Instituto Superior de Engenharia do Porto

# Exploring Distributed Computing Tools Through Data Mining Tasks

**Md. Anishur Rahman**

**Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Tecnologias do Conhecimento e Decisão**

**Orientador: Prof. Dr. Paulo Jorge Oliveira**

**Júri**:

Presidente:

Prof. Dr. Fátima Rodrigues, DEI, ISEP

Vogais:

Prof. Dr. Goreti Marreiros, DEI, ISEP

Prof. Dr. Paulo Jorge Oliveira, DEI, ISEP

Porto, October 2014

# Acknowledgements

I would like to give thanks to my supervisor Prof. Dr. Paulo Oliveira for his kind support and help regarding this work.

# Dedicatória

This thesis is dedicated to my parents, to my wife, Umme Salma, who inspired me a lot and to my lovely daughter Nabiha Rahman.

# Abstract

Harnessing idle PCs CPU cycles, storage space and other resources of networked computers to collaborative are mainly fixated on for all major grid computing research projects. Most of the university computers labs are occupied with the high puissant desktop PC nowadays. It is plausible to notice that most of the time machines are lying idle or wasting their computing power without utilizing in felicitous ways. However, for intricate quandaries and for analyzing astronomically immense amounts of data, sizably voluminous computational resources are required. For such quandaries, one may run the analysis algorithms in very puissant and expensive computers, which reduces the number of users that can afford such data analysis tasks. Instead of utilizing single expensive machines, distributed computing systems, offers the possibility of utilizing a set of much less expensive machines to do the same task. BOINC and Condor projects have been prosperously utilized for solving authentic scientific research works around the world at a low cost. In this work the main goal is to explore both distributed computing to implement, Condor and BOINC, and utilize their potency to harness the ideal PCs resources for the academic researchers to utilize in their research work. In this thesis, Data mining tasks have been performed in implementation of several machine learning algorithms on the distributed computing environment.

Keywords: Distributed computing, Condor, BOINC, Data mining task

# Resumo

Tirar partido dos recursos de CPU disponíveis, do espaço de armazenamento, e de outros recursos de computadores interligados em rede, de modo a que possam trabalhar conjuntamente, são características comuns a todos os grandes projetos de investigação em grid computing. Hoje em dia, a maioria dos laboratórios informáticos dos centros de investigação das instituições de ensino superior encontra-se equipada com poderosos computadores. Constata-se que, na maioria do tempo, estas máquinas não estão a utilizar o seu poder de processamento ou, pelo menos, não o utilizam na sua plenitude. No entanto, para problemas complexos e para a análise de grandes quantidades de dados, são necessários vastos recursos computacionais. Em tais situações, os algoritmos de análise requerem computadores muito potentes e caros, o que reduz o número de utilizadores que podem realizar essas tarefas de análise de dados. Em vez de se utilizarem máquinas individuais dispendiosas, os sistemas de computação distribuída oferecem a possibilidade de se utilizar um conjunto de máquinas muito menos onerosas que realizam a mesma tarefa. Os projetos BOINC e Condor têm sido utilizados com sucesso em trabalhos de investigação científica, em todo o mundo, com um custo reduzido. Neste trabalho, o objetivo principal é explorar ambas as ferramentas de computação distribuída, Condor e BOINC, para que se possa aproveitar os recursos computacionais disponíveis dos computadores, utilizando-os de modo a que os investigadores possam tirar partido deles nos seus trabalhos de investigação. Nesta dissertação, são realizadas tarefas de data mining com diferentes algoritmos de aprendizagem automática, num ambiente de computação distribuída.

Palavras-chave: Computação distribuída Condor, BOINC, tarefa de mineração de dados

# Table of Contents

# List of figures

# List of tables

# List of Acronyms

| | |
|---|---|
| **OLAP** | On-Line Analytical Processing |
| **KDD** | Knowledge Discovery in Databases |
| **JDBC** | Java Database Connectivity |
| **MPP** | Massively Parallel Processors |
| **ML** | Machine Learning |
| **EM** | Expectation Maximization |
| **K-NN** | K-nearest neighbor's algorithm |
| **SVM** | Support vector machine |
| **FT** | Functional Trees |
| **ADTree** | Alternating decision trees |
| **WU** | Work Unit |
| **DM** | Data Mining |
| **WEKA** | Waikato Environment for Knowledge Analysis |
| **BOINC** | Berkeley Open Infrastructure for Network Computing |

