國立交通大學

資訊科學與工程研究所

碩 士 論 文

基於雲端運算公有雲之資源利用率分析與預測框架

A Flexible Analysis and Prediction Framework on Resource Usage in Public Clouds

研 究 生： 林家瑜

指導教授： 曾煜棋 教授

王蒞君 教授

中 華 民 國 101 年 7 月

基於雲端運算公有雲之資源利用率分析與預測框架

A Flexible Analysis and Prediction Framework
on Resource Usage in Public Clouds

研 究 生： 林家瑜 Student：Chia-Yu Lin

指導教授： 曾煜棋 Advisor：Yu-Chee Tseng

王蒞君 Li-Chun Wang

國 立 交 通 大 學
資訊科學與工程研究所 研 究 所
碩 士 論 文

A Thesis
Submitted to Department of Computer Science
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science

July 2012
Hsinchu, Taiwan, Republic of China

中華民國 101 年 7 月

基於雲端運算公有雲之資源利用率分析與預測框架

學生：林家瑜　　　　　　　　　　　　　　　　　　　　　指導教授：曾煜棋 王蒞君 教授

國立交通大學資訊科學與工程學系(研究所)碩士班

摘　要

在雲端基礎設施服務(Infrasructure as A Service)裡，使用者可以向雲端服務的提供者承租虛擬機器來執行它們的程式，但是虛擬機器有多種的規格，使用者對於自己的程式應該花多少個虛擬機器來執行毫無概念。因此，我們提出了一個可以預估資源的服務框架，此服務框架能夠告訴使用者對於即將執行的程式需要最少需要多少個虛擬機器執行才能在特定的時間內完成，為了驗證此預估資源的服務框架可行性，我們還做了一些範例的研究，由於 Frequent Pattern Growth (FP-growth)、K-means、Particle Swarm Optimization (PSO)的輸入資料皆相當大量，運算複雜，因此我們將此三種演算法修改成平行化的執行方式，並運行在此框架底下進行資源預估，實驗結果顯示我們此框架可以成功地針對某一個新來的工作預估出在特定時間內完成所需要的虛擬機器個數。另外，從範例演算法中得知此框架不僅可運行輸入資料量小的工作，亦適用於輸入資料很大量的工作，此框架相當彈性，此彈性化以及實用的框架讓使用者在租用虛擬機器時不會租太多虛擬機器而浪費錢，也讓雲端服務的提供者能夠更方便地管理系統內的虛擬機器。

關鍵字： 資源預估框架、虛擬機器、平行化FP-growth、平行化K-means

A Flexible Analysis and Prediction Framework on Resource Usage in Public Clouds

Student: Chia-Yu Lin

Advisor: Prof. Yu-Chee Tseng
Prof. Li-Chun Wang

Department of Computer Science, National Chiao Tung University

ABSTRACT

In cloud computing environments, users can rent virtual machines (VMs) from cloud providers to execute their programs or provide network services. While using this kind of cloud service, one of the biggest problems for the users is that how many VMs are needed to complete the jobs without spending too much money and time. In this paper, we propose a resource prediction framework (RPF) which can help users rent the minimum number of virtual machines and complete their jobs within a user specified time constraint. In order to verify the feasibility of RPF, we have done 3 case studies, parallel frequent pattern growth (FP-Growth), parallel K-means and Particle Swarm Optimization (PSO), on the proposed framework. FP-growth, K-means and PSO are data intensive algorithms. These algorithms may be executed repeatedly with different execution parameters to find the optimal results. When evaluating RPF by these algorithms in cloud environments, we have to modify them to parallel versions. The evaluation results indicate that RPF can successfully obtain the minimum number of VMs with acceptable errors. According to the results of case studies, the proposed RPF can be adopted by data intensive jobs, which is flexible and useful for users and cloud system providers.

Keywords: resource prediction framework, virtual machines, parallel FP-growth, parallel K-means

致謝

首先特別要感謝的是我的指導老師曾煜棋教授以及王蒞君教授。每兩週與老師討論碩論內容，曾老師常常能在短時間內就點出我的一些盲點，即時修正，讓我能順利完成碩士口試畢業;王老師則是時常反問我碩士論文的貢獻以及創新點，讓我能在碩士論文的撰寫中更能強調作品的貢獻以及價值。另外，要感謝各位 HSCC 實驗室的學長姊們常能與我們討論給予一些經驗，讓我們在修課以及研究上更加順利，尤其要感謝陳彥安學長的指導，學長在每週的討論總會給予適當的意見及幫助，在論文的撰寫方面也給予許多重要的提點以及方向修正，口試完後還幫我修改投稿的論文，非常謝謝學長。除了實驗室的學長姐外，還要感謝實驗室的好夥伴、好同學在這兩年間各方面的幫忙，有你們的幫忙讓我的碩士生涯更加充實與豐富。最後，要感謝我的家人在我就讀交大的路途上一路支持著我，謝謝你們這兩年間的包容與體諒，讓我得以順利完成碩士學位。

Contents

List of Figures

Chapter 1

Introduction

Cloud computing provides the users with a large number of computing and storage resources. By renting the resource from cloud provider, the users could avoid the hardware investment and maintenance cost. The services of cloud computing is generally classified into 3 service types, software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). Amazon Elastic Compute Cloud (Amazon EC2), which is an IaaS provider, provides many types of VMs with different EC2 computing units and memory sizes. The billing method is based on chosen VM capabilities and the rented time period. MapReduce is a parallel programming platform for users to develop parallel programs in cloud environments. While users analyze the massive data in a cloud database, the parallel execution can reduce the execution time intensively. However, users usually do not know how many VMs they really need to execute a job. There is a trade-off between the rented VM numbers and the execution time of a job. Therefore, we propose a resource prediction framework (RPF) which predicts the minimum number of VMs for a MapReduce job under a user specified time constraint.

The proposed RPF consists of training and prediction algorithms. In training algorithm, users execute their parallel jobs with different execution parameters and the system records the execution parameters, execution time and allocated VM numbers. After collecting enough information, the training algorithm analyzes the collected information and computes a model for prediction. When the job is executed with a new execution parameters and an expected response time, the prediction algorithm computes the minimum number of VMs, which is based on the trained model, while considering the expected response time.

In this paper, we introduce 3 case studies, parallel frequent pattern growth (parallel FP-growth), parallel K-means and Particle Swarm Optimization (PSO), to verify the performance of RPF. Since these algorithms are data intensive, the execution times can be reduced intensively while executing in parallel. We modify them to parallel versions by referring to [5][15][7] and execute them on RPF. The evaluation results show that RPF can predict the minimum number of VMs with small root mean square errors (RMSE).

The contribution of this paper are as follows:

- A novel job-oriented resource prediction framework.
- The proposed training and prediction algorithms.
- The regression functions in prediction algorithms.

The rest of the paper is organized as follows: Chapter 2 details the related work of resource provisioning in cloud datacenters. Chapter 3 describes the system design. The algorithms of RPF includes training and prediction algorithm are discussed in Chapter 4. Chapter 5 describes the case which is demonstrated on RPF and the performance evaluation. Finally, Chapter 6 concludes the paper.

Chapter 2

Related Work

The recent research of cloud computing has been focused on SLA-based resource provisioning. [3] proposed a model to manage the VMs in the datacenter dynamically to fit SLA. The model monitors the resource demand during the current time window in order to make decisions about the server allocations and job admissions during the next time window. Therefore, the model can adjust the number of execution VMs of high performance computing (HPC) jobs and web jobs dynamically to conform with SLA. In [14] and [11], they meet SLA by resource prediction. The prediction results are the utilizations of CPU and memory. Users cannot easily realize the needed VM numbers from this estimation result. The framework we proposed output the minimum number of VMs, which is more intuitive for users. [13] proposed an optimal resource provisioning for MapReduce programs. They analyze the execution time of mapper and reducer process and use regression methods to estimate the execution time of a MapReduce job with different numbers of VM. The analyzing process they proposed is only suitable for simple MapReduce job such as WordCount and PageRank. Therefore, we propose a framework which can estimate the number of VM and execution time for any kind of MapReduce job.

Chapter 3

SYSTEM DESIGN

In this section, we introduce the architecture of RPF which can predict the minimum number of VM for a MapReduce job within the expected execution time. We assume that the MapReduce platform of training process and prediction process is the same.

