# 在格網計算上用於資料探勘之

# 決策樹建構

研究生：蔡淑姿　　　　　　　　指導教授：楊朝棟　博士

東海大學理學院

資訊工程與科學系

## 摘要

決策樹是資料探勘裡尋找未知資訊技術中一個很常被使用到的方法。由於它的特性適合做平行化，因此廣受高效能計算領域研究的青睞，發展各種平行決策樹演算法來處理大量資料及更複雜的運算。隨著科技不斷的進步，這幾年發展出來的格網計算可以說是叢集電腦的一個延伸，而成為很受重視的一個研究發展方向，它是繼傳統網際網路、Web 之後的第三次網際網路浪潮，可以稱之第三代網際網路應用。

在這篇論文裡，我們提出一個格網決策樹架構圖，希望決策樹應用軟體中不論平行或單一的演算法，都可以運用在格網計算的環境裡。同時根據資料探勘在格網環境裡依其應用的範圍及模式，就使用者對等的角度來看，將格網的角色定位分為三種類型。希望藉由這三種類型的定義，軟體開發者可以很清楚的定義系統流程，並區隔應用軟體的應用範圍。為了實現我們所提的架構圖，首先試著將一個現有的平行決策樹演算法(SPRINT algorithm)放在格網環境中，並用不同大小的資料，去比較該演算法在格網計算中的效能性及變化。我們將根據這個實驗結果做為未來工作的一個重要參考數據。

# Decision Tree Construction for Data Mining on Grid Computing

Student: Shu-Tzu Tsai Advisor: Dr. Chao-Tung Yang

Department of Computer Science and Information Engineering
Tunghai University
Taichung, 407, Taiwan, Republic of China

## Abstract

Decision tree is one of the frequently used methods in data mining for searching prediction information. Due to its characteristics which are suitable for parallelism, it has been widely adopted in high performance field and developed into various parallel decision tree algorithms to deal with huge data and complex computation. Following the development of other technology fields, Grid computing is regarded as the extension of PC Cluster and therefore it future research development is highly valued. This new wave of internet application is the 3rd generation of internet applications following the traditional internet and Web application.

In this thesis, we have presented the Grid-based decision tree architecture, and hope it can be applied on both parallel and sequential algorithms for the decision tree applications. Also, based on the scope and model of data mining applied in the Grid environment as well as user equivalent perspective, Grid roles can be categorized into three types. We are hoping that through these definitions, software developers can define clear system processes and differentiate the application scope for software applications. To fulfill our architecture, we first apply an existing parallel decision tree algorithm-SPRINT algorithm in the Grid environment. The performance and differences in many other areas are compared using different sizes of dataset. The experimental results will be used for future reference and further development.

# Acknowledgements

I would like to thank all the people who have made writing this thesis a more pleasant task. In particular, I would like to thank my principal advisor, Dr. Chao-Tung Yang, who introduced me to this topic and gave me broad support and guidance throughout my time as Tunghai. I would like to thank Professor Miao-Tsong Lin, Professor Yi-Min Wang and Professor Fang-Yie Leu for their valuable comments and advice given while serving on my reading committee.

There are many other people whom I would like to thank. Especially, the mates of High Performance Computing Laboratory, I could not achieve the experiment without their support. Also many classmates encouraged and supported me on studying. For them, I can make writing this thesis with no fear of disturbance in the rear.

Last, but certainly not the last, I would like to thank that my family and my boy friend whose unconditional support made this thesis possible.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Data Mining

Currently data mining has been widely studied in many different research communities. Data mining is defined as the persistent and interactive process of discovering valid, up-to-date, useful, and understandable models in data sets [4]. Therefore it can be applied in almost every domain that wants to find out the relationship among data. So actually data mining activity has been existed in the human history for a very long time. The only differences now are the amount of data is larger and the complexity of computation is increasing.

From the computer scientific viewpoint, the tools include a set of algorithms and techniques that were developed for extracting interesting patterns from huge databases is called Data Mining (DM) [13]. Typical data mining has two goals of prediction and description. The data mining architecture is shown in Figure 1.1. The discipline of data mining roots in the different traditional research fields of machine learning, statistic, and the algorithms are similar to those domain areas.

As to the algorithm of data mining, there are three main approaches to data mining, 1. I suspect 2. I know who, now tell me why, OR now predict who else 3. Tell me something interesting [2]. According above characteristics, data mining main tasks are association rules, classification, clustering, neural network and regression [13]. The function of association rules is to determine implication rules for a subset of record attributes; it is a part of description research. Classification which assign each record of a database to one of a predefined set of classes, it is a part of prediction area.

Clustering that find groups of records that are close according to some defined metrics.

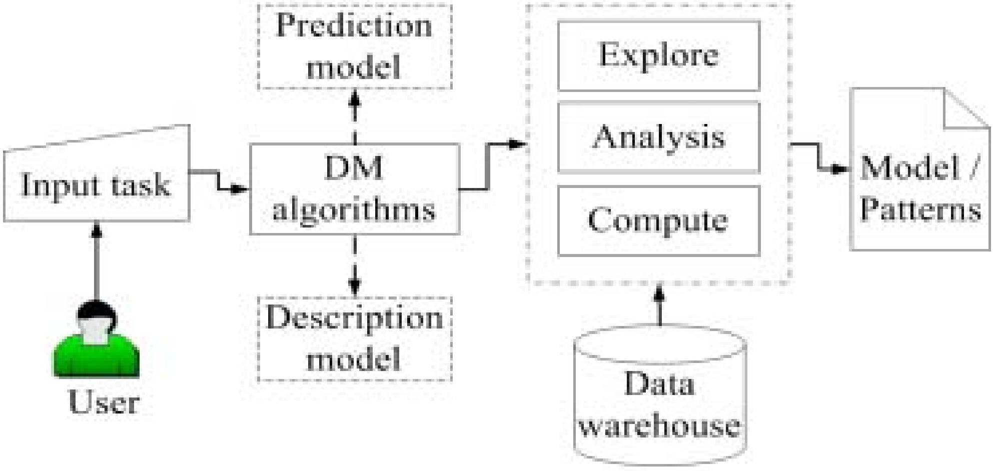


**Figure 1.1: The data mining architecture**

As the society changes rapidly, the commerce becomes more complex. There are economic and scientific needs to extract useful information from the collected data by computer technology. Now, more and more data has been gathered and stored and the field of mining is getting wider and more various. Also the development of computer is improving day by day. Thus the researchers have developed some algorithms and techniques that resolve for data mining in massive data sets for the human's demand. Because the huge data, we need more efficiency and effect parallelism. Now there are Symmetric Multiprocessors (SMP) and Cluster for high performance computing. Gradually the researchers are toward to modify the existing algorithms or propose new model for applying the parallel and distributed environment, either association rule or classification. Like SPIDER (scalable parallel interactive data mining and Exploration at Rensselaer) [26] and IBM DB2 Intelligent Miner [27]. Both are famous parallel data mining.

## 1.2 Parallel and Distributed Computing

Parallel and distributed computing is the most important technology for several decades. Seriously, parallel computing is that CPUs do the same things at the same time. Distributed computing is that CPUs do the same or different things at the same time. Both purposes are the fast solution of computationally large and data-intensive problems. To utilize supercomputer for high-performance computing has been common. Many powerful multiprocessor hardware systems have been developed to exploit parallelism for concurrent execution. The supercomputer that is a single big expensive machine with a shared memory and one or more processors meet the professional need. A large-scale processing and storage system that provides high bandwidth at cost-effective is expected.

From hardware viewpoint, there are three methods in parallel processing, Shared Memory Multiprocessor System, Distributed Memory Multiprocessor System (DMM) and Clustering System. The first also named Symmetric Multiprocessors (SMP) is shown in Figure 1.2, the second named Massive Parallel Processors (MPP) is shown in Figure 1.3. The Clustering system contains many the same or different individual common personal computers (PCs) is shown in Figure 1.4. Each cluster node has its own disk and equipped with a complete operating system, and therefore, it also can handle interactive jobs. Each node can function only as an individual resource while a cluster system presents itself as a single system to the user. A network is used to provide inter-processor communications. Applications that are distributed across the processors of the cluster use either message passing or network shared memory for communication.

Due to the PC Cluster is more cost-effective than supercomputers, most research communities can afford it. It is easy to build a unique Cluster system from available components. So the research resources of Cluster system have been proposed more and more. Now the standard message-passing middleware, such as Message Passing

Interface (MPI) and Parallel Virtual Machine (PVM) are mature, promote parallel computing applying in many different domains.

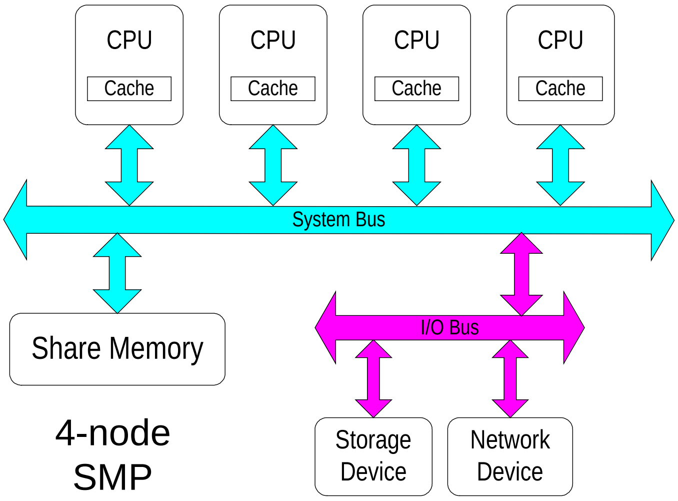


**Figure 1.2: The SMP architecture**

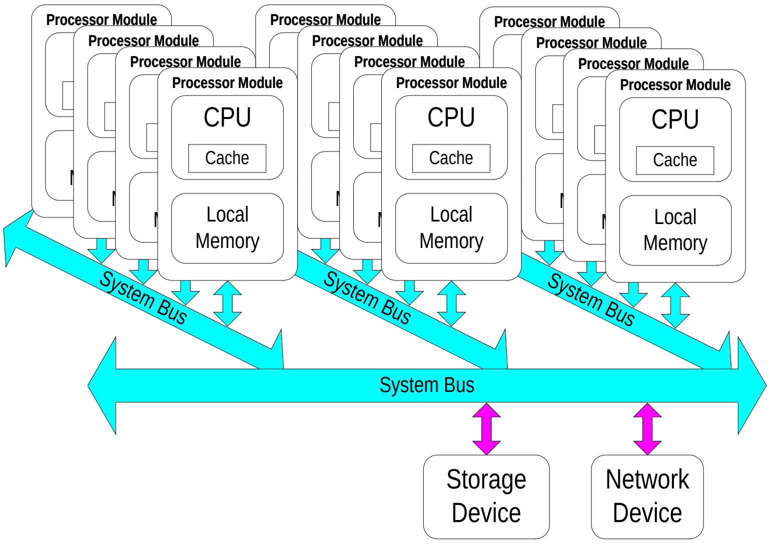


**Figure 1.3: The MPP architecture**

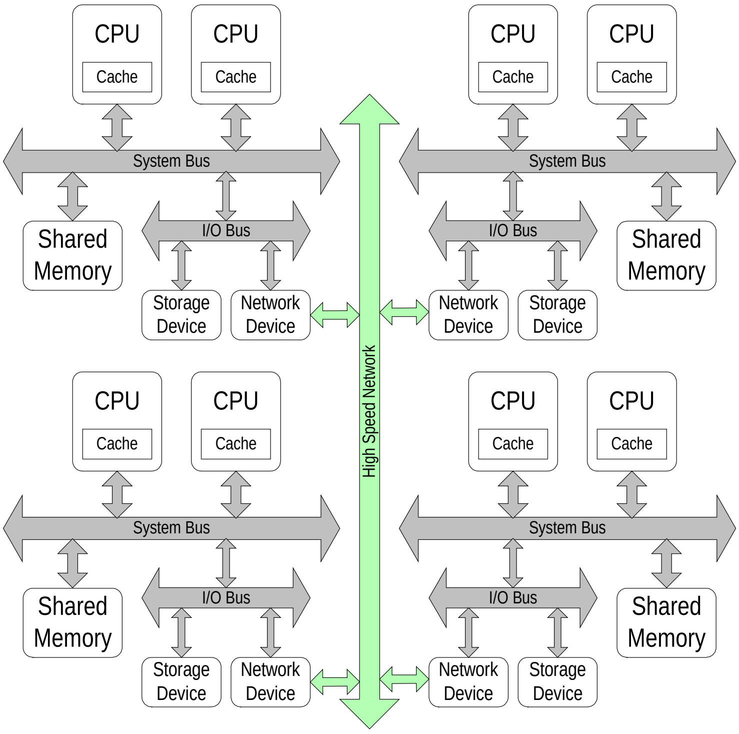


**Figure 1.4: The Cluster system architecture**

After the PC Cluster which cost is lower and has been widely applied, Grid computing has emerged recently as an integrated resource infrastructure for high-performance distributed computation through internet, so it can process huge data. It is the third generation internet application after the tradition internet and the Web. Truly, parallelism and distribution are the foundation in the Grid. The structure of Grid is similar to the extending Cluster. The PC Cluster has become a hot research topic in the field of parallel and distributed computing. Many methods and applications have came maturity, and established a good foundation for Grid.