# 1 Introduction

With the evolution of data accumulating technology the amounts of data stored in both corporations and research laboratories is astronomically immense (Figure 1). Such amounts of data may contain valuable information but cannot be handled anymore by human analysis by hand. Such amounts of data require computer-predicated automatically implements. Associated with Databases, a set of analytical implements, called On-Line Analytical Processing (OLAP), have been developed. These implements, however, enable a circumscribed and a single set of analysis tasks. For more utilizable and involute in-depth analysis, we require a different approach, we require a methodology and a set of techniques called Knowledge Discovery in Databases (KDD). KDD enables analysts to address a set of intricate data analysis quandaries such as Classification, Regression, Clustering, Association Rule revelation etc. far beyond the reach of the OLAP approach.

However, for involute quandaries and for astronomically immense amounts of data, sizably voluminous computational resources are required. For such quandaries, one may run the analysis algorithms in very puissant and expensive computers, which reduces the number of users that can afford such data analysis tasks.

Instead of utilizing single expensive machines, Distributed Computing offers the possibility of utilizing a set of much less expensive machines to do the same task.

We argue that distributed computing is a less expensive and subsidiary alternative for processing immensely colossal amounts of data in a KDD context.

In this thesis, we have explored distributed computing implements such as HTCondor, BOINC and have explored WEKA implements for data mining tasks.

In this thesis, we have got enforced a monitoring tool for the distributing computing environment that ought to embrace facilities for measuring performance, monitoring the behavior, debugging and fine tuning throughout the system´s operation.

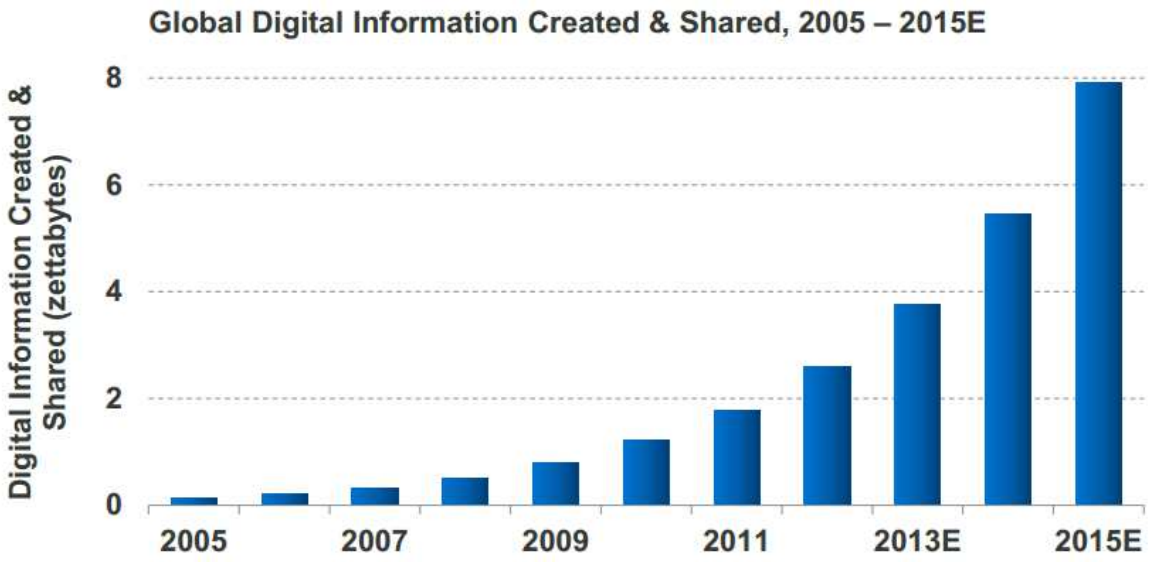



**Figure 1 - Global Digital Information [Gantz and David, 2011]**

## 1.1 Existing Systems

In this section we discuss about the Machine Learning systems for Data Mining and Distributed Computing systems.