The input of this framework is a set of execution parameters of the new MapReduce job, $P = \{p_1, p_2, \ldots, p_{n-1}\}$ and users' expected execution time, EPF. The RPF output the minimum number of VM which can complete the new MapReduce job within the users' expected time. The system architecture of this framework is shown in Fig. 3.1.

In service level agreement (SLA) module, users have to input the new input file, the execution parameters and the response time they expected. This information is sent to prediction model.

In training model, the system collects the historical resource usage data including the number of VM, execution parameters and execution time of old input files. In addition, regression-based model are adopted to train the historical usage data. The training model will output the model which has smallest RMSE to prediction model.

The historical resource usage data, the execution parameters of new jobs, the expected response time and the model which is output by training model are the input of prediction model. In prediction model, regression-based methods are adopted to predict the number of VM and execution time.

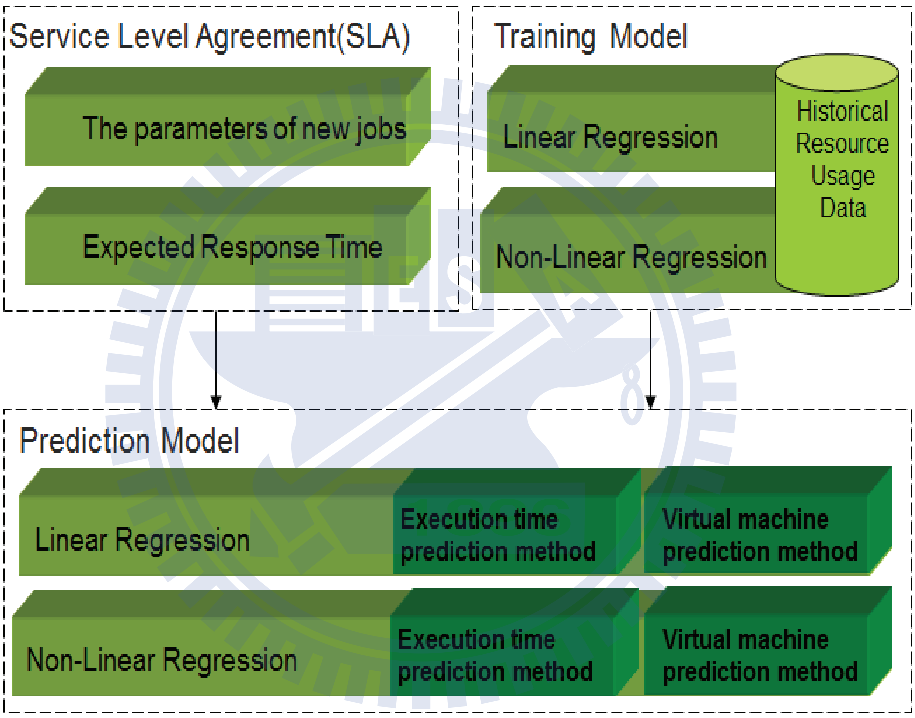


Figure 3.1: System architecture.

Chapter 4

RPF Algorithms

We propose two algorithms to predict the minimum number of VM of a MapReduce job. The details of two algorithms are shown in Section 4.1 and Section 4.2.

Algorithm 1 Training Algorithm

Input: $S : \{(s_{11}, \ldots, s_{1n}, z_1), \ldots, (s_{k1}, \ldots, s_{kn}, z_k)\}$,a set of execution parameters and execution time.

Output: $A : \{a_1, \ldots, a_m\}$,coefficient set of regression equation.

if *LinearModel* **then**

Fill the set S in $f(x_1, \ldots, x_n, A) = a_1 x_1^2 + \ldots + a_n x_n^2 + a_{n+1} x_1 + \ldots + a_{2n} x_n + a_m$

Find out A to minimize $\sum_{i=1}^{k}[z_i - f(s_{i1}, \ldots, s_{in}, A)]^2$.

$$\begin{cases} f(s_{11}, \ldots, s_{1n}, A) = a_1 s_{11}^2 + \ldots + a_n s_{1n}^2 + a_{n+1} s_{11} + \ldots + a_{2n} s_{1n} + a_m \\ f(s_{21}, \ldots, s_{2n}, A) = a_1 s_{21}^2 + \ldots + a_n s_{2n}^2 + a_{n+1} s_{21} + \ldots + a_{2n} s_{2n} + a_m \\ \ldots \\ f(s_{k1}, \ldots, s_{kn}, A) = a_1 s_{k1}^2 + \ldots + a_n s_{kn}^2 + a_{n+1} s_{k1} + \ldots + a_{2n} s_{kn} + a_m \end{cases}$$

else

if *Non − linearModel* **then**

Fill the set S in $f(x_1, \ldots, x_n) = a_1 e^{a_2 x_1} + \ldots + a_{2n-1} e^{a_{2n} x_n} + a_m$

Find out A to minimize $\sum_{i=1}^{k}[z_i - f(s_{i1}, \ldots, s_{in}, A)]^2$.

$$\begin{cases} z_1 = f(s_{11}, \ldots, s_{1n}) = a_1 e^{a_2 s_{11}} + \ldots + a_{2n-1} e^{a_{2n} s_{1n}} + a_m \\ z_2 = f(s_{21}, \ldots, s_{2n}) = a_1 e^{a_2 s_{21}} + \ldots + a_{2n-1} e^{a_{2n} s_{2n}} + a_m \\ \ldots \\ z_k = f(s_{k1}, \ldots, s_{kn}) = a_1 e^{a_2 s_{k1}} + \ldots + a_{2n-1} e^{a_{2n} s_{kn}} + a_m \end{cases}$$

end if

end if

4.1 Training Algorithm

Users have to execute the jobs whose input file is time-related or same with the new MapReduce job's input file before executing RPF prediction process. The input of training algorithm are a set of execution parameters and execution time of these jobs, $S : \{(s_{11}, \ldots, s_{1n}, z_1), \ldots, (s_{k1}, \ldots, s_{kn}, z_k)\}$. Because we use regression-based methods to predict the minimum number of VM, the output of training algorithm is the coefficient set of regression equation, $A : \{a_1, \ldots, a_m\}$. There are linear and non-linear regression model. In linear model, the set S is filled in $f(x_1, \ldots, x_n) = a_1x_1^2 + \ldots + a_nx_n^2 + a_{n+1}x_1 + \ldots + a_{2n}x_n + a_m$ to find out the set A. Otherwise, the set S is filled in $f(x_1, \ldots, x_n) = a_1e^{a_2x_1} + \ldots + a_{2n-1}e^{a_{2n}x_n} + a_m$ to find out the set A in non-linear model.

4.2 Prediction Algorithm

Algorithm 2 Prediction Algorithm 1

Input: $P : \{p_1, \ldots, p_{n-1}\}$,a set of execution parameters, Regression Model, $A : \{a_1, \ldots, a_m\}$, the set of coefficient of regression model, EPT:expected execution time
Output: N:number of execution VMs
for $i := 1$ to r **do**
  if $RegressionModel = LinearModel$ **then**
    Fill the set A,set P and EPT in
    $f(x_1, \ldots, x_n) = a_1x_1^2 + \ldots + a_nx_n^2 + a_{n+1}x_1 + \ldots + a_{2n}x_n + a_m$
    Use the following function to predict the number of execution VMs N.
    $N = f(p_1, \ldots, p_{n-1}, EPT) = a_1s_1^2 + \ldots + a_{n-1}s_{n-1}^2 + a_nv_i^2 + a_{n+1}p_1 + \ldots + a_{2n-1}p_n + a_{2n}vi + a_m$
  else
    if $Non-linearModel$ **then**
      Fill the set A ,set P and EPT in
      $f(x_1, \ldots, x_n) = a_1e^{a_2x_1} + \ldots + a_{2n-1}e^{a_{2n}x_n} + a_m$
      Use the following function to predict the number of execution VMs N.
      $N = f(p_1, \ldots, p_{n-1}, EPT) = a_1e^{a_2p_1} + \ldots + a_{2n-2}e^{a_{2n-1}p_{n-1}} + a_{2n}e^{a_{2n}v_i} + a_m$
    end if
  end if
end for

After training procedure, the system predicts the minimum number of VM of the new job by prediction algorithm. The input are a set of execution parameters, $P : \{p_1, \ldots, p_{n-1}\}$, regression model, a set of coefficient of regression model which is the output of training algorithm, $A : \{a_1, \ldots, a_m\}$ and the expected execution time, EPT.