## 1.3 Motivation

Why does data mining need parallel formulation? 1. To handle very large datasets. 2. Memory limitations of sequential computers cause sequential algorithms to make multiple expensive I/O passes over data. 3. Need for scalable, efficient and fast computation [4]. 4. To process more complex data, such as image, genome, geographical, ecological data. In general, data mining parallelism distinguish between data and task. Data parallelism is that make data to split among processors and each processor performs the same task for its own local data, scheduling work associated with different attributes to different processors. Also parallel database system, like parallel ORACLE, is the other method. Task parallelism is almost based on algorithms, that is, to partition into different or same task, each processor performs its own computation and the data probable replicated or partitioned.

Due to the amount of data is increasing, algorithms become more complex. In addition, SMP and Cluster have been mature and popular, specifically, the message passing language like PVM (Parallel Virtual Machine) or MPI (Message-Passing Interface) is complete, so parallel algorithms have been developed for high-performance computing. The ultimate goal is cost reduction and speed up. Now the internet is wide-spreading and convenient, and also the quality of network width is highly improving. Therefore data mining also extended into the field of Word Wide Web. The concept of Grid has been emerged recently as an integrated resource infrastructure for high-performance parallel and distributed computation over a wide area-network [4]. The point is that Grid is not only extending parallelism but also sharing the various resources of internet. So the information science and technology revolution has extended from SMP and Cluster to the Grid.

Grid computing has caught a lot of attention in recent years. Currently, some applications on Grid-based data mining infrastructures have been proposed, such as the Knowledge Grid (K-Grid) [15], NASA's Information Power Grid (IPG) [16].

Through Grid collaboration, it can make data mining tasks originally thought too difficult or complex to become possible. And all can be implemented in every different research field, despite the fact that current Grid-based data mining applications still have many problems and limit to different scientific communities. But from the perspective of the network development and global common objectives, Grid is workable and useful.

## 1.4 Contributions

We explore the detail relationship between Grid architecture and data mining, we think Grid can have three models: Top-Down, Reciprocal type and Broadcast type. These models can let people clearly know data mining application model and user equality perspective in the Grid environment. Also we present the architecture for the decision tree model that can be created in Grid computing environment for data miming. We hope through Grid collaboration, it can make data mining tasks originally thought too difficult or complex to become possible.

Of course, we want to know the performance of Grid computing. So the first thing we tried to compare the performance between Cluster computing and Grid computing. In the beginning, our principle is not to modify the original program of Cluster. Therefore, in this preliminary experiment we just transplanted the program of Cluster into Grid. The transplant is successful, although the experiment result, Grid performance is less currently.

Our experiment also reveals the other appearances of Grid. First, speedup will has a little down when processor work first time crosses over another platform, but speedup goes back later. It is worth notice, because it will affect the Grid type that consists of plenty single workstations. Second, the turn around time of multiple processor does not double increase flowing the double dataset, oppositely, the speedup is closed. It is satisfying.

## 1.5 Thesis Organization

This thesis starts with the brief introduction of data mining, parallel and distributed computing concept and motivation of parallel data mining. In chapter 2, we discuss the background of decision tree, Grid computing. And we consider why need to apply Grid computing technologies to decision tree. In chapter 3, we think Grid can have three models for the data mining application model and user equality perspective in the Grid environment, and present the architecture for Grid-based decision tree. In chapter 4, we show the performance between PC Cluster and Grid, and discuss the result of experiment. We have related work for our next experiment and introduce it in the chapter 5. Finally we make a brief conclusion and future work in chapter 6

# Chapter 2

# Background

## 2.1 Decision Tree

Typical data mining has two goals of prediction and description [4]. In former field, some people argue that the classification of data is the most important model of data mining and the most commonly applied technique [7]. Among classification algorithms, the decision tree that shown in Figure 2.1 is probably the most popular and practical method. A decision tree is composed of three types of nodes: the *root* node, the *intermediate* node and the *leaf* node. Because it is relatively fast to compute, it is fairly simple to interpret by human [5], also straightforwardly be converted into SQL statements that can be used to access databases efficiently [16].

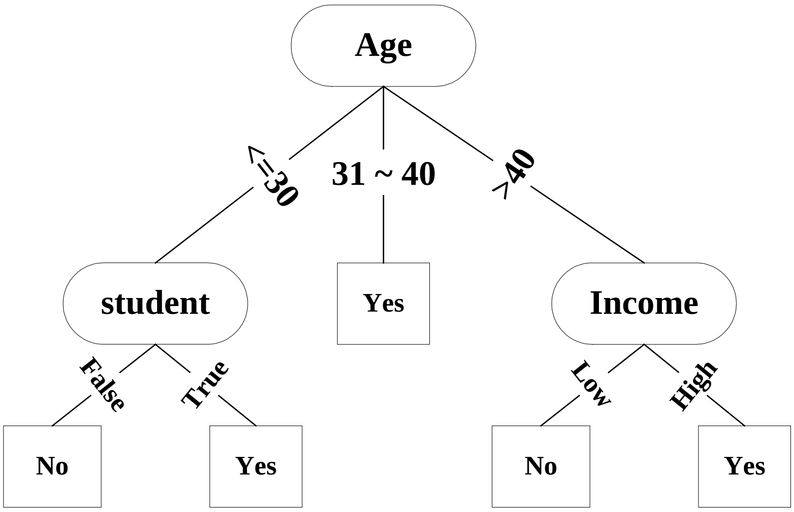


**Figure 2.1: A decision tree model for buying computer**

A classification problem has an input dataset called the training data set. And the dataset consists of a number of records, and each consists of several fields called attributes. Attributes can be continuous, coming from an ordered domain, or categorical, coming from an unordered domain. One of the attributes, called the class attribute, indicates the class to which record belongs. A decision tree is built on two phases: a construction phase and a prune phase [19]. First, an initial tree is built till the leaf nodes belong to a single class only. Second, pruning is done to remove any over fitting to the training data. Typically, time is spent most on construction phase, so we focus on the tree generation only.

While building the tree, there are three main steps [20] that must be performed for each node at each level of the tree:

(1) Evaluate split points for each attribute.

(2) Find the winning split-point for a node.

(3) To split all attribute lists into two parts, one for each node.

### 2.1.1 Gini index

Growing the decision tree, the goal of each node is to determine the split point best divides the training records associated with a node. Several splitting modes have been proposed to evaluate the best of the split. SPRINT algorithm use *gini* index which many algorithms used for this purpose [10].

For a data set T containing records from *N* classes, *gini*(T) is defined as:

$$gini(\mathrm{T}) = 1- \sum \mathrm{P}_j{}^2$$

where $p_j$ is the relative frequency of class *j* is T.

If T is partitioned into two subsets $T_1$ and $T_2$, the index of the divided data $gini_{split}$ (T) is given by:

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

The advantage of this gini index is that its calculation requires only the class value distribution at each internal node of the tree. For categorical attributes, all possible subsets of the attribute values are considered as potential split points. For continuous attributes, the candidate split points are mid-points between every two consecutive attribute values in the sorted data. To find the best split point for a node, scan each of the node’s attribute lists and choose the lowest value for the *gini* index is then used to split the node. We show the computing example of *gini* index in Table 2.1. It therefore requires ongoing calculation based on two adjacent values. When data amount is large, the calculation workload will be heavy, specifically the continuous attribute. That is what we focus on.

Rule:
If one value=0, then gini(t)=0
If two values is equal, then gini(t)=0.5
Computation:
Gini(97) = (6/10) * 0.5 +(4/10) * 0 = 0.3
Split point is the lowest value of gini index

**Table 2.1: The computing example of gini index**

| Credit | No | No | No | Yes | Yes | Yes | No | No | No |
|---|---|---|---|---|---|---|---|---|---|
| Income (10K)<br>(Sorted Values) | 60 | 70 | 75 | 85 | 90 | 95 | 100 | 110 | 120 |

| Split positions | 55 | | 65 | | 72 | | 80 | | 87 | | 92 | | 97 | | 105 | | 115 | | 125 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | <= | > | <= | > | <= | > | <= | > | <= | > | <= | > | <= | > | <= | > | <= | > | <= | > |
| Yes | 0 | 0 | 0 | 3 | 0 | 3 | 0 | 3 | 1 | 2 | 2 | 1 | 3 | 0 | 3 | 0 | 3 | 0 | 3 | 0 |
| No | 0 | 0 | 1 | 6 | 2 | 5 | 3 | 4 | 3 | 4 | 3 | 4 | 3 | 4 | 4 | 3 | 5 | 2 | 6 | 1 |
| Gini | 0.420 | | 0.400 | | 0.375 | | 0.343 | | 0.417 | | 0.400 | | 0.300 | | 0.343 | | 0.375 | | 0.400 | |

## 2.1.2 Parallel Decision Tree

A decision tree model employs a recursive divide-and-conquer strategy to divide the data set into partitions so that all of the records in a partition have the same class label. According this feature, it is easier to parallel. While classification is a well-studied problem, only recently has there been focus on algorithms that can handle large databases. Therefore, several decision-tree-based parallel methods proposed over the years. Such as parallel SPRINT on IBM SP2 [10], ScalParC on a Cray T3D [12], Parallel C4.5 [5], RainForest [9], etc.

Some algorithms assumed that training sets could fit in memory. The others can handle disk-resident data include SLIQ and its extension SPRINT, or they use distributed databases for parallel. No matter which model is used, all they can improve for parallelism. If it assumes the dataset fit in memory, then they can be distributed to different memory, or if handle disk-resident then can distribute data to the local disk on the computers of parallelism. The simple parallel process is shown in the Figure 2.2.

```
forall attributes in parallel
    for each leaf
        evaluate attributes
barrier
if (master) then
    for each leaf
        get winning attribute
barrier
forall attributes in parallel
    for each leaf
         partition attributes into two parts
```

**Figure 2.2: The basic parallel process**

## 2.2 Grid Infrastructure

Grid infrastructure supports the sharing and coordinated use for resources in dynamic global heterogeneous distributed environments. It involves resources that can manage computers, data, telecommunication, network facilities and software applications provided by different organizations [1]. Namely, a Grid is a collection of distributed computing resources available over a local or wide area network that appears to an end user or application as one large virtual computing system. Currently, the researchers use an XML-based approach for managing heterogeneous resources [1]. The Figure 2 bellowed, can help us to understand what Grid is.

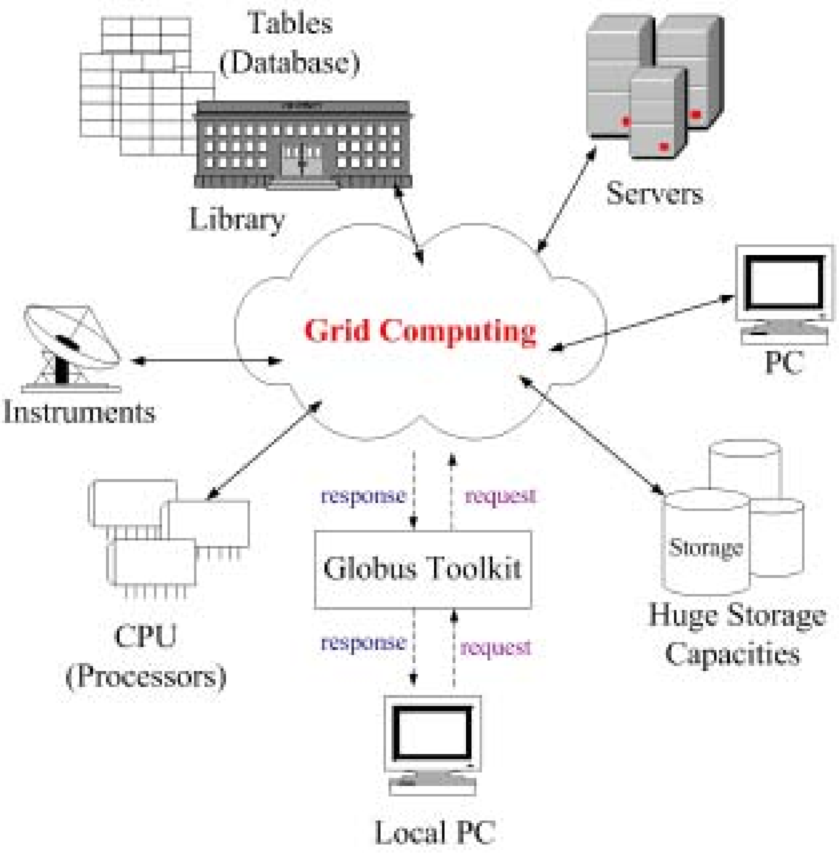


**Figure 2.3: The Grid computing infrastructure**

### 2.2.1 Globus Toolkit

The Grid represents the common properties have very large, distributed, dynamic and cross the boundaries of human organizations [17]. Due to these characteristics, its architecture must be very complex and its relationship is also like mazy. The critical point to realize Grid is to establish a common open standard. At its core is based on an open set of standards and protocols - e.g., Open Grid Services Architecture (OGSA) - that enable communication across heterogeneous, geographically dispersed environment. With Grid computing, organizations can optimize computing and data resources, pool them for large capacity workloads, share them across networks and enable collaboration [21]. Some protocols and applications have been proposed and actual work, such as Globus Toolkit, DICE and the NASA’s information Power Grid [1].