### 1.1.1 Systems for Data Mining

The rudimental operation of the Data Mining step of a KDD process is to construct a model. Customarily, a Machine Learning (ML) algorithm is utilized. There are an abundance of implementations of ML algorithms available on the Web for all of the DM quandaries (Classification, Clustering, etc.). These implementations run on different platforms and require, customarily, some expertise to tune their parameters in order to construct good models. If one wants to utilize several algorithms for the same DM quandary, then one has to amass several implementations and run each one by hand.

#### 1.1.1.1 WEKA

WEKA stands for the Waikato (W) Environment (E) for Knowledge (K) Analysis (A), developed by the Department of Computer Science, University of Waikato, New Zealand [Hall et al., 2009]. The system is written in Java, an object oriented programming language that is widely available for all major

computer platforms and WEKA can be run on Windows, Linux and Macintosh operating systems. Java is chosen for implementing the machine learning techniques. As an object oriented programming language, It allows a uniform interface to many different learning algorithms and methods for pre- and post processing. Moreover, for evaluating the result of learning schemes on any given dataset [ Witten et al., 2012 ]. The main contribution of WEKA is that it provides a uniform environment for running a plethora of ML algorithms and additionally some pre-processing procedures required in the KDD process utilizing WEKA and we can compare the results of different algorithms running them in the same environment. We have still to call each one of them individually and then manually compare the resulting models.

#### 1.1.1.2 Rapid Miner

Rapid Miner [Hofmann and Ralf, 2013] formerly YALE is one of the most widely used open-source data mining suites and software solutions. It has available a wide variety of ML algorithms. It has interfaces to WEKA so algorithms available in WEKA are accessible from Rapid Miner. RapidMiner sanctions the user to designate the steps of a KDD process and then run designated KDD process. However, this possibility of designating the workflow of the KDD process is run sequentially. No interfacing with a distributed system is provided. If the KDD process is involute and the amount of data is profoundly and immensely colossal, then the sequential execution may become a quandary.

#### 1.1.1.3 IBM Modeler (Clementine)

IBM SPSS Modeler [Green and Neil, 2010] is a data mining software developed by IBM. It facilitates users to utilize statistical and data mining algorithm without programming. It is pristinely a GUI predicated methodology. The following table shows Modeling Techniques in IBM SPSS Modeler.

**Table 1 - ModellinModelerues in IBM SPSS Modeler**

| Technique | Usage | Algorithms |
|---|---|---|
| **Classification (or prediction)** | Used to predict group membership or a number | Auto classifiers, Decision Trees, Logistic, SVM, Time Series, etc. |
| **Segmentation** | Used to classify data points | Auto Clustering, K-means, |

| | | |
|---|---|---|
| | into groups that are internally homogeneous and externally heterogeneous.<br><br>Identify cases that are unusual | etc.<br><br>Anomaly detection |
| **Association** | Used to find events that occur together or in a sequence | APRIORI, Carma, Sequence |

We have culled Weka for our work because of Weka software is simple to utilize and far expeditious than Rapid miner. Weka has good documentation about to learn simply the way to utilize Weka, however on the opposite hand Rapid miner does not have excellent documentation and it is hardly attainable to grasp the way to utilize Rapid miner from the manual. More importantly, no interfacing with a distributed system is provided by the Rapid miner. If the KDD process is intricate and also the amount of data is incredibly larger than the sequential execution may become a problem.

Moreover, we have got chosen Weka over Clementine because of it offers much more data mining and analysis possibility for no price.

### 1.1.2 Distributed Computing

#### 1.1.2.1 CONDOR

Condor [Thain et al., 2005] is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to Condor, Condor places them into a queue, culls when and where to run the jobs predicated upon a policy, monitors their progress, and apprises the user upon completion. Condor efficaciously utilizes the computing power of workstations that communicate over a network. Condor can manage a dedicated cluster of workstations. Its power emanates from the ability to efficaciously harness non-dedicated, pre-subsisting resources under distributed ownership.

#### 1.1.2.2 BOINC

BOINC [Korpela, 2012] stands for Berkeley Open Infrastructure for Network Computing. BOINC is an open source software platform to sanction distributed computing projects which use volunteer computer resources to run. Volunteer computing platforms such as BOINC are currently the most prosperous Distributed computing systems, which rely on donated computer cycles from mundane citizen communities. BOINC is currently being prosperously utilized by many projects to analyze data. BOINC is open source and is available at http://boinc.berkeley.edu.