The output is the minimum number of VM, N. The set of new execution parameters and the expected execution time are set to be the input of the regression model. After the regression process, the model outputs the minimum number of VM. The pseudo code of prediction algorithm is shown in Algorithm 2.

Algorithm 3 Prediction Algorithm 2

Input: $P : \{p_1, \ldots, p_{n-1}\}$,a set of execution parameters, Regression Model, $A : \{a_1, \ldots, a_m\}$, the set of coefficient of regression model,$V : \{v'_1, \ldots, v'_r\}$,a set of testing number of VM, EPT:expected execution time
Output: N:number of execution VMs
Let $T : \{t_1, \ldots, t_r\}$ be the set of predicted execution time responding to set V.
Let PRT: be the time in set T which is smaller than EPT and is the largest one in T.
for $i := 1$ to r **do**
  if $RegressionModel = LinearModel$ **then**
    Fill the set A,set P and set in
    $f(x_1, \ldots, x_n) = a_1x_1^2 + \ldots + a_nx_n^2 + a_{n+1}x_1 + \ldots + a_{2n}x_n + a_m$
    Use the following function to predict the execution time t_i.
    $t_i = f(p_1, \ldots, p_{n-1}, v_i) = a_1s_1^2 + \ldots + a_{n-1}s_{n-1}^2 + a_nv_i^2 + a_{n+1}p_1 + \ldots + a_{2n-1}p_n + a_{2n}vi + a_m$
  else
    if $Non-linearModel$ **then**
      Fill the set A and set X in $f(x_1, \ldots, x_n) = a_1e^{a_2x_1} + \ldots + a_{2n-1}e^{a_{2n}x_n} + a_m$
      Use the following function to predict the execution time t_i.
      $t_i = f(p_1, \ldots, p_{n-1}, v_i) = a_1e^{a_2p_1} + \ldots + a_{2n-2}e^{a_{2n-1}p_{n-1}} + a_{2n}e^{a_{2n}v_i} + a_m$
    end if
  end if
  Find the time PRT in set T which is smaller than EPT and is the largest one in T.
  After finding PRT, we can know the index i and get the corresponding VM number.
end for

In some programs, the parallelization is not achieved very well. It causes that the number of VM is only related with the set of execution parameters or expected execution time. Therefore, we cannot predict the number of VM directly. We propose another methods to solve this problem. There are two steps in this proposed prediction algorithm. First of all, we use regression model to predict the execution time of the new job executed by different number of VM and make up the set $T : \{t_1, \ldots, t_r\}$. The second step is comparing every execution time we predict with the EPT and find out the time which is called PRT. PRT is smaller than EPT and is the largest one in T. After finding PRT, we can output the VM number which is corresponding to PRT. The VM number, named

N is the minimum number which can complete the new job with $P : \{p_1, \ldots, p_{n-1}\}$ within the EPT. The pseudo code of prediction algorithm is shown in Algorithm 3.

The input are a set of execution parameters, $P : \{p_1, \ldots, p_{n-1}\}$, regression model, a set of coefficient of regression model which is the output of training algorithm,$A : \{a_1, \ldots, a_m\}$, the expected execution time, EPT and a set of testing number of VMs,$V : \{v'_1, \ldots, v'_r\}$. The output is the minimum number of VMs,N. There are two steps in proposed prediction algorithm. First of all, we use regression model to predict the execution time of the new job executed by different number of VMs and make up the set $T : \{t_1, \ldots, t_r\}$. The second step is comparing every execution time we predict with the EPT and find out the time which is called PRT. PRT is smaller than EPT and is the largest one in T. After finding PRT, we can output the VM number which is corresponding to PRT. The VM number, named N is the minimum number which can complete the new job with $P : \{p_1, \ldots, p_{n-1}\}$ within the EPT.

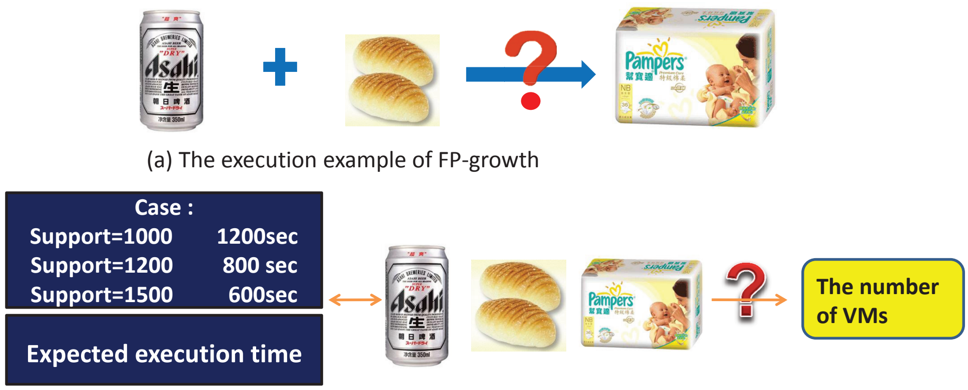
Chapter 5

Case Study

The data intensive jobs such as data mining jobs usually execute repeatedly by the same input files with different arguments. If users execute jobs in parallel, the execution time can be reduced intensively. While users rent the VMs to execute jobs in parallel, RPF enables users to use the least resource to execute parallel jobs. We demonstrate parallel frequent pattern growth (FP-growth) and parallel K-means on RPF to evaluate the performance of RPF. FP-growth is the most popular algorithm to analyze association rules from a lot of data. We modify FP-growth algorithm to parallel FP-growth algorithm which can execute on Hadoop platforms and collect the historical job usage data to estimate the fewest VM number of a new FP-growth jobs. The introduction of FP-growth is in Section 5.1.

5.1 FP-growth

A retailer such as Walmart may analyze the purchase relationship of products for product arrangement. They can execute FP-growth algorithm with different support which is the appearance frequency of products to find out the association rules of products. Therefore, the resource prediction of executing FP-growth with different support is really important. In addition, the input data of FP-growth such as the sales information of Walmart is huge. Executing FP-growth in parallel can reduce execution time massively. Therefore, we use FP-growth to be the example of RPF. Fig. 5.1 (a) shows the example of FP-growth. FP-growth can find out that whether they buy diapers if customers have already bought beers and bread. Fig. 5.1 (b) shows the execution process of FP-growth on RPF. While we are executing parallel FP-growth to find out the association rules, RPF records



(a) The execution example of FP-growth

(b) The execution example of FP-growth on RPF

Figure 5.1: The execution scenario of FP-growth.

min_support = 3

Customer ID	Items bought	Ordered
1	{a, c, d, f, g, i, m, p}	{f, c, a, m, p}
2	{a, b, c, f, i, m, o}	{f, c, a, m}
3	{b, f, h, j, o}	{f, b}
4	{b, c, k, s, p}	{c, b, p}
5	{a, c, e, f, l, m, n, p}	{f, c, a, m, p}

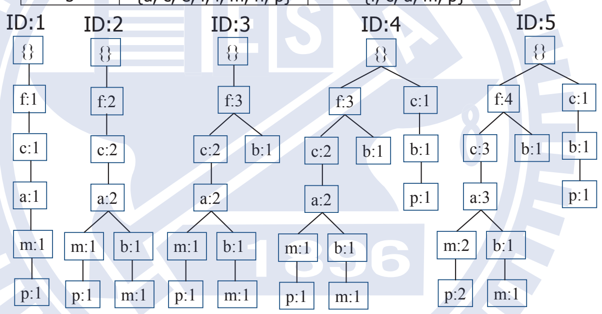


Figure 5.2: Constructing FP-trees.

the execution time and execution status such as the number of VMs and support. This historical resource usage data can be used to predict the execution number of VMs.