The Globus Toolkit is an open architecture, open source software of services and software libraries that support Grids and Grid applications. It facilitates the creation of computational Grids. It provides software tools enabling coupling of people, computers, databases and instruments. With Globus, you can run job on two or more high-performance machines at the same time, even though the machines might be located far apart and owned by different organizations. Globus software helps scientists deal with very large datasets and complex remote collaborations. The toolkit includes software for security, information infrastructure, resource management, data management, communication, fault detection, and portability.

The toolkit components that are most relevant to OGSA are the Grid Resource Allocation and Management (GRAM) protocol, the Meta Directory Service (MDS-2) and the Grid Security Infrastructure (GSI). These components provide the essential elements of a service-oriented architecture.

- GRAM protocol: provides for secure, reliable, service creation and management of arbitrary computations.

- MDS-2: provides a uniform framework for information discovering through soft state registration, data modeling and a local registry.
- GSI protocol: supports single sin on, authentication, communication protection and certification mapping.

### 2.2.2 Web Services

In the Grid environment, the critical point how to combine the different heterogeneous resources. Web services defines a technique for describing software components to be accessed via internet, it is a communication media between different platforms. Web services standards are defined within the W3C that the most industries support, and interact between the components in the service processes are based on XML, SOAP, WSDL and UDDI. Web services architecture is shown in Figure 2.4. [22]

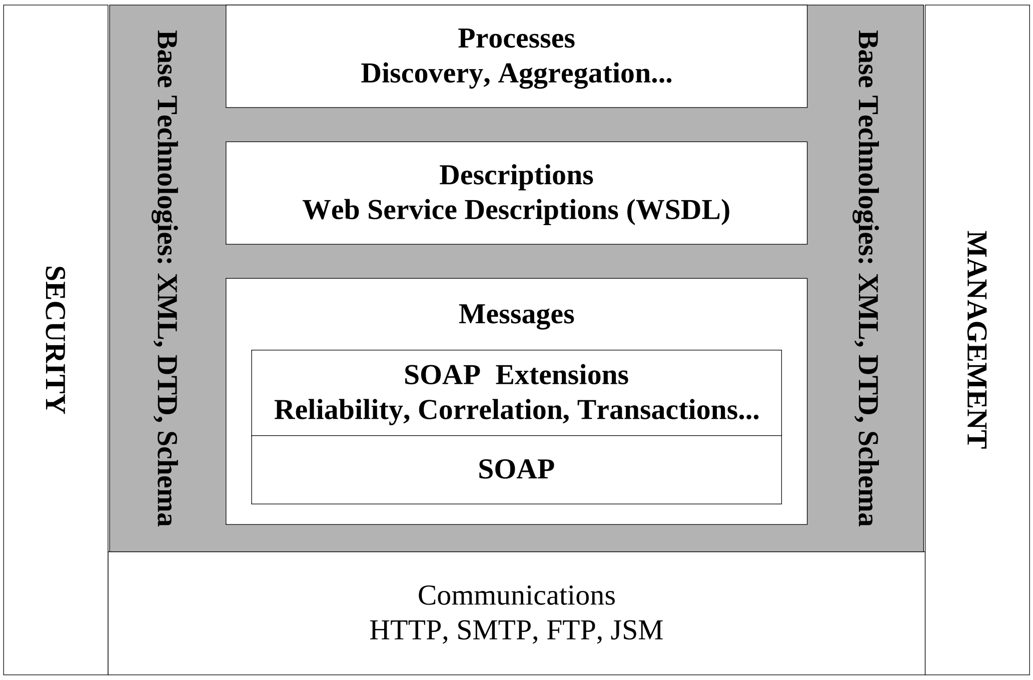

SECURITY
Base Technologies: XML, DTD, Schema
Processes
Discovery, Aggregation...
Descriptions
Web Service Descriptions (WSDL)
Messages
SOAP Extensions
Reliability, Correlation, Transactions...
SOAP
Base Technologies: XML, DTD, Schema
MANAGEMENT
Communications
HTTP, SMTP, FTP, JSM

**Figure 2.4: Web services diagram**

The term Web services is described by an XML that covers all the details necessary to interact with the services, including message formats, transport protocols and location. SOAP provides a means of messaging between a service provider and a service requestor. It is independent of the underlying transport protocol. SOAP can carry on HTTP, FTP and SMTP. WSDL is an XML-based language to describe Web services how to access them, it provides a formal framework to describe services in terms of protocols, servers, ports and operations that can be invoked. The specification provides a SOAP binding which is the most natural technology to use to implement Web Services. Universal Description, Discovery and Integration (UDDI), obviously, we can know that it provides the registry and search mechanism for Web services. In brief, WSDL describes the format SOAP messages, and UDDI serves as a discovery service for the WSDL descriptions.

The Grid emphasize that use of Web services does not only indicate the use of SOAP for all communications. If needed, alternative transports can be used, for example to achieve higher performance or to run over specialized network protocols.

In the Grid environment, the critical point how to combine the different heterogeneous resources. XML-based metadata is popular for solving the problem, so it has been used widely [1]. The XML document can not only help managing facilities, but also interchange between different databases in the data mining. It is a useful communicating media. By the integration of Globus Toolkit, Web Services, XML parser and data mining application, we can achieve e-commerce data mining in a the Grid environment.

# Chapter 3

# Decision Tree with Grid

## 3.1 Categories of Grid Roles

There are basically four types of Grid by their functions: computational grids, access grids, data grids and datacenter grids [3]. Skillicorn further categorized Grid into free grids, public grids, and virtual private grids [3]. According to the data mining application model and user equality perspective, we present Grid can have the following three models.

- Top-down Model
- Reciprocal type
- Broadcast type

### 3.1.1 Top-Down Model

Basically this model consists of computational grids and distributed database functions. The diagram is shown in Figure 3.1, it can make us know clearly about this model. It belongs to private and closed grids. The computer type is server-based and is suitable for global enterprises and research institutes, because their computer resources have been shared and exchanged frequently. This model restricts computer connections and, therefore, it has higher level of security. Also the level of data sharing is high, and is suitable for database sharing. It is an extended application of distributed database system. Task type is mostly top to down. The “top” task probably gives data and formula to the “down” task and uses the spare computer CPU time for the “down” task. Due to the effects of time elapse, the model can work 24 hours a day.

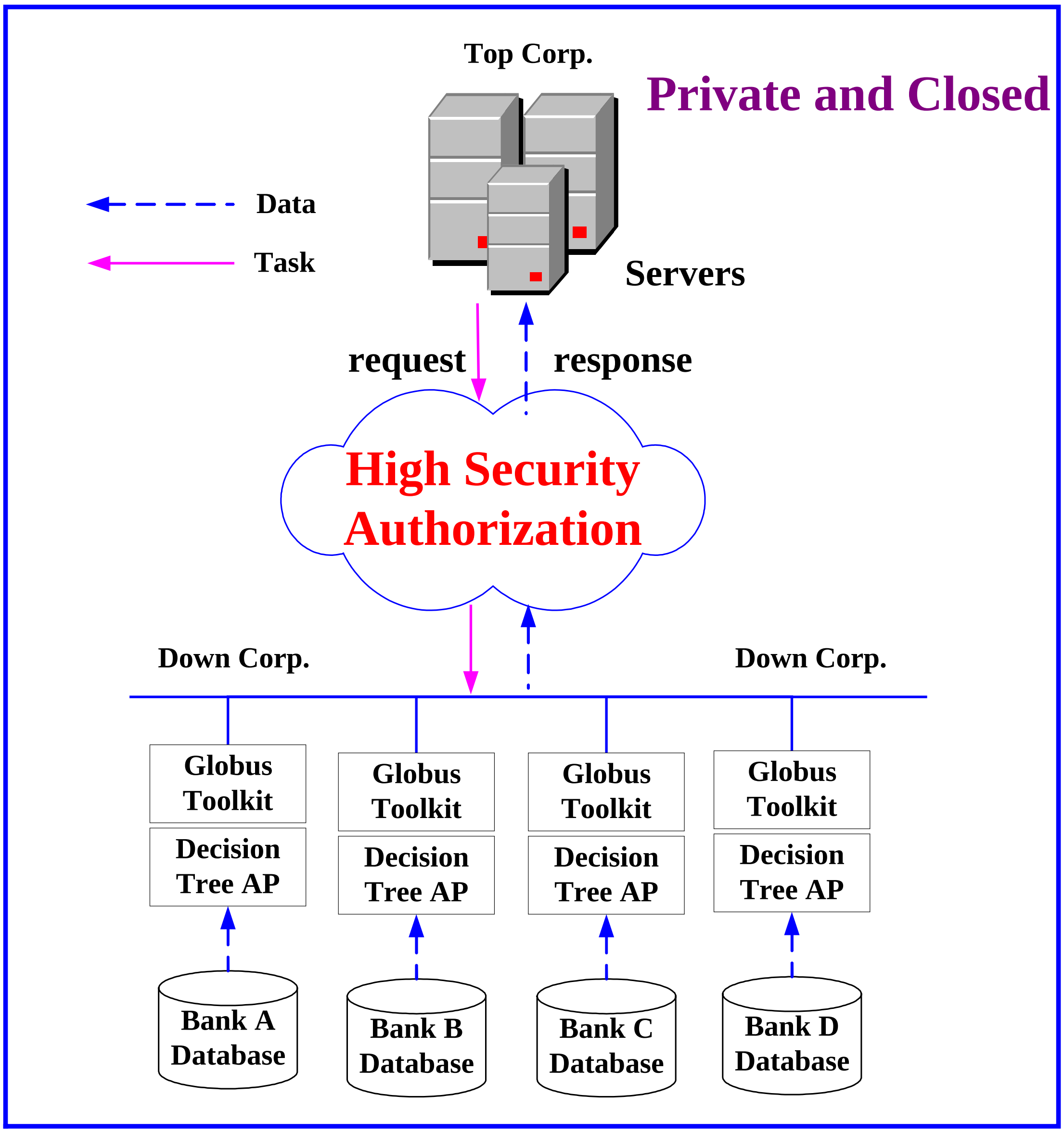


**Figure 3.1: Top-Down Model**

For example, the credit card company VISA wants to promote a new project. Before doing that, it needs to analyze the market situation. Because it is a company with a global view, it requires mining data of global consumer behavior. Traditionally, this would require the data collection from participated banks all over the world, but time and money spend would be tremendous. Also it will spend much time on calculation. It could be lost for market opportunities. With Grid, the company only needs to put the data mining application on the Web for these banks to download. Once installed, company can then gives instruction to the downloaded applications to mine the database located in various banks. These bank processes the task with idle CPU, especial at night. So maybe it can work 24 hours a day in the global. Finally, the company VISA collects the result from these banks and integrates those results. The major difference with the current service is that the local side shares not only data also computation are all distributed, the instructions are given by the master company VISA.

### 3.1.2 Reciprocal type

This model is like Peers-to-Peers model. It is an application of multiple-to-multiple network resources sharing. The diagram is shown in Figure 3.2. This model uses PCs and servers. With PCs, there is less restriction and most in public domains, the distributed tasks are small and less restrictive. However, with servers, most involving parties are from research institutes or other larger research units with higher restrictions on membership. They often share their databases with higher level of data security, same as the first model mentioned above. The task characteristics for this model are real-time, data variety and complex calculation. They, however, share their whole computer resources. Any node in the Grid can request and receive tasks.

The first wave of applications based on the Grid is from the aerospace and climate researches. It is obvious that Grid computing is very desirable for these two research fields, because the world climate and environment analysis is complex and interconnected to other research fields. Through the computer networks all over the world, i.e. the multiple to-multiple connections, we can understand the global climate and environment changes and inform the human being in the shortest time possible.