## 1.2 Aim of the Thesis

The aim of the thesis is to facilitate the solution of Data Mining quandaries. To accomplish this, we require a distributed computing environment and subsisting implementations of several ML algorithms. For the purport we will evaluate the usefulness of two distributed computing environments, Condor and BOINC, and we will utilize the data mining tasks through WEKA software for the ML implementations.

## 1.3 Document Structure

This thesis is divided into five main chapters.

The current chapter contains the Introduction, subsisting systems, quandaries with the subsisting system, the aim of the thesis and document structure.

The second chapter is followed by State of Art that includes Knowledge Discovery in Databases (KDD), Data Mining, Weka, Distributed computing systems and monitoring systems. This chapter intends to show a general view of the subsisting options for each of the essential main areas of thesis work development.

The third chapter is fixated on Data mining system Weka and the Distributed computing systems, Condor and BOINC, and identified the distributed computing monitoring systems Nagios.

The fourth chapter describes the implementation of data mining task in distributed systems and briefly discusses the output results.

Finally, in the fifth chapter is presented the Conclusion of the thesis and about the future work.

# 2 State of Art

This chapter introduces and explores the technologies. For that, a theoretical background must be comprehended . The different matters are divided in this chapter by five stages.

The first section refers to Knowledge Discovery in Databases (KDD), the process of analyzing data.

The second and third section refers to Data mining and Weka and elongated Weka works in the grid environment.

The fourth section refers to distributed computing systems, discuss about the CONDOR and BOINC systems.

The fifth section refers to monitoring system for monitoring of resources in the grid for failure detection, notifications and automatic recuperation.

## 2.1 Knowledge Discovery in Databases (KDD)

KDD is the process of analyzing data from divergent perspectives and summarizing it into subsidiary information; information that can be acclimated to incremental revenue, cuts costs, or both. KDD software is one of a number of analytical implements for analyzing data [Liu, 2010]. It sanctions users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, KDD is the process of finding correlations or patterns among dozens of fields in immensely colossal relational databases.

KDD is a process that involves a sequence of steps shown in Figure 2 [Fayyad, 1996] that we now describe.

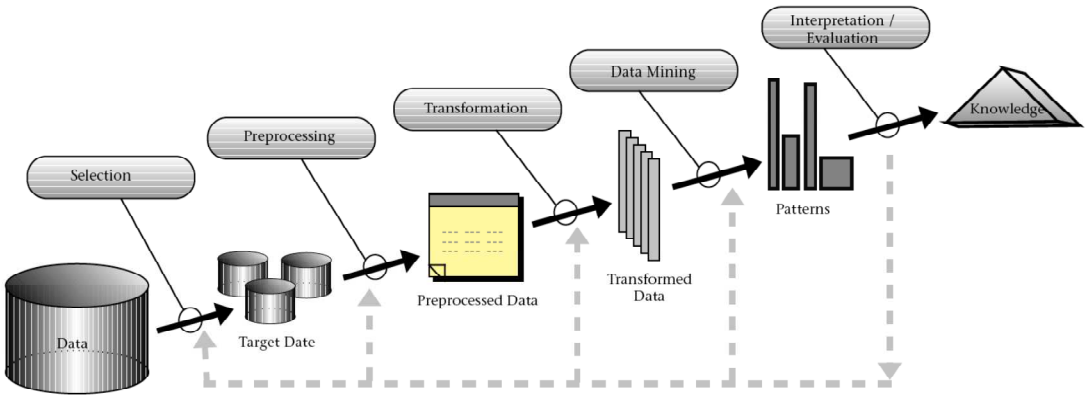



**Figure 2 - Sequence of steps in the process of knowledge discovery [Fayyad, 1996]**

1. Extract data from the sources.
2. Data cleaning to abstract noise and inconsistently erratic data.
3. Data integration where multiple data sources may be cumulated.
4. Data is culled where data pertinent to the analysis task are retrieved from the database.
5. Data transformation where data are transformed or consolidated into forms opportune for mining by performing summary or aggregation operations, for instance.
6. Data mining an essential process where keenly intellective methods are applied in order to extract data patterns.
7. Pattern evaluation to identify the authentically intriguing patterns representing Knowledge predicated on some intriguing measures.
8. Knowledge presentation where visualization and knowledge representation techniques are used to present the mined knowledge to the user

