: 10.1145/2537728.2537730

[12] D, Liu., X, Sui., & L, Li. (2016). *An Energy-efficient Virtual Machine Placement. Algorithm in Cloud Data Center*. 719-723. doi: 10.1109/FSKD.2016.7603263

[13] M, A, Islam., S, Ren., A, H, Mahmud and G, Quan. (2016). *Online Energy Budgeting for Cost Minimization in Virtualized Data Center. 9(3).* 421-431. doi: 10.1109/TSC.2015.2390231

[14] I, Rodero., J, Jaramillo., & A, Quiroz. (2010). *Energy-efficient application-aware online provisioning for virtualized clouds and data centers.* doi: 10.1109/GREENCOMP.2010.5598283

[15] C, Reiss., & J, Wilkes. (2011). *Google cluster-usage traces: format + schema*. Retrieved from https://googleclusterdata.googlecode.com/files/Google%20cluster-usage%20traces%20-%20format%20%2B%20schema%20(2011.10.27%20external).pdf

[16] J, E, Smith., & R, Nair. (2005). *Virtual Machines: Versatile Platforms for Systems and Processes*. Retrieved from https://books.google.com.au/books?id=JPhQw41vD2MC

[17] Hewlett-Packard Development Company. (2003). *HP virtualization*. Retrieved from ftp://ftp.hp.com/pub/enterprise/ent-old/Virtualization_5-5-03.pdf

[18] A, Maydanchik. (2007). *Data quantity assessment*. Retrieved from https://books.google.com.au/books?id=aK5yaxAnmIQC

[19] F, L, Pires., & B, Baran. (2015). Virtual Machine Placement Literature Review. Retrieved from https://arxiv.org/pdf/1506.01509.pdf

[20] M, Aiello., E,B, Johnsen., S, Dustdar., & I, Georgievski. (2016). Service-Oriented and Cloud Computing. Retrieved from https://books.google.com.au/books?id=Vr_mDAAAQBAJ

Appendix 1 Reflection Discussion

In the thirteen weeks of this semester, I have built a Java project on VM placement and the project could help for energy saving. This is the first time I have worked on big data and associates with real life. Different from small-scaled data processing, manipulating large-scaled data is really a big challenge. I have gained a lot through the project and felt a sense of accomplishment.

The first aspect I have gained is the importance of algorithm efficiency and coding standard. The first time I studied data structure was in my bachelor. I did not have a deep understanding of the difference of data type and the importance of algorithm efficiency. As the amount of data I used before in practice was too small, no matter what data type I chose and how many redundant loops I used, it was always quick enough to have the result. However, when I tried to extract data from the files of over 1 million lines, I was astonished to found there was not enough CPU on my laptop and the program was force to get stuck. In order to free memory, I learnt how many bytes that List, String, Float and Double take and changed each variable to the most appropriate data type. I also realized that any unnecessary functions, even an extra searching statement, could affect computation time that should be removed.

I also have an improvement on my coding standard from the weekly communication with my supervisor. I did not used to add comments for codes unless the structure of the function is complex enough. However, different people have different coding styles, it is hard for programmers to understand the code written by others without comments. Besides, as it takes a long time for data manipulation, I learnt to add a “progress bar”, that is, an output sentence to tell the programmer the current running method.

The second part I have gained is a deep concept of big data programming. Though my major is data science, I did not have a concept of big data until I did the project. Unlike web programming or mobile device programming, there are no existing APIs for big data programming. All the functions need to be designed from scratch and it is quite important to apply a good algorithm with high accuracy and efficiency. Except that, data analyzing is also a key segment. It is far

more than generating colourful charts and read the figures. The hidden information behind the figures is the most valuable and it is not easy to search for the useful information. A high-efficiency programming environment is also necessary. Those factors make big data programming a high valuable technique.

Though the project has accomplished successfully before the deadline and meets all the demands, the procedure was not smooth. The biggest challenge I faced is data manipulation. As I used laptop for programming, it is inevitably that it will take a long processing time. Hence each modification should be effective enough to avoid more time costing on running the program. I spent lots of time on algorithms to enhance efficiency, even focused on data type. Though it took a lot of time for programming, I have gained a deep understanding of algorithm efficiency.

Except that, at the beginning, I did not have a throughout understanding of the data features and just coding mechanically without a deep thinking of why I should code like this and what the result should be like till I handled the result. Through the discussion with supervisor, I realized that the result was completely different from expectations. I had to waste time on going through the code to find errors. This was the worst thing I did in the project and it was really a meaningful lesson for me.

At last, it was really a valuable programming experience in my final semester. In the future, I will regard this experience as a guide on my further study on big data programming.

Appendix 2 Weekly Report