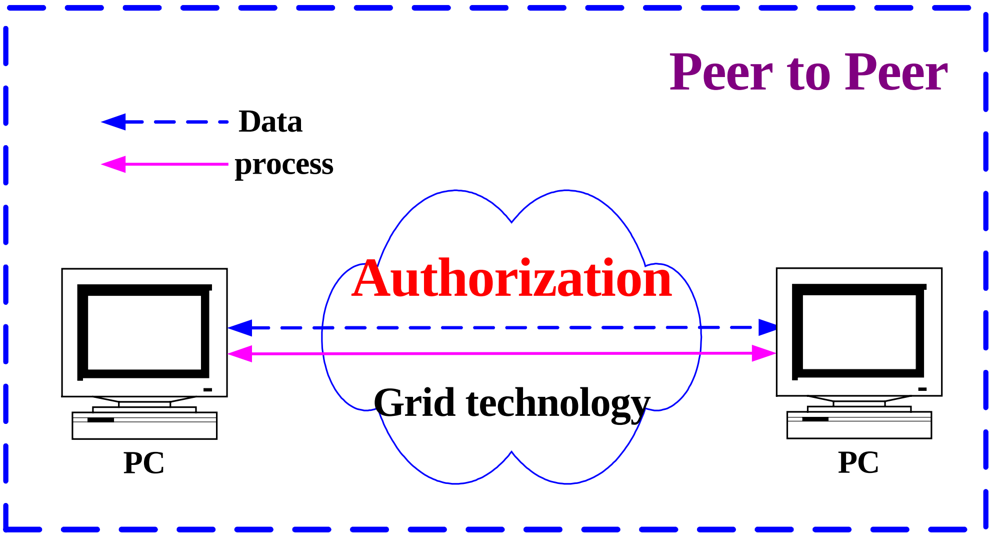


**Figure 3.2: Reciprocal Model**

## 3.1.3 Broadcast type

This type shown in Figure 3.3 is belongs to computation grids or free grids. It can also be called “project type network resource sharing”. It is basically a one-to-multiple, public, one-way communication type similar to the large scale internet Client/Server architecture. Usually this is a project developed by a research community, in order to do a mass and long term search on data. Basically, the community invites the enrollment of computers all over the world and then the interested parties download the required applications from the Web. After installation, the client computer becomes a sharable resource on internet. The model uses ordinary PCs or small servers. When the CPU resources are free on client computers, the applications then request tasks from the main server or the server dispatches tasks automatically to client computers. The client computers perform data mining and return the mining result. The data security for this model is poor. Therefore, the mining data should contain no sensitive data. The mining task types are ones that require large and complex calculation and can distribute data in small pieces.

For this model, the most famous example is Search for Extraterrestrial Intelligence at home (SETI@home). It is a non-profit organization. Everyone can download the applications from their public Website. How is works? The UC Berkeley SETI team has discovered that there are already thousands of computers that might be available for use. Most of these computers sit around most of the time with toasters flying across their screens accomplishing absolutely nothing and wasting electricity to boot. They will do this with a screen saver that can get a chunk of data from them over internet, analyze the data, and then report the results back to them. When you need your computer back, our screen saver instantly gets out of the way and only continues it is analysis when you are finished with your work [23]. Genome comparison calculation is another good example, besides SETI@home example.

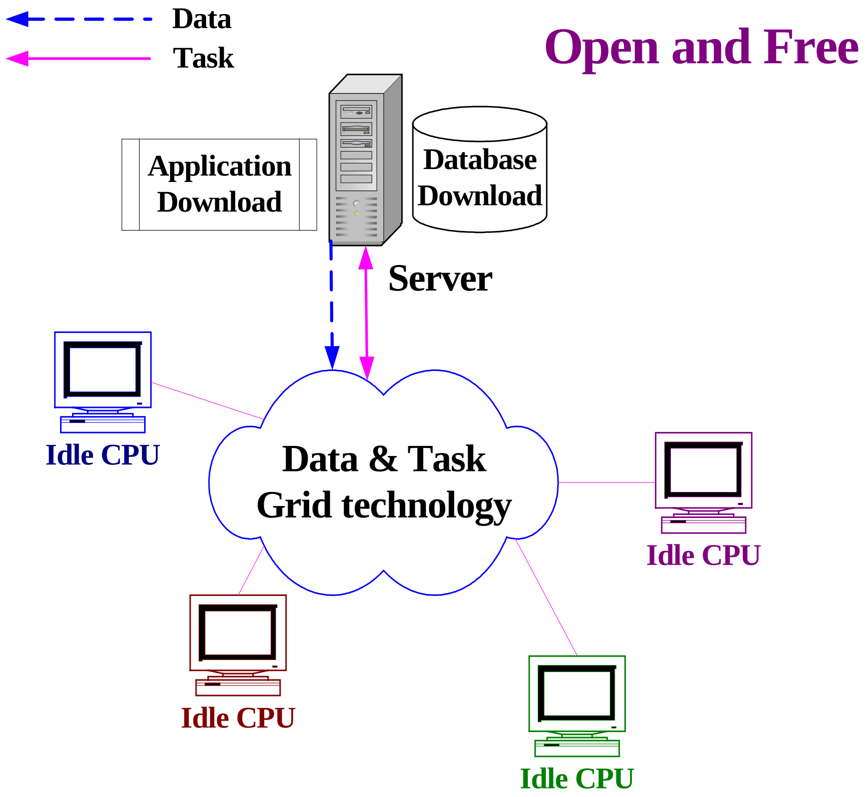


**Figure 3.3: Broadcast Model**

No matter which type is used, if the data security is a concern, the data should be encrypted when in transfer and calculation. From user point of view, this model only utilizes their computer resources without exposing the real data contents.

The purpose in we presenting above three models, we hope these models can help the researchers explore deeply in data mining by our clear definition. We believe if the roles of Gird define explicitly, process flow and data flow of data mining will distinguish their functions easily and help to develop Grid-based data mining model.

## 3.2 A Grid-based Decision Tree Architecture

Of course, we can set up a local Grid instead of global, because the basic idea is sharing information, not necessary depend on the scale. The purpose is to avoid wasting resource. The main advantage is in the resource integration. For example, through Grid computing technology we can find the idle computers on internet and divide task into these computers. Because of the efficient usage of resources, we can spend only one hour in finishing the tasks that originally needs ten hours to complete by one computer.

Since, Grid computing has caught a lot of attention. Some applications on Grid-based data mining infrastructure have been proposed, such as the Knowledge Grid (K-Grid) [15], NASA's Information Power Grid (IPG) [16]. Both are the input on the data mining algorithms work to extract new knowledge. The objective is to build the next generation computing infrastructure providing intensive analysis and integration over Word Wide Web. In addition, the DataGrid Project developed by European Union is focused on core Grid data services, like data access and metadata access. It can process the data sets from hundreds of TeraBytes to PetaBytes [17].

We present the "Top-Down" closed Grid architecture for decision tree, because it is normally used for e-commerce. Finding the best split point and performing the split are the main tasks in the decision tree. The former is a very important as well as very tedious job, specifically the continuous attribute, for which many algorithms use *gini* index [11] to get the split position. The split point is a mid-point between every two sorted consecutive attribute values. Therefore it requires ongoing calculation based on two adjacent values. When data amount is large, using Grid technology to split up the workload is very useful. So our main task in our Grid architecture is to calculate the split point.

In general, basic Grid technology services include security service, scheduling, data service, database service, user service, application management service, autonomy and monitoring service, information service, composition service and message service. Currently, Grid architecture is mature, and we will not elaborate the various service details here. Our architecture is shown in Figure 4, we just use Open Source Globus Toolkit as Grid middleware, it provides three functions, that is, resource management, data management and information services, they are built in GSI (Grid Security Infrastructure) [18]. The data exchange protocol, allocation, control and security in the Grid environment are all depends on this middleware. Web Service provides XML and HTTP services. It will be a corresponding suite of XML schema describing the object and services associated with the application [6]. The particular application is built on the top of them.

Users submit instructions to the decision tree application and through Globus find the suitable resources. Workload is estimated according to the resource efficiency. Following, attribute data and description will be specified in XML. Note the attribute data maybe have been sorted by task type. All the work is processed through Web Services. Those XML documents not only contain attribute data, but also describe the task including some parameters for computing split point. Finally, Globus sends the resulting documents to the client side.

In order to fully utilize the network resources, no matter using PCs or servers, there may be different application functions. But they all use the same Grid mechanism with the differences lie at the algorithm of parallel or sequential process. The client resource is processed as follow: through Web Services, XML files are transformed into database table data or text file by client attributes. If clients are using parallel mechanism, such as servers or Clusters, the software application will execute with parallel decision tree algorithms. On the other hand, if it is a PC, it will compute by the sequential algorithms.

The calculation results are then transformed into XML-based format which not include original attribute data. Because the decision tree application just needs the candidate split point values. The same process, send back to the local PC by Grid mechanism. The local PC receives the result, via Web Service, and then stores into the storage for data management. Through these repeating processes of resource search and allocation task, a large and various data mining will proceed faster than single resource process.

Through collaboration, it can extend the application scope of decision tree using Grid technologies. Parallel and sequential mechanisms both can be used. With the enhancement of calculation speed and Grid-related technologies, programmers do not need to modify too many codes to transfer the original system onto the Grid environment. Some algorithms are very useful and accurate, but the fact that during calculation all the tasks need to be fit into a single site is requiring further enhancement or replacement with the increasing number of data. Therefore, if data can be distributed to every computer in the world and participates in the Grid with each host processing part of the data and sending back response result. There is no need to modify the algorithm in order to reach the same objective with enhanced speed and performance and lower cost. Of course, the supercomputers or Clusters resources can also be included in the architecture.

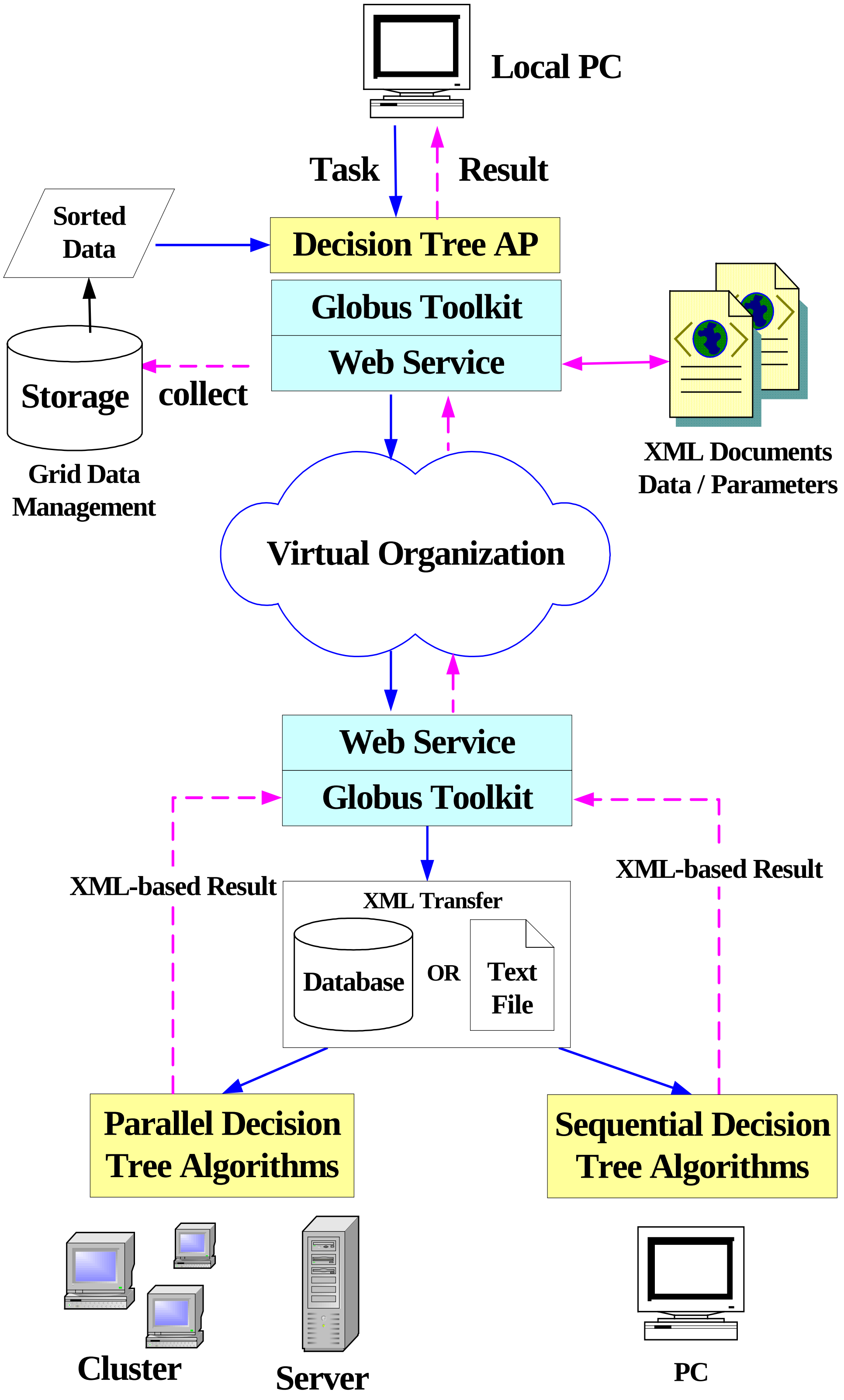


**Figure 3.4: Grid-based decision tree architecture**

# Chapter 4

# Experimental Result and Discussion

Currently, there are not many Grid experimental results for decision tree model in data mining research. First, in our experiment we just want to know how efficiency Grid in the different cases, and also how different variety between Cluster and Grid. Our using algorithm will partition the data to each client and computing it, not just only running the math formula in the memory. Our principal is that we try to run the same program between Cluster and Grid and do not modify the original program of Cluster. Of course, they need compiler in their own environment.