5.1.1 Parallel FP-growth

FP-growth uses the divide and conquer strategy to find the frequent items. There are two phases in FP-Growth algorithm. First phase is constructing FP-tree. In this phase, we find frequent items whose appearance frequency is larger than minimum support and sort items in frequency descending order, Fig. 5.2. And then we construct a root node which is marked by null and add the branch of each traction. The second phase is FP-Growth. In this phase, we find out conditional pattern bases which are associated with every frequent item in FP-tree. The conditional bases whose frequency are larger than minimum support

• (a) The conditional tree of 'a' and 'm'.

Item	Cond. Pattern base	Cond. FP-tree	Frequent patterns
c	f:3	f:3	fc:3
a	fc:3	fc:3	fca:3,fa:3,ca:3
b	fca:1,f:1,c:1		
m	fca:2,fcab:1	fca:3	fm:3,cm:3,am:3, fcm:3,fam:3,cam:3, fcam:3
p	fcam:2,cb:1	c:3	cp:3

• (b) The result of FP-growth.

Figure 5.3: Conditional tree and result of FP-growth.

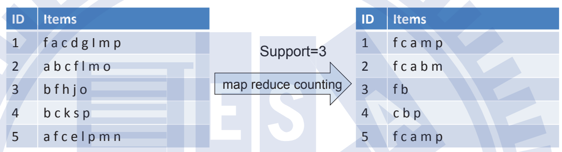


Figure 5.4: MapReduce counting in Parallel FP-growth.

are put in conditional FP-trees. After constructing conditional FP-tree of every frequent item, we can output the frequent patterns, Fig. 5.3.

In parallel FP-growth, we execute MapReduce counting to find frequent items whose appearance frequency is larger than minimum support and sort items in frequency descending orderFig. 5.4. After MapReduce counting, frequent items are separated into many parts for every mapper to execute. Every mapper output the conditional patterns according to input files and send the result to the reducers. The reducers have two step to execute. First, the reducers collect the conditional patterns from every mapper and combine these patterns which are associated with same frequent item. Second, the reducers construct the conditional FP-tree of every frequent item and output the frequent patterns Fig. 5.5. The execution time of parallel FP-growth is much shorter than the execution time of FP-growth.

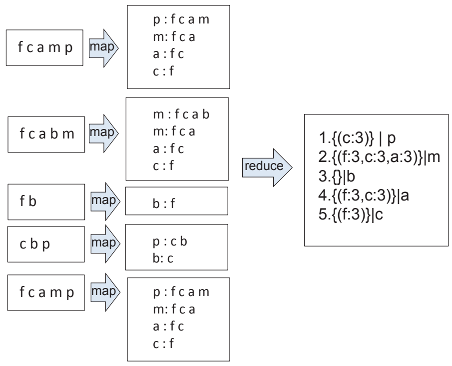


Figure 5.5: The execution process of parallel FP-growth.

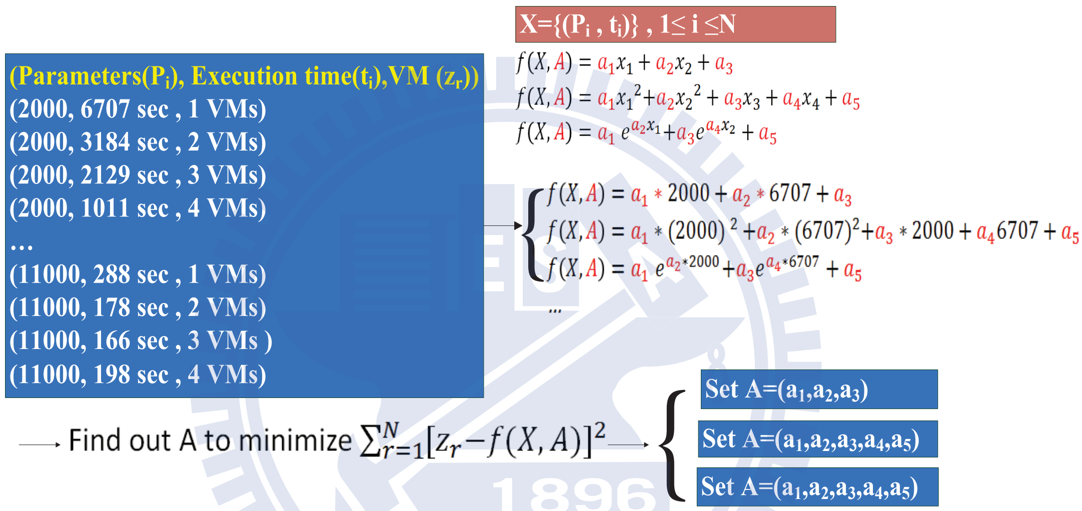


Figure 5.6: The example of training algorithm by FP-growth.

5.1.2 Parallel FP-growth on RPF

We execute parallel FP-growth on map-reduce platform with different support. Therefore, we can collect different execution parameters, such as support and execution time to be the input of training algorithm. In Fig. 5.6, we input support 2000 to 11000 and the historical execution time in the training algorithm. After executing training algorithm, we can obtain an regression model whose root mean square error (RMSE) is the smallest. We use this regression model to predict the number of execution VMs for the new jobs. The new job's support which is 5000 and expected response time 750 second which is set by users is the input of prediction algorithm. We put the support, the expected response time and the regression coefficient set A which is the output of training algorithm into the

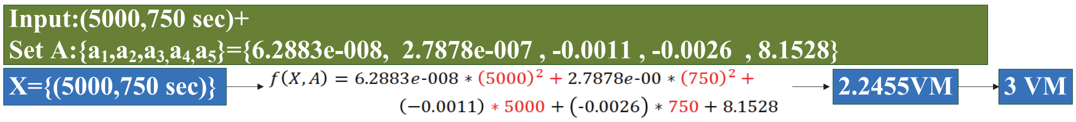


Figure 5.7: The example of prediction algorithm 1 by FP-growth.

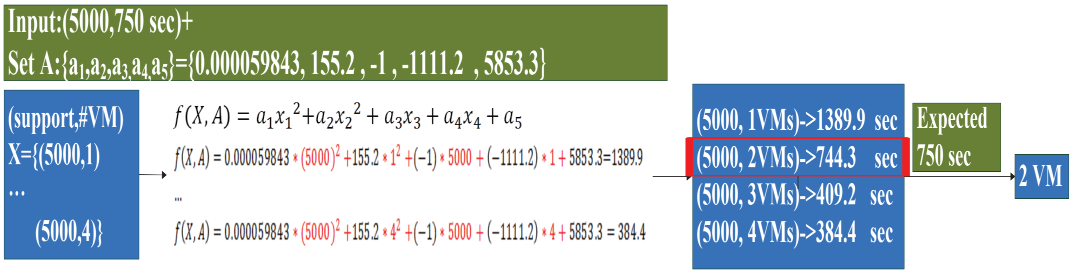
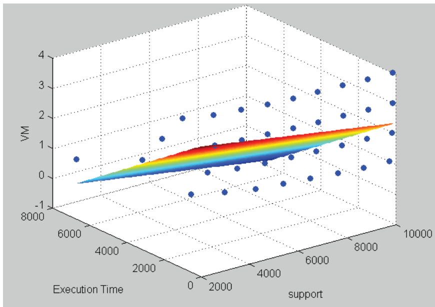


Figure 5.8: The example of prediction algorithm 2 by FP-growth.

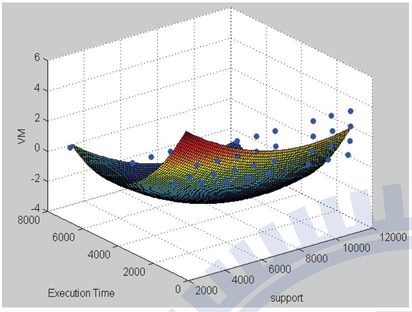
prediction equation. The equation outputs the fewest number of VMs. The example of prediction algorithm is shown in Fig. 5.7. If the prediction result is not accurate enough, we can try to use prediction algorithm 3 to estimate the number of execution VMs. We put the support, the number of VMs and the regression coefficient set A which is the output of training algorithm into the prediction equation. The equation outputs the execution time of every situation. We can use the expected execution time and the result of prediction algorithm to decide the fewest number of VMs of the new job. Fig. 5.8 shows the example of prediction execution time methods.