The Knowledge grid works for Knowledge revelation in the grid environment [Ahmed et al., 2010], [Zhuge, 2002]. The Knowledge Grid that offers a high-level system to provide Grid-predicated

Knowledge revelation accommodations. Through these accommodation professionals and scientists are sanctioned to engender and manage involute Knowledge revelation applications. These types of application are composed like workflows that integrated data sets and mining implements provided as distributed accommodations on a grid. Moreover, with these accommodations, users are sanctioned to store, share and execute these Knowledge revelations workflows [Cesario, 2012]. It is sanctioned end users to be more concentrated on the Knowledge revelation process rather not to be more worried about the Grid infrastructure details.

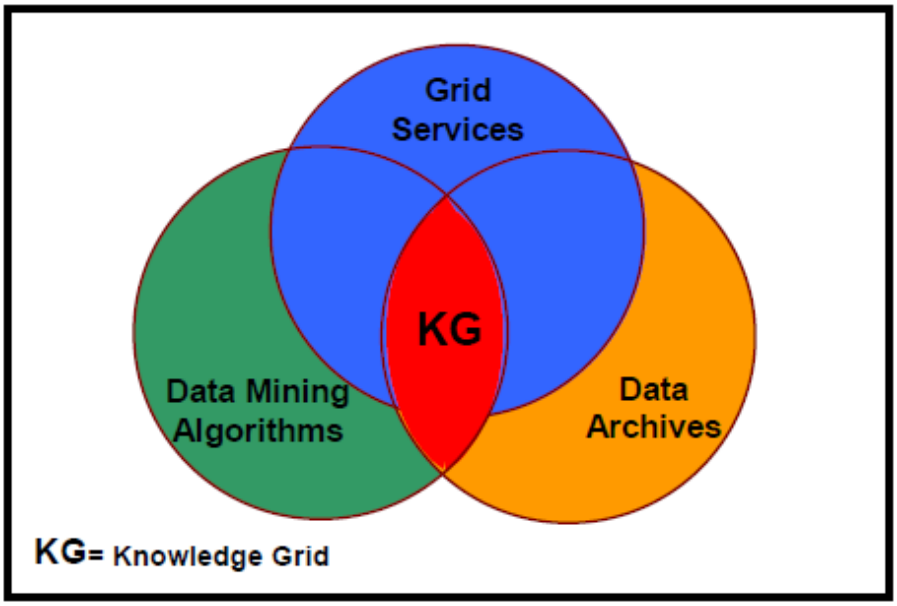



**Figure 3 - Knowledge grid system [Congiusta et al., 2008]**

## 2.2 Data Mining

The Data Mining step may interact with the user or a Knowledge base. The intriguing patterns are presented to the user and may be stored as incipient Knowledge in the Knowledge base. According to this view, data mining is only one step in the entire process, albeit an essential one since it denudes hidden patterns for evaluation. Data Mining refers to the KDD step when an algorithm is applied to the data to construct a model. This step as received a great deal of attention and is considered as the main step of the KDD process. Due to these two considerations, it has often seemed in the literature that the term Data Mining is utilized as a supersession to KDD. In this thesis,

we will utilize Data Mining and KDD interchangeably. The steps afore Data Mining are globally called data pre-processing steps [Han, 2011].

The number and nature of the steps in the KDD process is not rigid. Several proposals have been made. However, a consequential methodology for Knowledge discovery is now a standard called CRISP-DM (CRoss Industry Standard Process for Data Mining) [Shearer, 2000], [Olson and Shi, 2006]. Utilizing Data Mining one can transcend the OLAP analysis and address several categories of intricate quandaries. The main quandaries Data Mining can address in the following:

- Classification problems where the goal is to construct a predictive model that, given a new case (register), can determine its class label.

- Regression Data Mining is utilized to construct a function that prognosticates the numerical value of one of the variable (dependent variable) utilizing the other independent variables.

- Clustering Data items are grouped according to logical relationships or consumer predilections. For example, data can be mined to identify market segments or consumer affinities.

- Association Rule revelation Data can be mined to identify associations. The potation-diaper example is an example of associative mining.