## 4.1 The Algorithm

We employ a parallel SPRINT (Scalable Parallel Induction of Decision Tree) algorithm that implemented its parallelization on an IBM SP2 [10]. We exploit the same program into Cluster and Grid environments, and try not to modify the originally program of Cluster. Also we want to know SPRINT algorithm whether suitable for Grid computing or not. Then, we introduce it briefly in next section and focus on the continuous attribute. Note this algorithm was running SMP architecture, but we run in Cluster architecture. So it can affect the experimental result, maybe different from the original paper of SPRINT.

### 4.1.1 SPRINT algorithm

SPRINT algorithm removes all of the memory restrictions, and it is fast and scalable. It is designed for being easily parallelized, allowing many processors to work together to build a single consistent model. The algorithm assumes a share-nothing parallel environment where each N processors has private memory and disk. The processors are connected by a communication network and can communicate only by passing messages [10]. SPRINT program [25] we downloaded that has been written by C++ and MPI, compiler with mpiCC v2.96. It is implementation of the SPRINT parallel synchronous decision tree construction.

For continuous attributes, there are two histograms are associated with each node for splitting. The example is shown in Figure 4.1 [10]. All attributes will partition to each own attribute list. The attribute list will average to every processor when process. $C_{below}$ maintains the attribute records that have already been processed, whereas $C_{above}$ for remainder that has not. Both $C_{below}$ and $C_{above}$ have all the necessary information to compute the *gini* index. For the root node, it is obtained at the time of sorting, the other nodes is obtained when the node is created. Since attribute lists are processed one at a time, memory is required for only one set of such histograms. It is one characteristic of SPRINT algorithm. Note this algorithm needs to sort the continuous attribute first before the put into the attribute list.

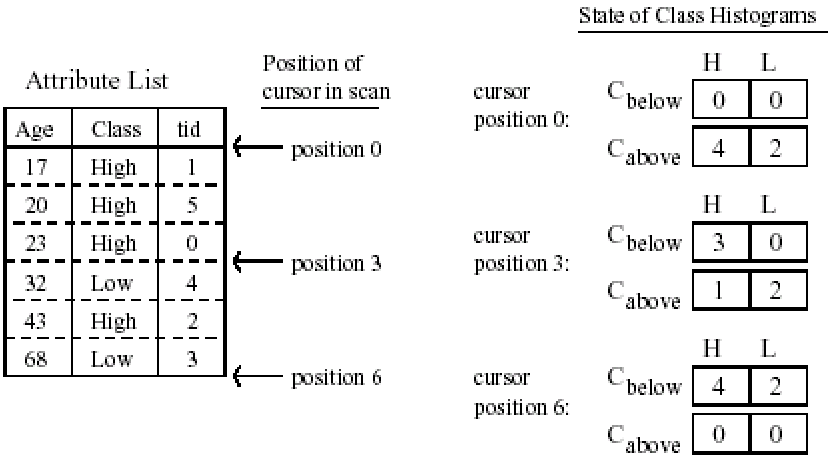


**Figure 4.1: Evaluating continuous split points**

For performing the split, it used a hash table that collected the attribute value, class and its record number, namely attribute list. You can refer to Figure 4.1. So there is nothing to which child the record was moved. The retrieved information told us with which child to place the record. Therefore, the memory is less used. If the hash-table is too large for memory, splitting is done in more than one step. The attribute list for the splitting attribute is partitioned according to the attribute record for which the hash table will fit in memory, and the process is repeated for the remainder.

### 4.1.2 Data placement and workload balancing

As data placement, the partitioning is achieved by first distributing the records equally among all the processors, it is shown in Figure 4.2. Each processor generates its own attribute list partitions in parallel. Also they can parallelize computing to find split point for each own record.

Processor 0

| Age | Class | rid |
|---|---|---|
| 17 | High | 1 |
| 20 | High | 5 |
| 23 | High | 0 |

| Car Type | Class | rid |
|---|---|---|
| Family | High | 0 |
| Sports | High | 1 |
| Sports | High | 2 |

Processor 1

| Age | Class | rid |
|---|---|---|
| 32 | Low | 4 |
| 43 | High | 2 |
| 68 | Low | 3 |

| Car Type | Class | rid |
|---|---|---|
| Family | Low | 3 |
| Truck | Low | 4 |
| Family | High | 5 |

**Figure: 4.2: Parallel Data Placement**

In a parallel environment, each processor can work independently for processing 1/N of the total data. Each processor has a separate contiguous section of a global attribute list. $C_{below}$ must initially reflect the class distribution of all sections of an attribute-list assigned to processors of lower rank. $C_{above}$ must likewise initially reflect the class distribution of the local section as well as all sections assigned to processors of higher rank. Once all the attribute list sections of a leaf have been processed, each processor will have what it considers to best the best split for that leaf. The processors then communicate to determine which of the N split points has the lowest cost.

# 4.2 Experimnetal Enviroment

## 4.2.1 Datasets

The dataset we use synthetic datasets with text file. We focus on the continuous attribute computing, we mentioned the cause in the former chapter. So there are not categorical attributes in the datasets. We try two different datasets: cov4 and big4, big4 is almost double size of cov4. Both have 4 continuous attributes and 1 class, record amount separate about 581K and 1M, is shown in Table 4.1. In these synthetic datasets, the value distribution is, the class is 1,2,3,4,5,6,7, attribute 1 range 200 to 3999, attribute 2 range 1 to 500, attribute 3 is 1 to 99, attribute 4 is 0 to 99, and the interval between attribute value is various.

**Table 4.1: The dataset list**

| Dataset | Attributes | | | Record | Size |
|---|---|---|---|---|---|
| | Continuous | Categorical | Class | | |
| cov4 | 4 | 0 | 1 | 581,012 | 15.7MB |
| big4 | 4 | 0 | 1 | 1,162,024 | 31.7MB |

### 4.2.2 Cluster

Our Cluster C149 is a low cost Beowulf-type supercomputer that utilizes multi-computer architecture for parallel computations, as shown in Table4.2. It consists of 8 PC-based connected by three 24-port 100Mbps Ethernet switches with Fast Ethernet interface. Each node has two AMD Athlon MP 2600+ or 2400+ 1GHz processors. There are total 16 processors. The disks are made a NFS shared local disk. The processors of Cluster connected with a private network. It is a less heterogeneous Cluster architecture.

All computers in this Cluster run the RedHat Linux release 8 operating system. Program is developed using C++ language, compiler within MPI library LAM 6.5.8-4 and gcc 3.2.2. The program and dataset are just put in the master node, also compiler and run here.

**Table 4.2: Hardware configuration of Cluster C149**

| Cluster C149 | | | |
|---|---|---|---|
| No | Host Name | Processor (CPU=16) | Memory |
| 1 | Amd1* | AMD Athlon MP 2600+ × 2 | 2048MB |
| 2 | Amd2 | AMD Athlon MP 2600+ × 2 | 1024MB |
| 3 | Amd3 | AMD Athlon MP 2600+ × 2 | 1024MB |
| 4 | Amd4 | AMD Athlon MP 2400+ × 2 | 1024MB |
| 5 | Amd5 | AMD Athlon MP 2400+ × 2 | 1024MB |
| 6 | Amd6 | AMD Athlon MP 2400+ × 2 | 1024MB |
| 7 | Amd7 | AMD Athlon MP 2400+ × 2 | 1024MB |
| 8 | Amd8 | AMD Athlon MP 2400+ × 2 | 1024MB |

* stand for Master node of the cluster, the others is slave node.

### 4.2.3 Grid

Our Grid architecture is implemented on top of Globus Toolkit, name grid-cluster. It is built three Clusters to form a multiple cluster environment. Cluster1 and Cluster2 consist of 4 PCs that each has one master node and three slave nodes. Cluster3 consists of 3 PCs, has one master node and two slave nodes, as shown in Table 4.3. Each node is interconnected through 3COM 10/100 Fast Ethernet Card to Accton Switch HUB. SGE QMaster daemon is run on the master node of each Cluster, and SGE execute daemon is run to manage and monitor incoming job and Globus Toolkit v2.4. Each slave node is running SGE execute daemon to execute income job only.

The operating system is RedHat Linux release 9. Parallel application we use MPICH-G2 v1.25 for message passing. The program and dataset have to be put in the master node of each Cluster and also need to be put into the same directory. But we can run just in Cluster1. Cluster1 is a master of grid-cluster. If we use mpirun over 5 processors, it means that the process will communicate current Cluster with another Cluster. The order of mpirun is Cluster1, Cluster2 and Cluster3. Note the order will affect the result of run. During execution, if mpirun needs to cross the other processor, it will run sequentially by machine file that we have written.

**Table 4.3: Hardware configuration of Grid-cluster**

| Cluster1* | | | | |
|---|---|---|---|---|
| No | IP | Host Name | Processor (CPU=4) | Memory |
| 1 | beta1.hpc.csie.thu.edu.tw | beta1* | Intel Celeron 1.7GHz | 512MB |
| 2 | beta2.hpc.csie.thu.edu.tw | beta2 | Intel Celeron 1.7GHz | 256MB |
| 3 | beta3.hpc.csie.thu.edu.tw | beta3 | Intel Celeron 1.7GHz | 256MB |
| 4 | beta4.hpc.csie.thu.edu.tw | Beta4 | Intel Celeron 1.7GHz | 256MB |

| Cluster2 | | | | |
|---|---|---|---|---|
| No | IP | Host Name | Processor (CPU=8) | Memory |
| 1 | gamma1.hpc.csie.thu.edu.tw | gamma1* | Pentium III × 2 | 512MB |
| 2 | gamma2.hpc.csie.thu.edu.tw | gamma2 | Pentium III × 2 | 512MB |
| 3 | gamma3.hpc.csie.thu.edu.tw | gamma3 | Pentium III × 2 | 512MB |
| 4 | gamma4.hpc.csie.thu.edu.tw | gmma4 | Pentium III × 2 | 512MB |

| Cluster3 | | | | |
|---|---|---|---|---|
| No | IP | Host Name | Processor (CPU=6) | Memory |
| 1 | alpha1.hpc.csie.thu.edu.tw | alpha1* | AMD Athlon MP 2400+ × 2 | 1024MB |
| 2 | alpha2.hpc.csie.thu.edu.tw | alpha2 | AMD Athlon MP 2000+ × 2 | 1024MB |
| 3 | alpha3.hpc.csie.thu.edu.tw | alpha3 | AMD Athlon MP 1800+ × 2 | 512MB |

* stand for Master node of the cluster, the others is slave node.

## 4.3 Performance Result

There are two kinds of environment and two datasets in our experiment. We first examined the original cluster C149, the result is shown in Table 4.4 and diagram of curve is shown in Figure 4.3. The turn-around time is the total response time. We found the best performance is 6 processors, because the former 3 CPU of nodes are higher. The latter 5 nodes are lower, so the performance is getting down. There is just a little difference in CPU, but the performance is sensitive. Although the result is not good for increasing processors, it needs more communication as processor is increased. Fortunately, the execution time is not double when the amount of dataset is double. In the experiment of original SPRINT paper is implemented in IBM SP2, it is all in the SMP architecture with intra communication. Our Cluster has external communication with each node.