5.1.3 Parallel FP-growth on RPF Experiment Setup

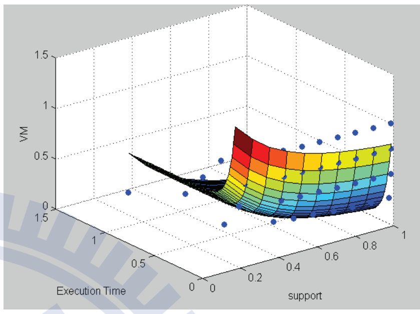
We construct two physical machines as the platform. The CPU of physical machines is Intel(R) Core(TM) i7-2600 CPU 3.40GHz and 4G ram. There are four virtual machines in each physical machine. There are 1 core and 1G memory in each virtual machine. Hadoop is installed in virtual machines. We refer to [5] and Apache Mahout project [10] to modify the Fp-growth in parallel. The dataset is from [8]. We choose the dataset which consists of 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.



(a) The curve of linear regression.



(b) The curve of quadratic regression.



(c) The curve of exponential regression.

Figure 5.9: The prediction curve of FP-growth by prediction algorithm 1.

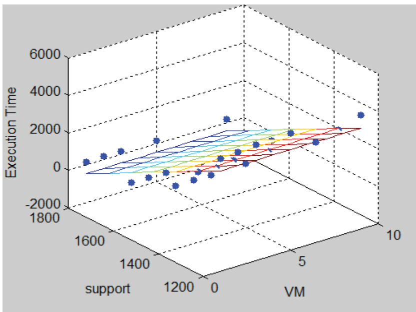
5.1.4 Evaluation

In prediction algorithm 1, we use linear and non-linear regression to predict the number of VM. The linear and non-linear prediction curves are shown in Fig. 5.9.

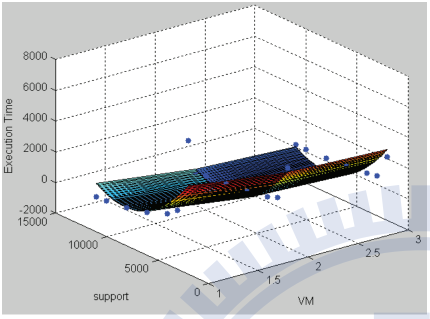
In prediction algorithm 2, we use linear and non-linear regression to predict the execution time. The linear and non-linear prediction curves are shown in Fig. 5.10. Fig. 5.11 shows the RMSE of prediction algorithm 1 and prediction algorithm 2. We can see that the prediction result of exponential function is the most accurate and the result of quadratic function has the maximum error. Overall, the regression methods we propose can output the fewest number of execution VMs to users accurately.

5.2 K-means

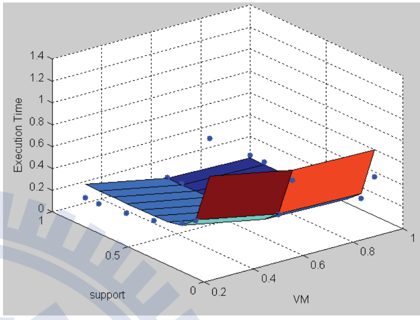
K-means is the simplest cluster algorithm to find clusters from a lot of data. We modify K-means algorithm to parallel K-means, execute it on Hadoop platforms to collect the historical job usage data and estimate the fewest VM number of a new K-means jobs. The scenario and introduction of K-means is in Section 5.2.1.



(a) The curve of linear regression.



(b) The curve of quadratic regression.



(c) The curve of exponential regression.

Figure 5.10: The prediction curve of FP-growth by prediction algorithm 2.

5.2.1 Parallel K-means

K-means clustering algorithm is a well known unsupervised clustering algorithm and it partitions objects into groups by analyzing the relationship or similarity of objects. Usually, the input data of K-means called dataset is intensive. For example, the BigCross dataset [1] which is 11,620,300 points in 57-dimensional space and the Census1990 dataset [2] which is 2,458,285 points in 68 dimensions are used in [12]. Furthermore, analyzing the relationship between objects could be processed in parallel. The sequential K-means algorithm could be modified to a parallel K-means algorithm. Then, we use the parallel K-means to be a case study of RPF.

K-means is used to partition the data into K clusters. The input of K-means are the cluster number K and the data which is used to cluster. The output is K clusters. There are four steps of K-means. First step is choosing K data to be the central node of every cluster. Every node compute the distance between it and the central node is second step. The third step is assign every node to the cluster according to the distance result in second step. The fourth step is recompute the central node of every cluster according to the average value of the data in the clusters. The second, third and fourth steps are

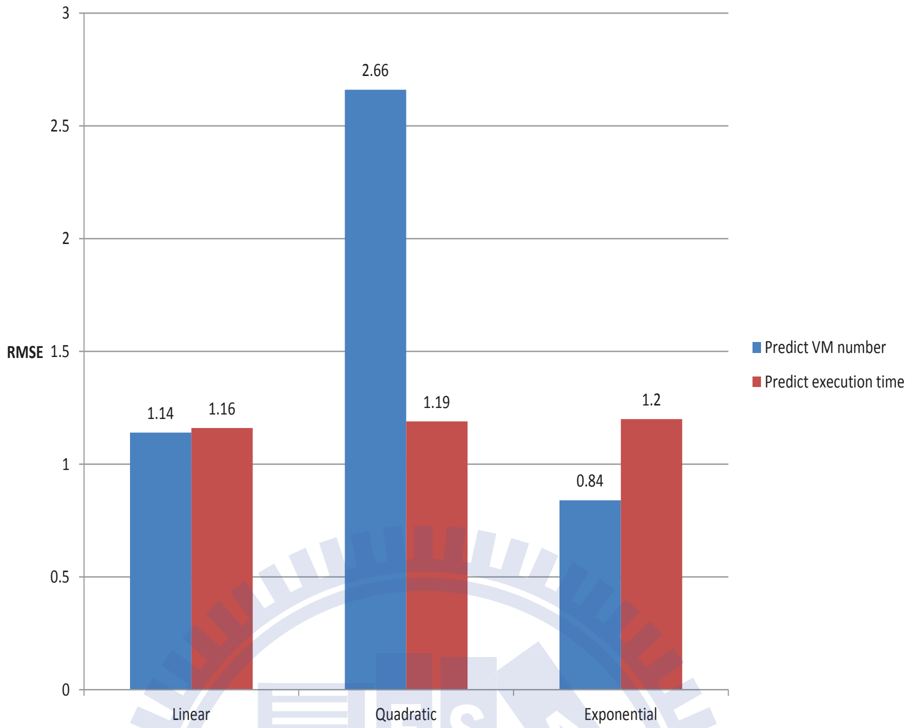


Figure 5.11: RMSE of prediction algorithm.

repeated until the the clusters are not changed anymore. The example of K-means is shown in Fig. 5.12.

Fig. 5.13 shows the execution process of parallel K-means. The input data node A,B and D is allotted into different mapper to compute the distance between it and every central node C1, C2. After mappers output the distance result of A,B and D, the reducers have two step to execute. First, the reducers find the closest cluster for A,B and D according to the distance result. Therefore, B is changed to cluster 2. Second,the reducers recompute the central node of every cluster according to the average value of the data node in the clusters. The central node of cluster 2 is changed to D. The mapper and reducer process are iterative executed until the clusters are not changed anymore.