**Table 4.4: The turn around time of C149 list**

| Dataset | 1 | 2 | 4 | 6 | 8 | 12 | 14 | 16 |
|---|---|---|---|---|---|---|---|---|
| cov4 | 51.355s | 40.998s | 28.343s | 27.613s | 29.294s | 39.494 | 43.671 | 45.229s |
| big4 | 1m48.368 | 1m19.489 | 59.458s | 53.6948s | 55.204s | 1m6.699s | 1m13.178s | 1m19.374s |

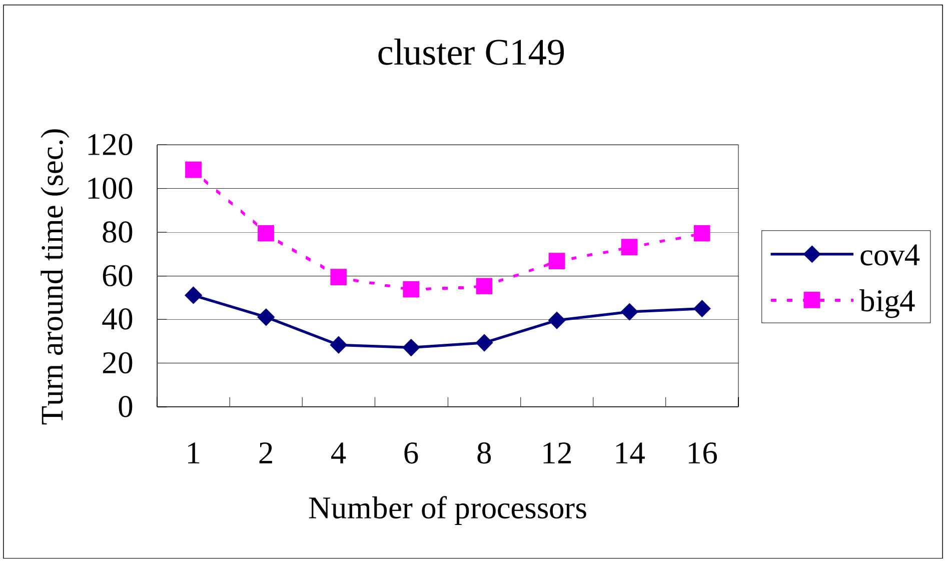


**Figure 4.3: The turn-around time of C149 diagram**

Table 4.5 and Figure 4.4 show our experimental result of Grid environment. We are satisfied that we successfully transplant original program into Grid just in need of re-compiler. During the experiment test, for the same case we got the different turn-around time every time, but the interval within one minute. We try to do our best to get the result in the same condition. It is still changeable. It is comprehensible because the interchange communication is unsettled in the internet. Therefore, we choose the least time of every processor for this experimental presentation. So the results of 6 processors and 12 processors are higher suddenly, because they just cross the external node. After that, the workload is average cause the speedup is getting increasing. We get the amazing discovery that we can get the lower turn-around time for the maximum processor. As is seen from the figure, all the execution times are very close after 4 processors, but big4 dataset is double than cov4 dataset. All prove that Grid computing is available in the particular condition. It means we can have a good scaleable for large dataset.

**Table 4.5: The turn-around time of grid-cluster list**

| Dataset | 1 | 2 | 4 | 6 | 8 | 12 | 16 |
|---|---|---|---|---|---|---|---|
| cov4 | 2m44.869s | 3m1.86s | 3m1.378s | 3m5.732s | 2m57.125s | 2m58.220s | 3m8.833s |
| big4 | 10m26.781s | 5m34.492s | 5m25.467s | 4m4.191s | 2m46.032s | 3m53.775s | 3m57.813s |

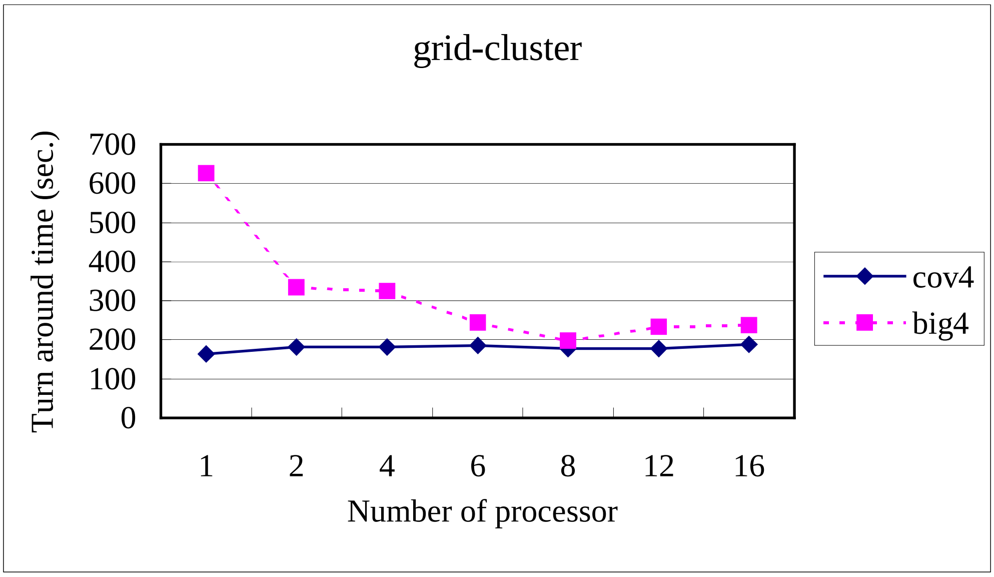


**Figure 4.4: The turn-around time of grid-cluster diagram**

Next, we compare Cluster with Grid. We know their hardware are different, we just put their results together and compare their curve varieties. Furthermore, their CPU gap is not too much. The data value as the same former experiment, are shown in Table 4.6 and Table 4.7, and the diagrams are shown in Figure 4.5 and Figure 4.6. Obviously, for the speedup, Cluster is much better than Grid, even just one processor. We believe that the reason is due to the different MPI library, MPICH-G2 has a more communication layer with Globus toolkit. But in other hand, the increasing processor is good for Grid. As seen is the figure, the execution time is just a little decreasing when the processor is increasing in the Grid, and the Cluster is not. We can see that the performances of large dataset are near gradually. Particularly, processor is increasing.

**Table 4.6: Cluster and Grid comparison for cov4 dataset**

| Type | 1 | 2 | 4 | 6 | 8 | 12 | 16 |
|---|---|---|---|---|---|---|---|
| C149 | 51.355s | 40.998s | 28.343s | 27.168s | 29.294s | 39.494 | 45.229s |
| grid-cluster | 2m44.869s | 3m1.86s | 3m1.378s | 3m5.732s | 2m57.125s | 2m58.220s | 3m8.833s |

**Table 4.7: Cluster and Grid comparison for big4 dataset**

| Type | 1 | 2 | 4 | 6 | 8 | 12 | 16 |
|---|---|---|---|---|---|---|---|
| C149 | 1m48.368 | 1m19.489 | 59.458s | 53.694s | 55.204s | 1m6.699s | 1m19.374s |
| grid-cluster | 10m26.781s | 5m34.492s | 5m25.467 | 4m4.191s | 3m18.227s | 3m53.775s | 3m57.813s |

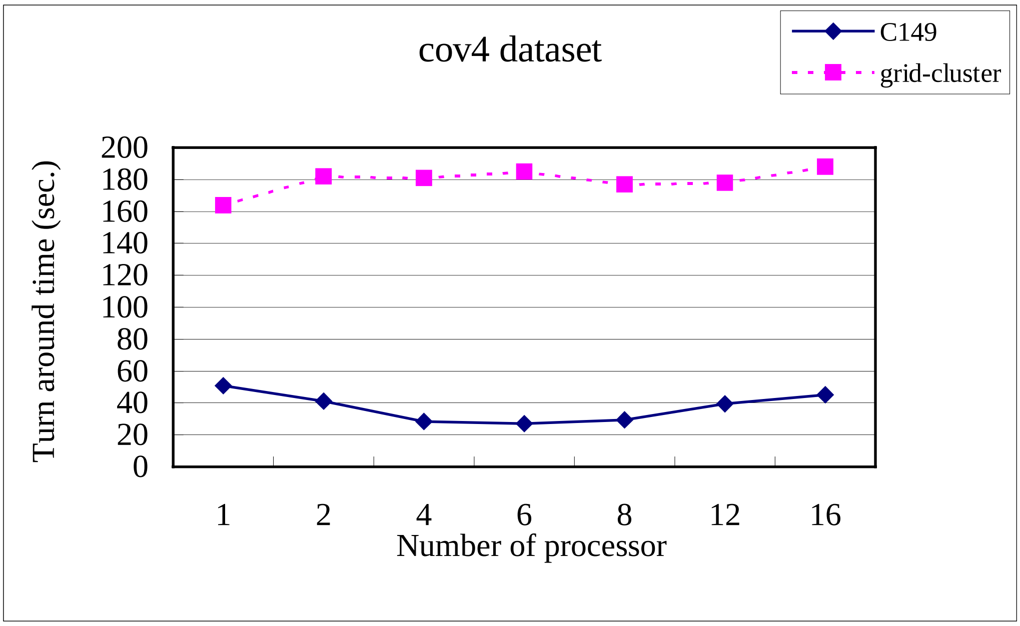


**Figure 4.5: Cluster and Grid comparison diagram for cov4 dataset**

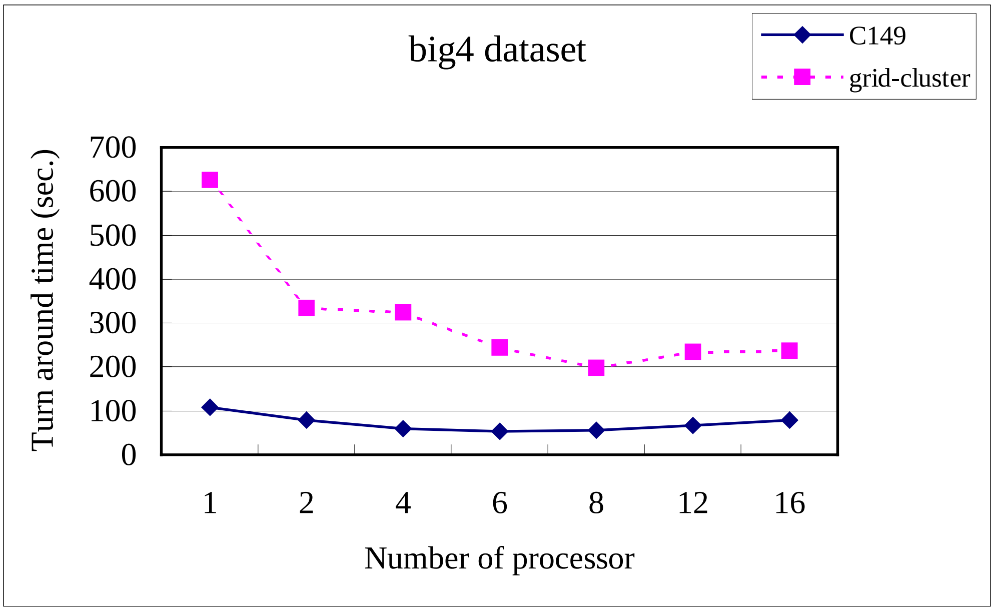


**Figure 4.6: Cluster and Grid comparison diagram for big4 dataset**

## 4.4 Experiment Discussion

Obviously, for the speedup, Cluster is much better than Grid in our case. In the other hand, we do not schedule the workload into the heterogeneous configuration. It also affects the entire performance. We believe that because more processors more exchange between slave and master, but if the dataset is bigger and processor is more, then calculation time of each processor will increase, also the workload will be partitioned into less, the parallel efficiency will instead of exchange. Frankly, we have bottlenecks for more attributes and larger dataset in the Grid environment. The dataset is too large to Grid segmentation. We plan use larger synthetic dataset in next experiment and try the other way to solve it, even to modify the current program or try another parallel decision tree algorithms. After all, we demonstrate that Grid is not bad in the multiple processors.

Our experiment reveals the other appearance that the speedup will be a little down when the processor first time touches to the other Cluster processor in the Grid.

But it speeds up later. Of course it will affect the Grid type that consists of plenty single workstations. It is worth notice.

The turn-around time of multiple processor does not double increase when the dataset is double, oppositely, the speedup is closed. It is also amazing and satisfying. All the experimental results can make us get going to research Grid computing for data mining.

# Chapter 5

# Related Work

Data mining still has problems to the scientific community. Among them, issues related to the performance of such algorithm still limit their ability to affectively hand really hung dataset. Possibly distributed across several sites and with variable. And our next work is to handle more attributes and hung dataset. At that moment, we need a tool to manage distributed data. So we also study the EU Data Grid Project and expect the decision tree application to associate it. The important point is that built on top of Globus Toolkit as the same with our current Grid environment.

The EU Data Grid Project developed by European Union is focused on core Grid data services, like data access and metadata access. The objective is to build the next generation computing infrastructure providing in intensive computation and analysis of shared large-scale databases. It can process the data sets from hundreds of TeraBytes to PetaBytes [17]. Data Grid services are built on top of Globus and simplify the task of managing computations that access distributed and large data sources. Data Grid environment consists of two important elements: GRIS (Grid Resource information Services) and MDS (Metacomputing Directory Services). GRIS provides an infrastructure for storing and managing the status and components around the Whole Grid. MDS is based on LDAP (Lightweight Directory Access Protocol), which is used for storing and looking up data related to information on Grid [17]. The architecture is shown in Figure 5.1.

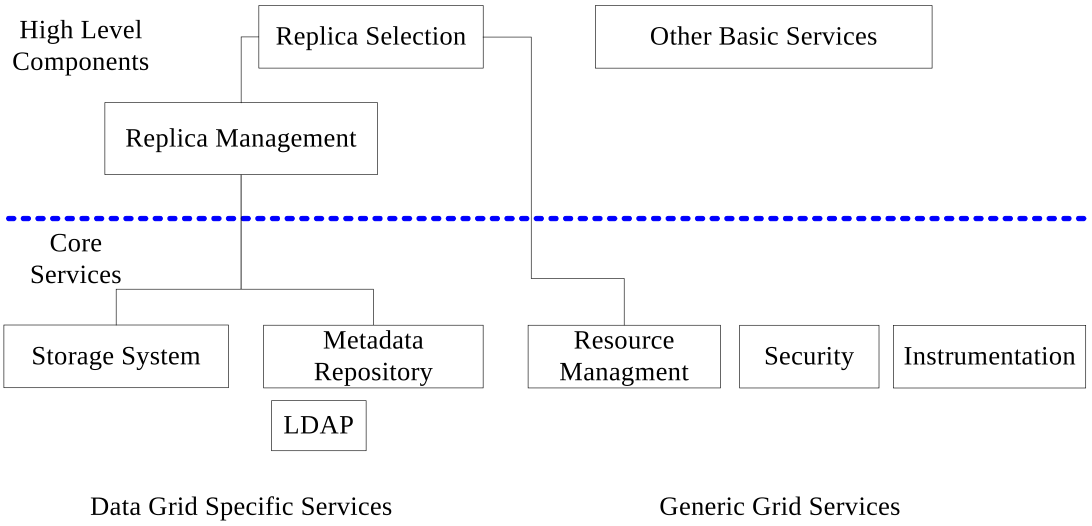


**Figure 5.1: The Data Grid architecture**

The final work, we expect to implement the Data Grid into our Grid environment, and integrate the decision tree algorithm and large-scale database. To speedup and efficient in large dataset is the ultimate goal for data mining. Of course, we know that the other searchers have proposed similar system already, such as Knowledge Grid [24]. But our focus on decision tree application and both parallel and sequential programs can be used in the Grid environment. In addition, the roles of Grid that we have present can make the definition task. We believe there is some difference with the others.

# Chapter 6

# Conclusion

The main goals of this thesis are providing the idea architecture for decision tree model in the Grid environment, and analyzing Grid roles of user equality perspective in the data mining application. We hope the Grid-based decision tree architecture plus the roles of Grid can help the emerging Grid community. Also expecting evaluation report of the experimental performance in different platform is interesting to the computer researchers.

This preliminary experimental is the first step to achieve our Grid-based decision tree architecture. After all, it is a big plan, we need to know what arduous problem we will meet, and step by step achieve it. We think if the transplant is easy about Cluster into Grid, then we can smoothly go to the next work that implements the decision tree application. This ideal application can assign separately the task to single PC and parallel computer according to the response of Globus Toolkit. Further, we will involve the different data media, database or text file base on computer side needs.

In this thesis, we just want to know how is the efficiency between Cluster and Grid and try not to modify the original program. Our preliminary experimental results show that Cluster is better than Grid in our datasets, but increasing processor advantage Grid. Therefore, our next work is to implement the large dataset and more attributes in Grid, and maybe try to use another decision tree algorithm. Finally, we will have an application that can manage and assigns the work to remote workstation to achieve Grid-based decision tree architecture that we propose.

Our experiment reveals Grid computing is expansibility, we believe it is expectable that Grid computing will be a trend and common. Now, there are more and more searchers and large industries invest much time and money in Grid investigates.

We will have more tools to assist in achieving our goal that is integrated Grid-based decision tree application.

We hope, through Grid collaboration, it can make data mining tasks originally thought too difficult or complex to become possible. And all can be implemented in every different research field, despite the fact that current Grid-based data mining applications still have many problems and limit to different scientific communities. But from the perspective of the network development and global common objectives, Grid is an ideal of human being and a feasible platform.

# References

[1] C. Mastroianni, D. Talia, P. Trunfio: Managing Heterogeneous Resources in Data Mining Applications on Grids Using XML-Based Metadata, “Proceedings IPDPS” 2003,” IEEE Computer Society Press”, April 2003.

[2] David B. Skillicorn, Parallel Data Mining:
http://www.cs.queensu.ca/home/skill/cascon.html

[3] David B. Skillicorn, Motivating Computational Grids: In Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems at “CCGrid” 2002.

[4] Domenico Talia: High-Performance Data Mining and Knowledge Discovery, “Euro-Par”, Paderborn, Germany, August 2002.

[5] G. Narlikar, A Parallel, Multithreaded Decision Tree Builder: “Tech. Report” CMU-CS-98-184, December 1998.

[6] Geoffrey Fox, Education and the enterprise with the Grid, Grid Computing: Making the Global Infrastructure a Reality, F. Berman, G. Fox, and T. Hey eds. John Wiley & Sons, 2003.

[7] Gerald Benoit, DATA MINING, Annual Review of Information Science and Technology ,2002, B. Cronin, ed., in press.

[8] I. Foster, C. Kesselman, and S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, “Intl. J. Supercomputer Applications”, 3(15), 2001.

[9] J. Gehrke, R. Ramakrishnan, and V. Ganti: Rainforest – A framework for fast decision tree construction of large datasets. “VLDB” 1996.

[10] J. Shafer, R. Agrawal, and M. Mehta: SPRINT: A scalable parallel classifier for data mining, In “Proc. of VLDB”, 1996.

[11] J. Hagiwara, T. Doi, T. Shindo, Y. Yaginuma, K. Maeda: Commercial applications on the AP3000 Parallel Computer, “IEEE Massively Parallel Programming Models '97”, 1997.

[12] M. J. Zaki, C.-T. Ho, and R. Agrawal: Parallel classification for data mining on Shared-Memory multiprocessors, “IEEE International Conference on Data Engineering”, May 1999, pages 198--205.

[13] Paolo Palmerini: High Performance Distributed Systems for Data Mining, November 19, 2002

[14] Rahul Ramachandran, Helen Conover, Sara Graves, Ken Keiser, Sunil Movva, and Steve Tanner: Flexible Earth Science Data Mining System Architecture, “AHPCRC”, 2001.

[15] S. Orlando, P. Palmerini, R. Perego, F. Silverstri: Scheduling High Performance Data Mining Tasks on a Data Grid Environment, “proceedings of Europar”, 2002.

[16] Thomas H. Hinke, Jason Novotny: Data Mining on NASA's Information Power Grid, “HPDC”, 2000.

[17] W. Allcock, A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke: The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, “Journal of Network and Computer Applications”, 2001,23:187–200.

[18] Foster, I., Kesselman, C., Nick, J. and Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, “Globus Project”, 2002.

[19] Alsabti K., Ranka S. and Singh V.: CLOUDS: A Decision Tree Classifier for Large Datasets, in “Proc. KDD ‘98”, 4th Intl. Conf. on Knowledge Discovery and Data Mining, New York City, 1998, pp. 2-8.

[20] Mohammed Javeed Zaki, Ching-Tien Ho, Rakesh Agrawal: Parallel Classification for Data Mining on Shared-Memory Multiprocessors. “ICDE 1999”: 198-205.

[21] I. Foster, I., Kesselman, C., Nick, J. and Tuecke, S:. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, “Globus Project”, 2002.

[22] Web Services Architecture, W3C Working Draft 8 August 2003, http://www.w3.org/TR/ws-arch/

[23] How SETI@home works, http://setiathome.ssl.berkeley.edu/about_seti/about_seti_at_home_1.html

[24] S. Orlando, P. Palmerini, R. Perego and F. Silverstri: Scheduling High Performance Data Mining Tasks on a Data Grid Enviroment, “proceedings of Europar”, 2002.

[25] http://www-personal.monash.edu.au/~dtaniar/VPAC/parsprint.zip

[26] http://www.cs.rpi.edu/~zaki/datamining.html, SPIDER.

[27] http://www-306.ibm.com/software/data/iminer/fordata/, IBM DB Intelligent Miner for Data.

# Appendix

I will describe my experimental process here for someone who interested it. The detail description is very hard to be written in English, but I do my best. So forgive me if they are difficult to understand or they are not enough detail.

## The program how to run in Cluster environment

IP: 140.128.101.149
Program and data put in /home/ct/sharon/sprint
Reference file: readme.txt , readme-2.txt

**Compiler:**

```
mpiCC sprint.cpp chtbl.cpp support.cpp treenode.cpp list.cpp showTime.cpp -o sprint
```

also you can run execute a batch file that I have written

```
$./cc.bat
```

then will produce a execution file, sprint

Note: compiler will produce warning message, just leave it. Cause the original program was compiler in C++ v2.96, but our compiler source is C++ v.3.2. Fortunately, it do not affect the execution file run.

**Execution steps:**

**1. Boot mpi**

```
cd /home/ct/sharon
lamboot –v lamhost
```

If you want to kill mpi, run wipe –v lamhost

**2. Run**

```
cd /home/ct/sharon/sprint
mpirun –np nodesize sprint datafile record number
```

Simple Example:

```
mpirun –np 4 sprint wheater 14
```

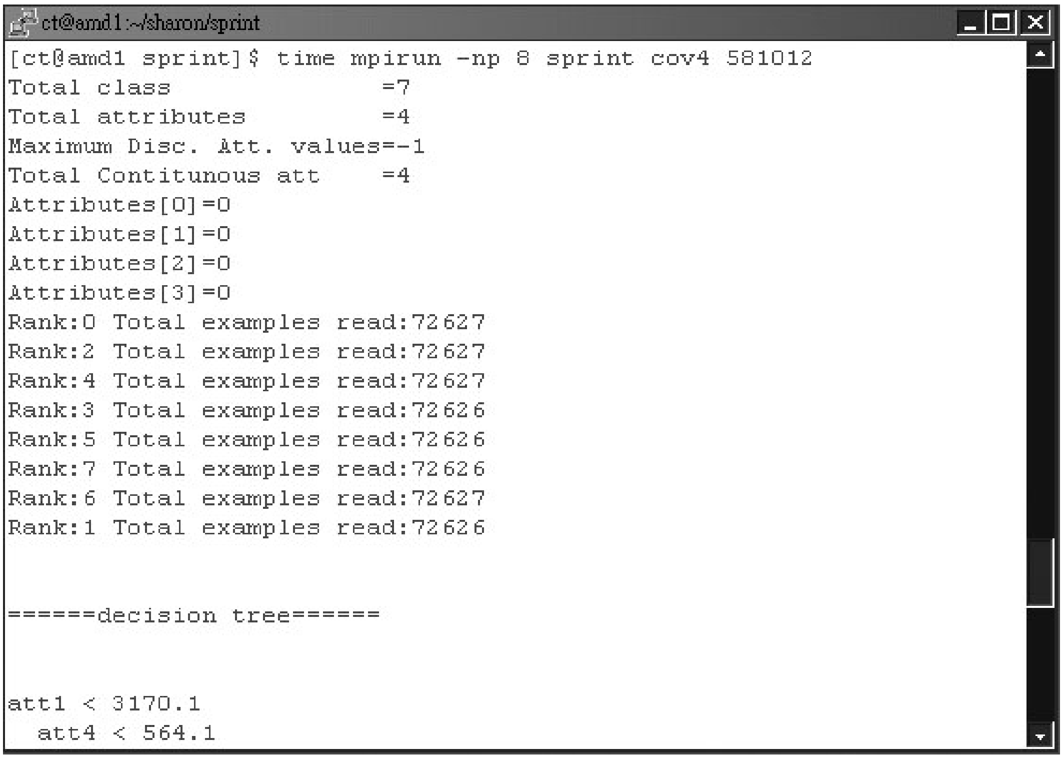
If you want to show the execution time, just add “time” in front of mpirun.