5.2.2 Parallel K-means on RPF

We execute parallel K-means on map-reduce platform with different K. Therefore, we can collect different execution parameters, such as K and execution time to be the input of training algorithm. In Fig. 5.14, we input $k = 1, 3..., 21$ and the historical execution time in the training algorithm. After executing training algorithm, we can obtain an

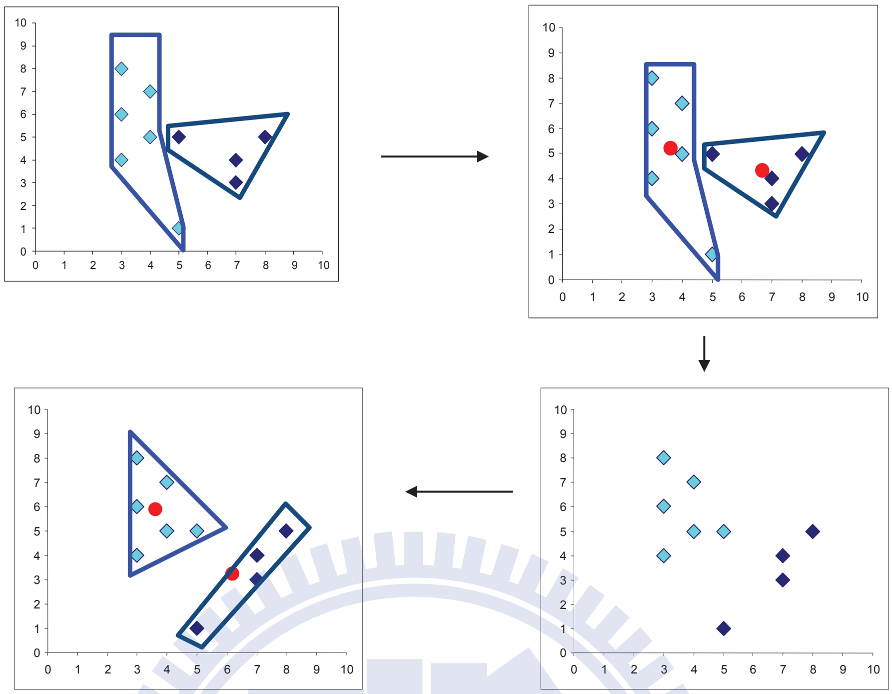


Figure 5.12: The example of K-means.

regression model whose root mean square error (RMSE) is the smallest. We use this regression model to predict the number of execution VMs for the new jobs. We input the new job's K which is 10 and expected response time 2000 which is set by users in the prediction algorithm. We put the K, the expected response time and the regression coefficient set A which is the output of training algorithm into the prediction equation. The equation outputs the fewest number of VMs. The example of prediction algorithm is shown in Fig. 5.15. If the prediction result is not accurate enough, we can try to use prediction algorithm 3 to estimate the number of execution VMs. Fig. 5.16 shows the example of prediction execution time methods. We put the K, the number of VMs and the regression coefficient set A which is the output of training algorithm into the prediction equation. The equation outputs the execution time of every situation. Finally, we can use the expected execution time and the result of prediction algorithm to decide the fewest number of VMs of the new jobs.

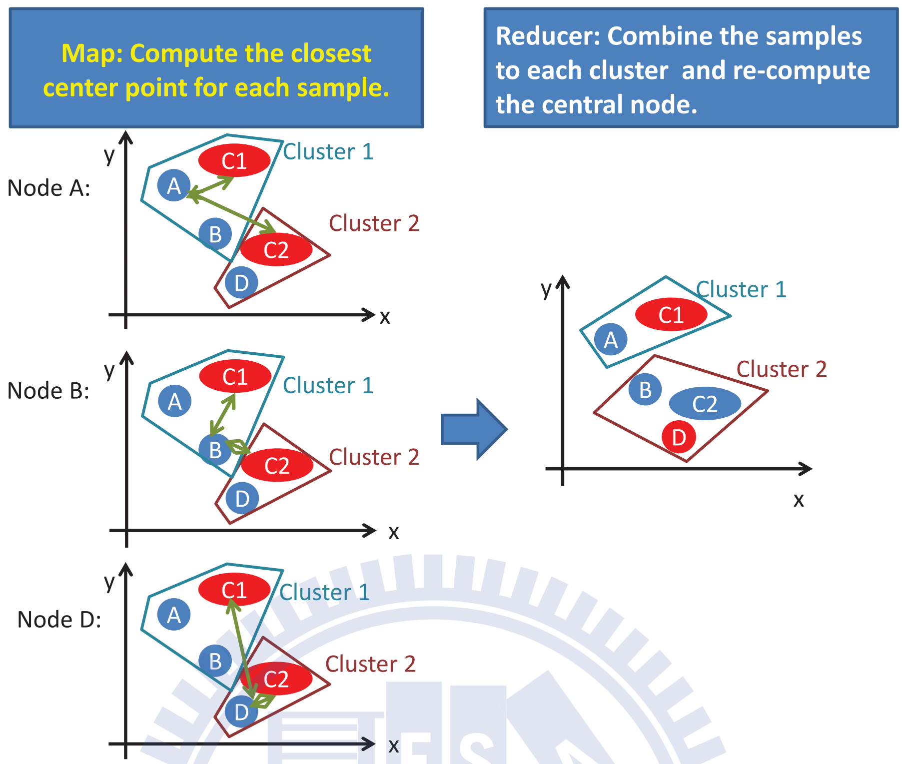


Figure 5.13: The execution example of parallel K-means.

5.2.3 Parallel K-menas on RPF Experiment Setup

The experiment environment is same with the FP-growth experiment. We refer to [15] and Apache Mahout project [10] to modify the K-means in parallel.The dataset is also same with FP-growth dataset which is rating data sets from the MovieLens web site [8]. We choose the dataset which consists of 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. We use K-means to cluster the users who have the same interest of movies.

5.2.4 Evaluation

In prediction algorithm 1, we use linear and non-linear regression to predict the number of VM. The linear and non-linear prediction curves are shown in Fig. 5.17.

In prediction algorithm 2, we use linear and non-linear regression to predict the execution time. The linear and non-linear prediction curves are shown in Fig. 5.18. Fig. 5.19 shows the RMSE of prediction algorithm 1 and prediction algorithm 2. We can see that the prediction result of quadratic function is the most accurate and the result of linear

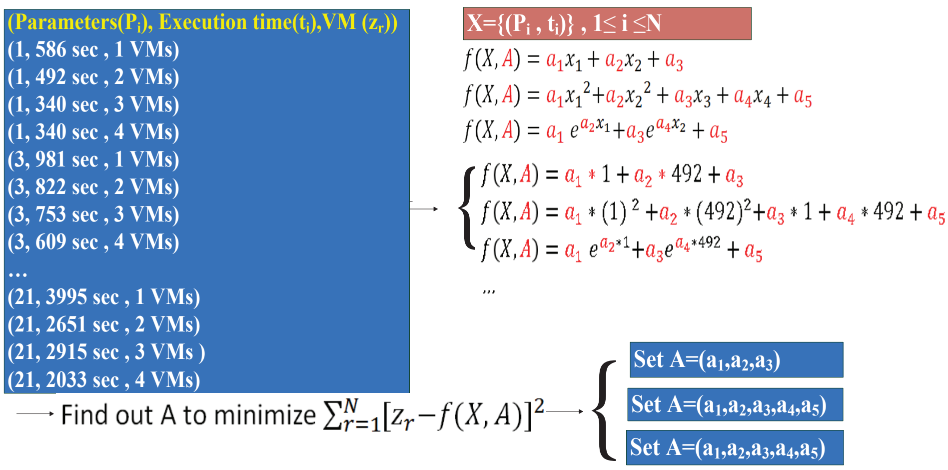


Figure 5.14: The example of training algorithm by K-means.

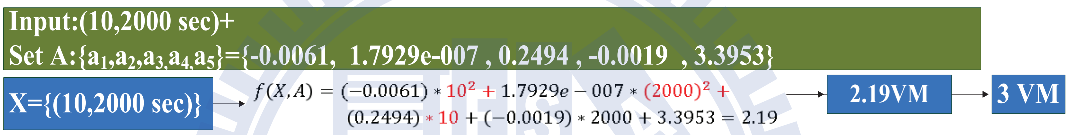


Figure 5.15: The example of prediction algorithm 1 by K-means.

function has the maximum error. We also can find that the RMSE of RPF is no more than 1.5. In brief, RPF can predict the fewest number of execution VMs to users accurately.

5.3 PSO

Particle Swarm Optimization (PSO), an optimization algorithm that was inspired by bird and fish foraging social behavior [4]. In the beginning, the birds have no idea about the food location. They guess and fly to better location by their experience and intuition. When a bird finds the food, it broadcasts the location to other birds. Other birds fly to the food location. Bird foraging behavior is the concept of mutual influence in society which can lead all individual bird toward the location of the optimal solution. PSO has become popular because it is simple, requires little tuning, and has been found to be effective for a wide range of problems. Since every node in PSO can compute the local best value by itself, PSO is suitable for MapReduce framework. In this paper, we refer to [7] to modify PSO in parallel and demonstrate parallel PSO on RPF.