There are test data in this directory, as below cov4 (4 attributes 581012)

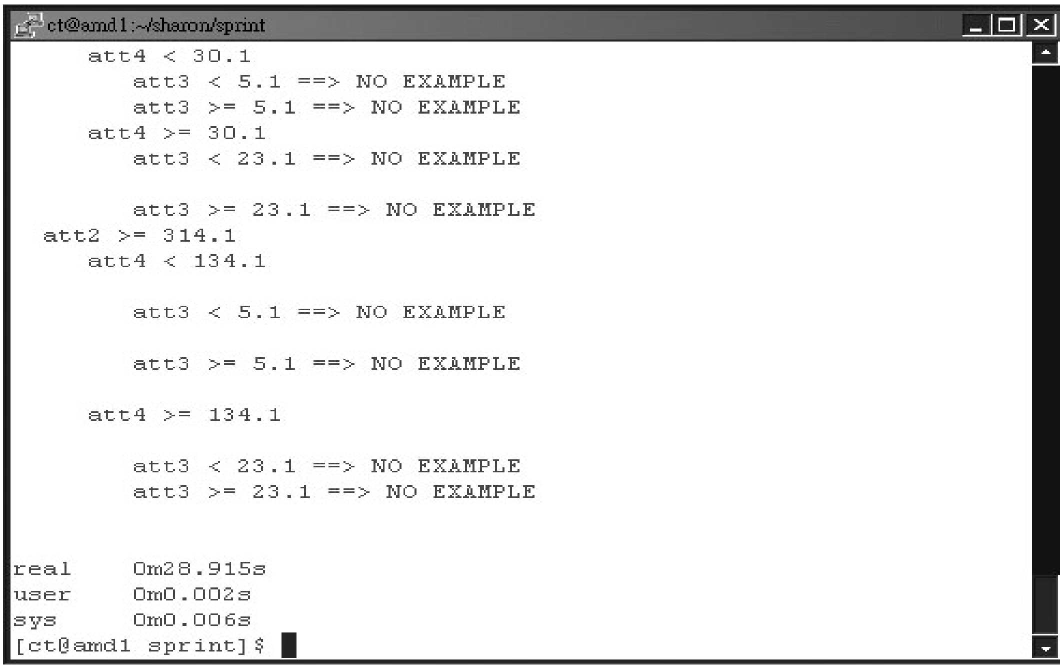
cov6 (6 attributes 581012)

big4 (4 attributes 1162024) , big4 is double cov4, just combine the same two cov4s

## 3. Result



```
[ct@amd1 sprint]$ time mpirun -np 8 sprint cov4 581012
Total class              =7
Total attributes         =4
Maximum Disc. Att. values=-1
Total Contitunous att    =4
Attributes[0]=0
Attributes[1]=0
Attributes[2]=0
Attributes[3]=0
Rank:0 Total examples read:72627
Rank:2 Total examples read:72627
Rank:4 Total examples read:72627
Rank:3 Total examples read:72626
Rank:5 Total examples read:72626
Rank:7 Total examples read:72626
Rank:6 Total examples read:72627
Rank:1 Total examples read:72626


======decision tree======


att1 < 3170.1
  att4 < 564.1
```



```
     att4 < 30.1
        att3 < 5.1 ==> NO EXAMPLE
        att3 >= 5.1 ==> NO EXAMPLE
     att4 >= 30.1
        att3 < 23.1 ==> NO EXAMPLE

        att3 >= 23.1 ==> NO EXAMPLE
  att2 >= 314.1
     att4 < 134.1

        att3 < 5.1 ==> NO EXAMPLE

        att3 >= 5.1 ==> NO EXAMPLE

     att4 >= 134.1

        att3 < 23.1 ==> NO EXAMPLE
        att3 >= 23.1 ==> NO EXAMPLE


real    0m28.915s
user    0m0.002s
sys     0m0.006s
[ct@amd1 sprint]$
```

**Note:**

There are showing some messages during the run time. Just do not care. They all show out the program has used the other rank of Cluster. The final result show the decision tree, it mean it is ok. For the decision tree, because the data is no meaning, attribute values are random, cause some attributes belong to no example.

## The program how to run in the Grid environment

IP: beta1.hpc.csie.thu.edu.tw
Program and data put in /home/ct/sprint

**Compiler**:

There are different from Cluster, the compiler source is not current setup, so put in our directory tony. And compiler will not product warning message.

```
./tony/mpich/bin/mpicc -I/home/cll/sprint/tony/mpich/include -include
/home/cll/sprint/tony/mpich/include/mpi.h  -Wno-deprecated sprint.cpp chtbl.cpp
support.cpp treenode.cpp list.cpp showTime.cpp  -o sprint
```

also you can run execute a batch file that I have written

```
$./cc.bat
```

then will produce a execution file, sprint

The Grid connects by three Clusters, beta1, gamma1, alpha1. So the execution file (sprint) and data need copy to each master node of Cluster. But I can just compiler in beta1, and then copy to everyone.

**Execution steps:**

**1. Boot Grid**

I do not boot mpi in Grid, but I need boot grid-proxy-init if it is not booted or invalid.

When you run grid-proxy-init, it will ask you identity, then you just key in.

**2. Run**

The run process is total the same as Cluster, also the data is the same.

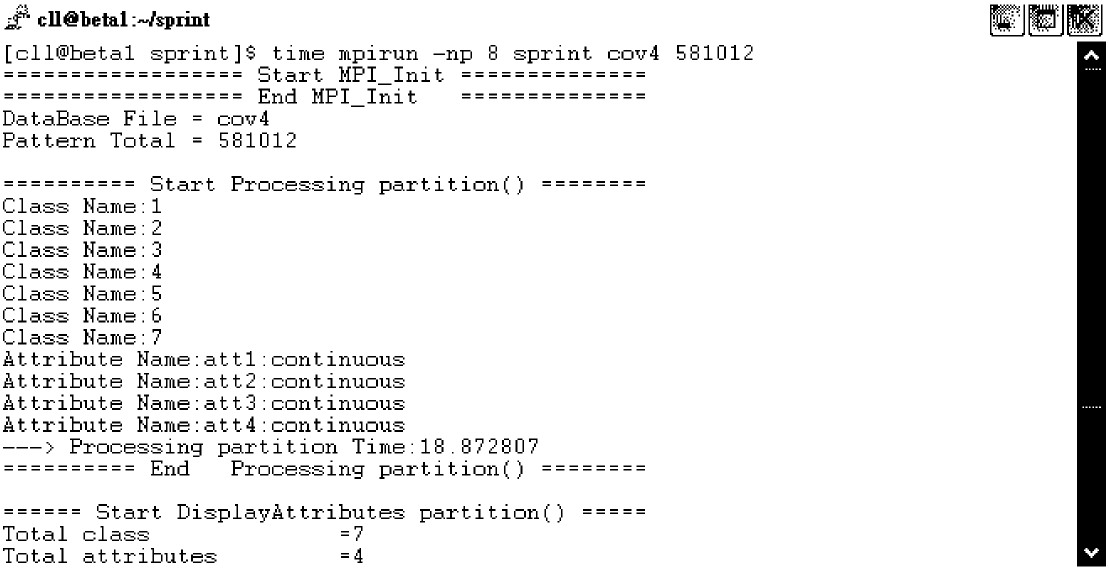
```
cd /home/ct/sprint
mpirun –np nodesize sprint datafile record number
```

Simple Example:

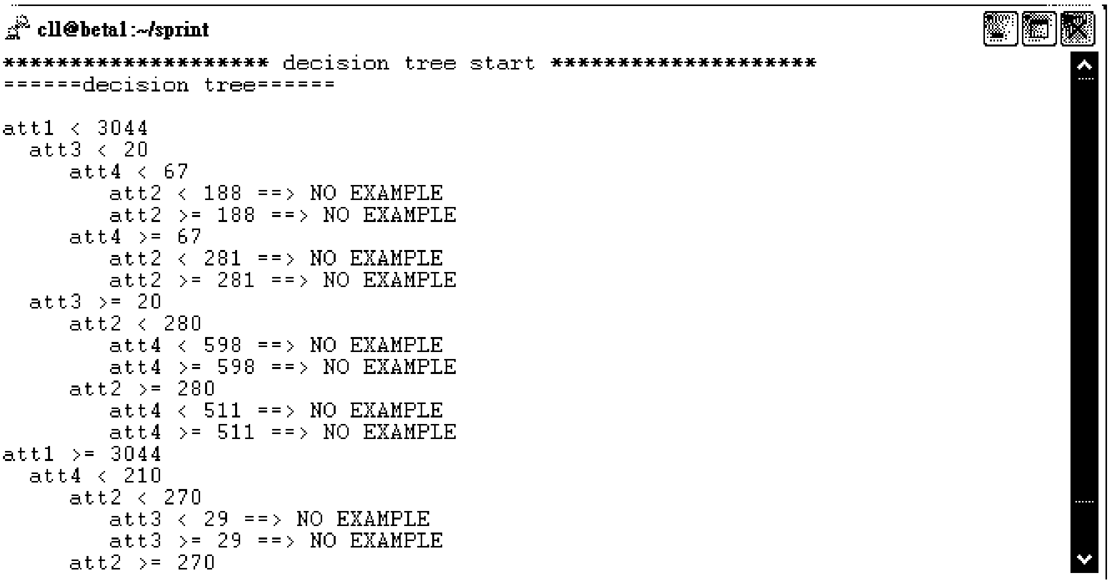
mpirun –np 4 sprint wheater 14

If you want to show the execution time, just add “time” in front of mpirun.

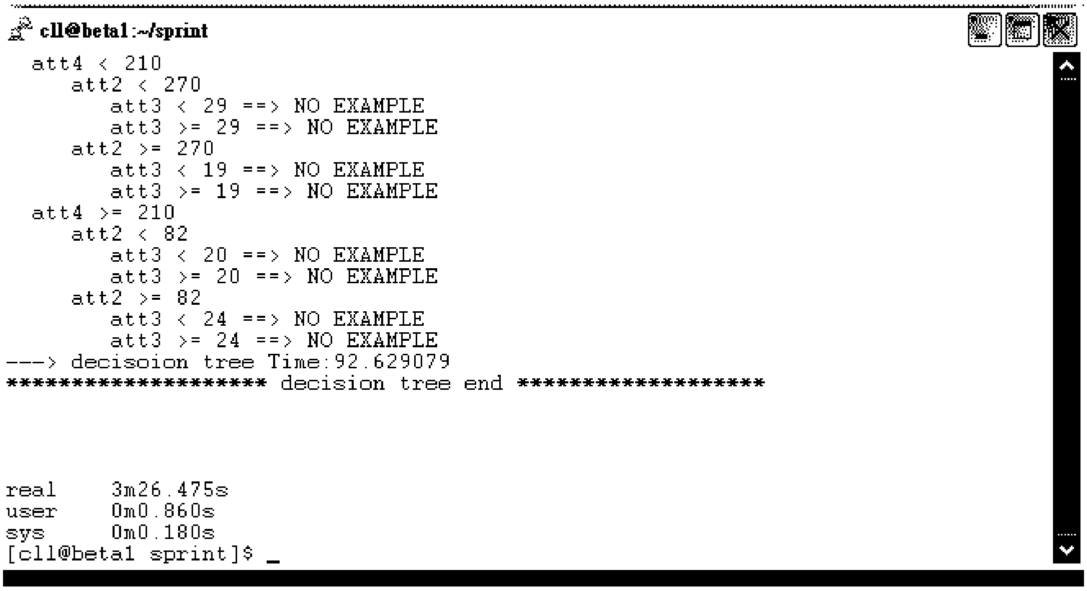
## 3. Result



```
[cll@beta1 sprint]$ time mpirun -np 8 sprint cov4 581012
================== Start MPI_Init ==============
================== End MPI_Init   ==============
DataBase File = cov4
Pattern Total = 581012

========== Start Processing partition() ========
Class Name:1
Class Name:2
Class Name:3
Class Name:4
Class Name:5
Class Name:6
Class Name:7
Attribute Name:att1:continuous
Attribute Name:att2:continuous
Attribute Name:att3:continuous
Attribute Name:att4:continuous
---> Processing partition Time:18.872807
========== End   Processing partition() ========

====== Start DisplayAttributes partition() =====
Total class            =7
Total attributes       =4
```



```
******************** decision tree start ********************
======decision tree======

att1 < 3044
  att3 < 20
     att4 < 67
        att2 < 188 ==> NO EXAMPLE
        att2 >= 188 ==> NO EXAMPLE
     att4 >= 67
        att2 < 281 ==> NO EXAMPLE
        att2 >= 281 ==> NO EXAMPLE
  att3 >= 20
     att2 < 280
        att4 < 598 ==> NO EXAMPLE
        att4 >= 598 ==> NO EXAMPLE
     att2 >= 280
        att4 < 511 ==> NO EXAMPLE
        att4 >= 511 ==> NO EXAMPLE
att1 >= 3044
  att4 < 210
     att2 < 270
        att3 < 29 ==> NO EXAMPLE
        att3 >= 29 ==> NO EXAMPLE
     att2 >= 270
```



```
  att4 < 210
     att2 < 270
        att3 < 29 ==> NO EXAMPLE
        att3 >= 29 ==> NO EXAMPLE
     att2 >= 270
        att3 < 19 ==> NO EXAMPLE
        att3 >= 19 ==> NO EXAMPLE
  att4 >= 210
     att2 < 82
        att3 < 20 ==> NO EXAMPLE
        att3 >= 20 ==> NO EXAMPLE
     att2 >= 82
        att3 < 24 ==> NO EXAMPLE
        att3 >= 24 ==> NO EXAMPLE
---> decisoion tree Time:92.629079
******************** decision tree end *******************



real    3m26.475s
user    0m0.860s
sys     0m0.180s
[cll@beta1 sprint]$ _
```

## 4. Trick

a. You can check all work queues.

$ qstat –f

b. You can delete all work queues, but you have to be root.

$ qdel –uall –f

Sometimes the load-avg is still 99.99 and states is u after qdel, then you can not execute the program, you have to wait until load-avg is less 99.99.

c. You can adjust the machine execution order by what you want. I have written a machine file, that is, machines.

$ mpirun –machinefile machines –np 8 sprint cov4 581012

$ cat machines

```
"beta1/jobmanager-grd" 1
"beta1/jobmanager-grd" 1
"beta1/jobmanager-grd" 1
"beta1/jobmanager-grd" 1
```

```
"gamma1/jobmanager-grd" 2
"gamma1/jobmanager-grd" 2
"gamma1/jobmanager-grd" 2
"gamma1/jobmanager-grd" 2
"alpha1/jobmanager-grd" 2
"alpha1/jobmanager-grd" 2
"alpha1/jobmanager-grd" 2
```

d. If you want to know what machine you will run.

```
$ mpirun –dumprsl –np 8 sprint cov4 581012
```

Note: this command just show, not run.