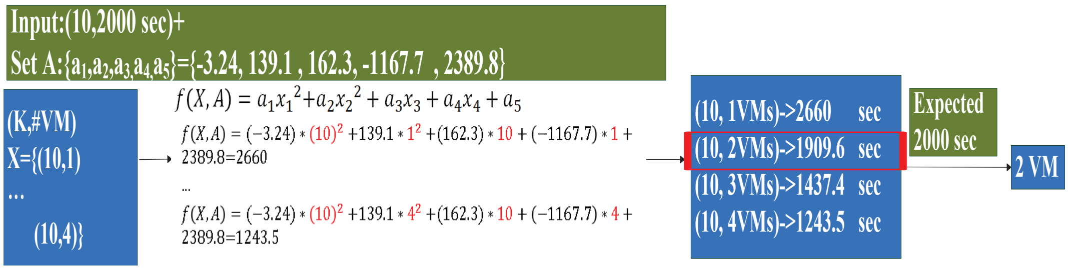


Figure 5.16: The example of prediction algorithm 2 by K-means.

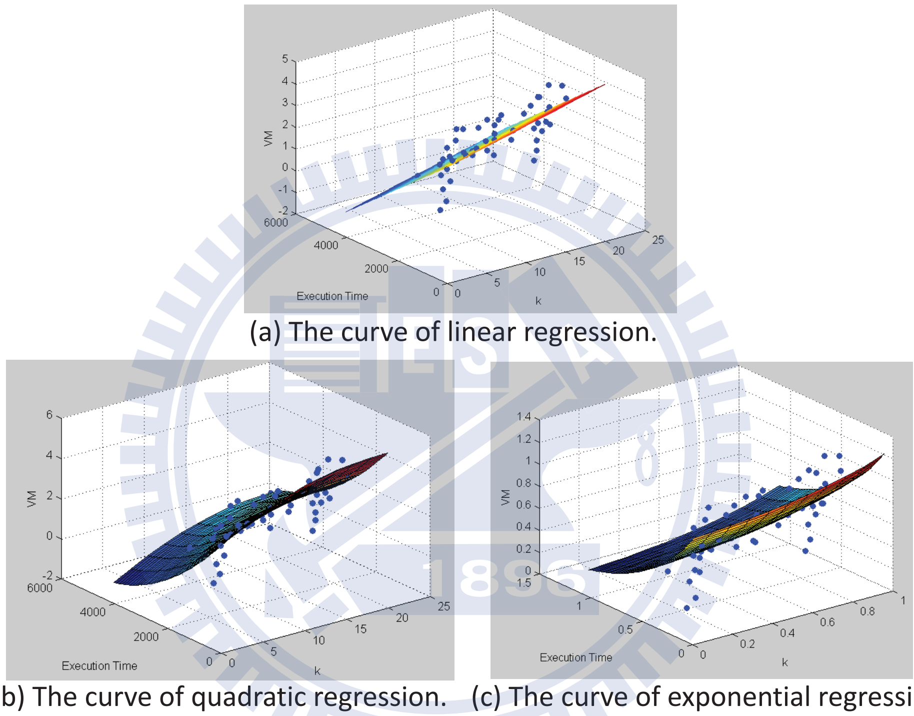


Figure 5.17: The prediction curve of K-means by prediction algorithm 1.

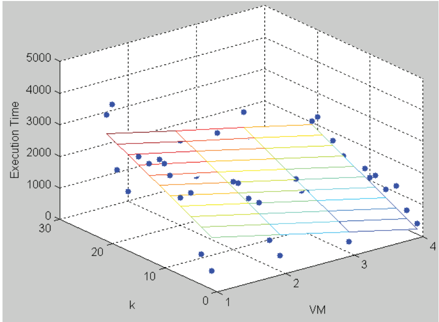
5.3.1 Parallel PSO on RPF

The steps of resource prediction of parallel PSO is same with parallel FP-growth and parallel K-means. We execute parallel PSO with *Rosenbrocks* test function [9] which is defined following by different dimension d.

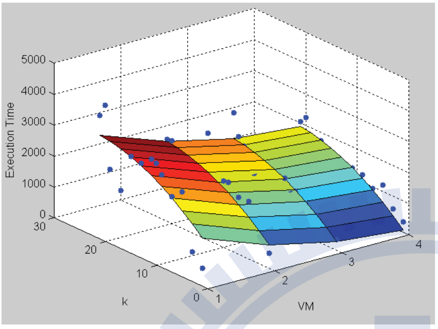
$$f(x) = \sum_{i=1}^{d-1}\left[(1-x_i)^2 + 100\left(x_{i+1}-x_i{}^2\right)^2\right]$$

$x_i \in [-100, 100]$.

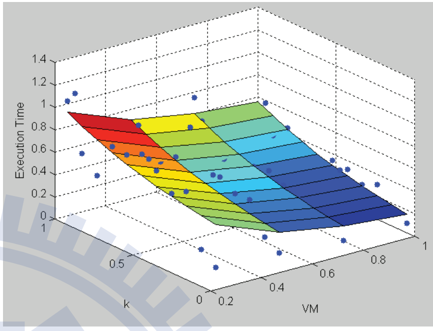
We collect the different parameter d and execution time to be the input of training



(a) The curve of linear regression.



(b) The curve of quadratic regression.



(c) The curve of exponential regression.

Figure 5.18: The prediction curve of K-means by prediction algorithm 2.

algorithm. After the process of training, we predict the minimum number of VMs of a new job with different d by algorithm 2 or algorithm 3. The performance of parallel on RPF is shown below.

5.3.2 Parallel PSO on RPF Experiment Setup

The experiment environment is same with the FP-growth and K-means experiment. The input dataset is generated by latin hypercube sampling [6] with different domain. The optimization function of PSO is Rosenbrocks test function and $x_i \in [-100, 100]$. We use PSO to find optimal solution of Rosenbrocks function.

5.3.3 Evaluation

We use linear and non-linear to predict the number of VM in prediction algorithm 2. The curve of linear and non-linear prediction are shown in Fig. 5.20.

We use linear and non-linear regression to predict the execution time. The linear and non-linear prediction curves are shown in Fig. 5.21. The RMSE of prediction algorithm 1 and prediction algorithm 2 is shown in Fig. 5.22. We can see that the prediction result of

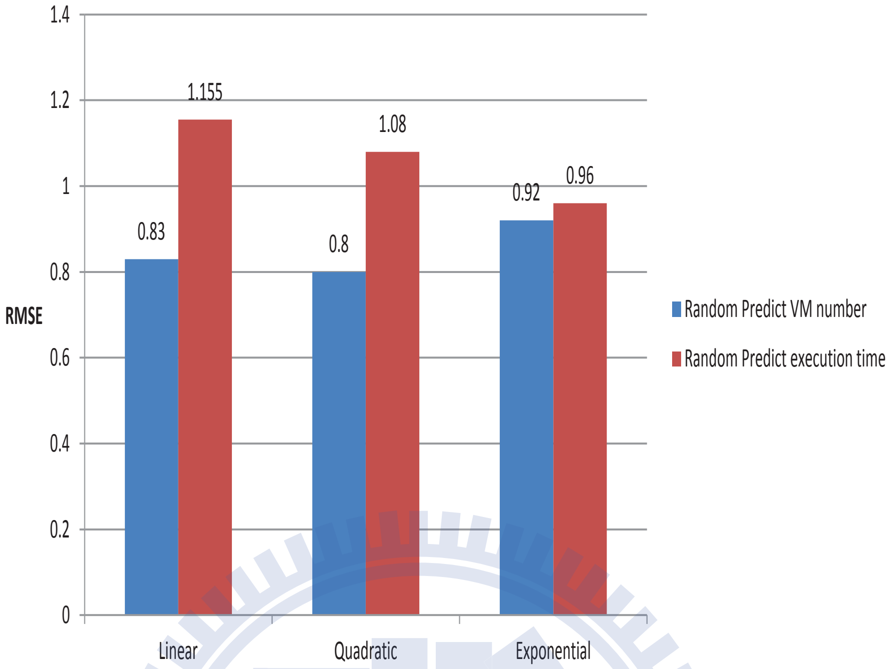
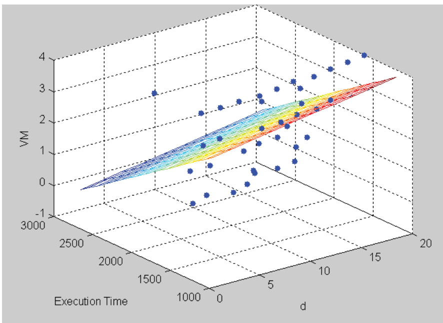
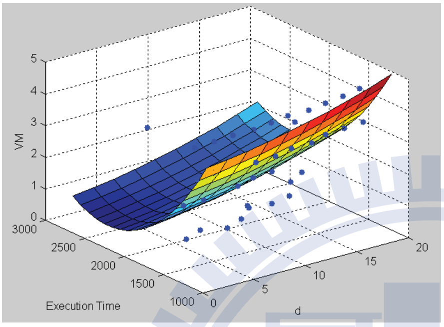


Figure 5.19: RMSE of prediction algorithm.

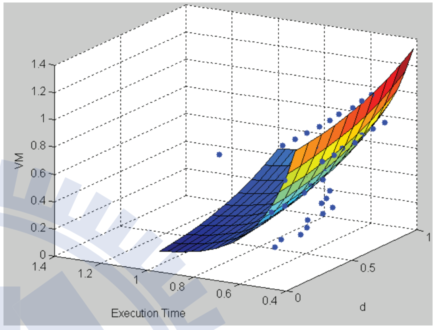
algorithm 2 is more accurate than algorithm 1. The reason is that parallel PSO program is not thorough parallel. That is, while users' program is not fully parallel, algorithm 3 can provide better solution.



(a) The curve of linear regression.

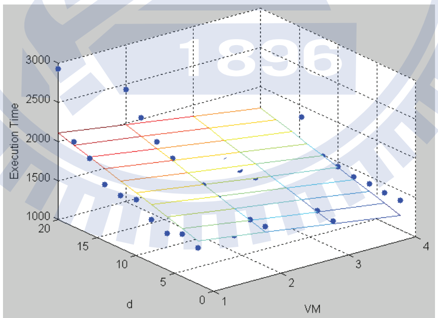


(b) The curve of quadratic regression.

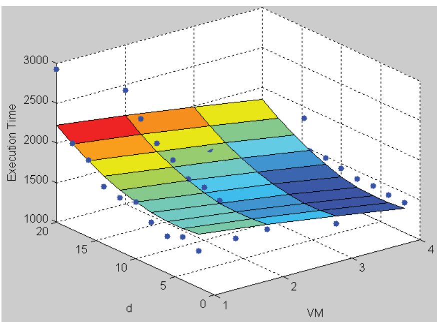


(c) The curve of exponential regression.

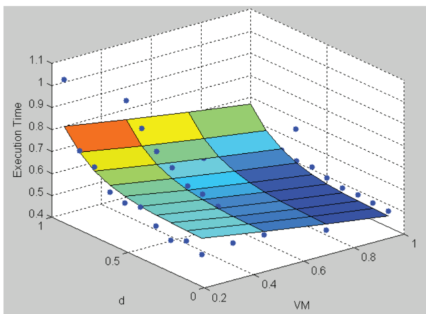
Figure 5.20: The prediction curve of PSO by prediction algorithm 1.



(a) The curve of linear regression.



(b) The curve of quadratic regression.



(c) The curve of exponential regression.

Figure 5.21: The prediction curve of PSO by prediction algorithm 2.

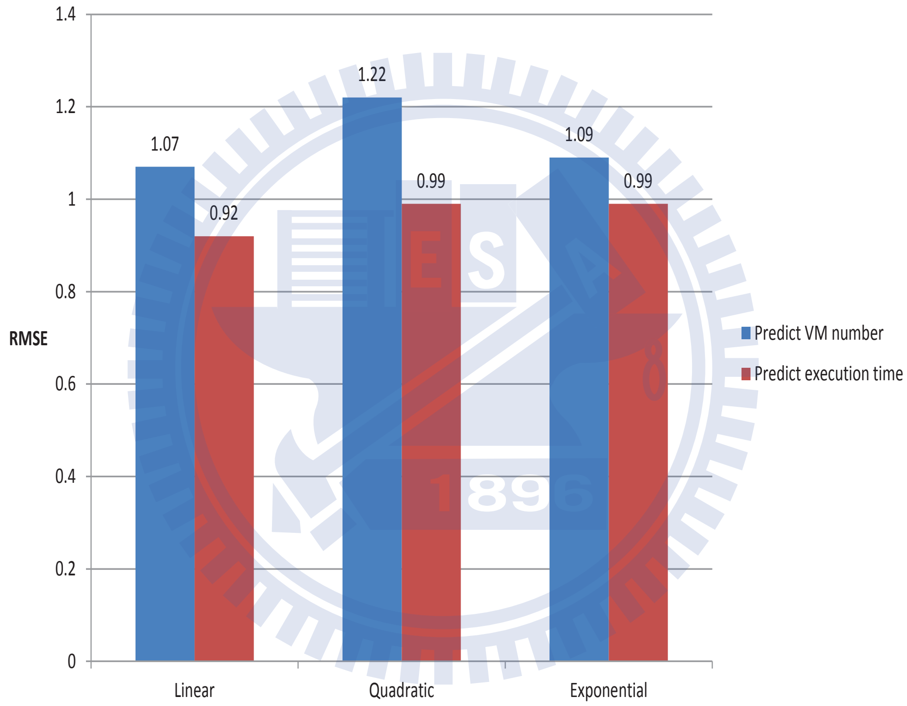


Figure 5.22: RMSE of prediction algorithm.

Chapter 6

CONCLUSION

In this paper, we proposed a resource prediction framework (RPF) to predict the minimum number of execution VMs which can execute the users' jobs within a user specified response time. We not only proposed RPF but also demonstrated parallel FP-growth, parallel K-means and parallel PSO on RPF to evaluate the performance of RPF. The evaluation results showed that RPF can predict the number of VM accurately and can be adopted by data intensive algorithms. This is a big progress in resource provisioning field. In the future, we will propose a VM allocation model to combine with RPF which can make resource provisioning more accurately.

Bibliography

[1] The bigcross datasets. http://www.cs.uni-paderborn.de/en/fachgebiete/ag-bloemer/research/clustering/streamkmpp.

[2] A. Frank and A. Asuncion. Uci machine learning repository. *http://archive. ics. uci. edu/ml*, 10, 2010.

[3] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya. SLA-based Resource Provisioning for Heterogeneous Workloads in a Virtualized Cloud Datacenter. In *IEEE International Conference on Algorithms and Architectures for Parallel Processing*, 2011.

[4] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

[5] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Chang. PFP: Parallel FP-Growth for Query Recommendation. In *Proceedings of the ACM conference on Recommender systems*, 2008.

[6] M. McKay, R. Beckman, and W. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, pages 239–245, 1979.

[7] A. McNabb, C. Monson, and K. Seppi. Parallel pso using mapreduce. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 7–14. IEEE, 2007.

[8] Movielens data sets. http://www.grouplens.org/node/7.

[9] J. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[10] S. Owen, R. Anil, T. Dunning, and E. Friedman. *Mahout in action.* Manning Publications Co., 2011.

[11] G. Reig, J. Alonso, and J. Guitart. Deadline Constrained Prediction of Job Resource Requirements to Manage High-Level SLAs for SaaS Cloud Providers. *IEEE International Symposium on Network Computing and Applications*, 2010.

[12] M. Shindler, A. Wong, and A. Meyerson. Fast and accurate k-means for large datasets.

[13] F. Tian and K. Chen. Towards Optimal Resource Provisioning for Running MapReduce Programs in Public Clouds. In *IEEE Conference on Cloud Computing*, 2011.

[14] P. Xiong, Y. Chi, S. Zhu, H. J. Moon, C. Pu, and H. Hacigumus. Intelligent Management of Virtualized Resources for Database Systems in Cloud Environment. In *IEEE International Conference on Data Engineering*, 2011.

[15] W. Zhao, H. Ma, and Q. He. Parallel k-means clustering based on mapreduce. *Cloud Computing*, pages 674–679, 2009.