Data mining is sanctioned to investigate information from immensely colossal databases to find a solution for the business decision quandary [Larose, 2014]. It is somehow the extensions of statistics. Data mining is a technology, such as to find or to receive information regarding an incipient product from a retail shop. Utilizing this information, data mining software can build a

model of the comportment of customers to predict the customer response to the incipient product. Utilizing this result, a marketing manager can make some business decision to promote the incipient product to the right customers. Nowadays, data mining applications are available on all size systems for mainframe, client-server, and PC platforms. System prices range from several thousand dollars for the most minute applications up to USD 1 million a terabyte for the most immensely colossal. Enterprise-wide applications generally range in size from 10 gigabytes to over 11 terabytes. Relational database storage and management technology is adequate for many data mining applications less than 50 gigabytes. However, this infrastructure needs to be significantly enhanced to fortify more immensely applications. Some vendors have integrated extensive indexing capabilities to ameliorate query performance. Others use incipient hardware architectures such as Massively Parallel Processors (MPP) to achieve order-of-magnitude amendments at query time.

## 2.3 WEKA

The Waikato Environment for Knowledge Analysis (Weka) is an open source tool is developed in Java. It is a collection of machine learning algorithms for data mining tasks [Witten et al., 2011], [Eibe et al., 2005]. Weka tool is included with the data preprocessing, classification, regression, clustering, association rules and visualization. It also has included a variety of tools for transforming data sets.

Weka is very popular for application of practical implementation of data mining and machine learning tools and techniques. It also have recognized as a landmark system in data mining and machine learning [Mehemed, 2011].

Some main features of WEKA are following below:

- Data preprocessing – WEKA supports the ARFF format (native file format) and also other format supports such as CSV, Matlab, ASCII files
- Classification – Classifier is divided into Bayesian method (Naïve Bayes, Bayesian nets, etc.), Lazy methods (nearest neighbor and variants), rule-based methods (decision tables, OneR, RIPPER), tree learners (C4.5, Naïve Bayes tree, M5), function-based learners (linear

regression, SVMs, Gaussian processes) and miscellaneous methods. Furthermore, WEKA includes meta-classifiers like bagging, boosting, stacking; multiple instance classifiers; and interfaces for classifiers implemented in Groovy and Jython.

- Clustering – Unsupervised learning is supported by several clustering schemes, including EM-based mixture models, k-means and various hierarchical clustering algorithms.
- Attribute selection – Various selection criteria and search methods are available in WEKA.
- Data visualization – This is the specialized tools for visualization of data in WEKA.
- WEKA also support for association rule mining, comparing classifiers, data set generation.
- WEKA supports Graphical User Interface.

Weka tools [David and Ian, 2013] are provided with three main graphical user interfaces that is easily accessible through the underlying functionality. These three graphical user interfaces are included:

- Explorer
- Classify
- Experimenter

Weka implements are fortifying several input file formats such as CSV, LibSVM's format, C4.5's format and ARRF (Attribute-Relation File Format). In Weka, different data mining algorithms are implemented through graphically connecting components that is representing data sources, preprocessing implements, learning algorithms, evaluation methods and visualization implements. Through a command line or graphical user interface, the user is able to select one or more options of algorithms according to their work merit. It is easy to implement learning algorithm to a dataset through Weka implements. It has an abundance of implements to transform datasets. Algorithms for decentralized process is included in weka implements, is kenned as the most consequential task in data mining. Without writing a bunch of programming code, a user is able to reprocess a dataset, feed into a cognition scheme and analyzes the results and its performance through Weka implements.

Weka runs only on a single machine, it is a standalone application. To overcome this constraint, several projects have been developed to elongate Weka for distributed data mining [Talia and Paolo, 2010]. The development projects are as follows: Weka-Parallel – facilities a distributed cross-validation, GridWeka2 – facilities distributed scoring and testing as well as cross-validation, Weka4WS – facilities web accommodations and WekaG – facilities data mining Weka toolkit to perform data mining tasks on a grid environment.

### 2.3.1 Weka Parallel

Weka-Parallel is a modification of Weka and it is written in Java program. The main purport of Weka parallel is to reduce the time for running cross-validation on a dataset utilizing any given classification. Weka parallel can handle the cross-validation, result aggregation and multiprocessor communication [Engel, 2014].

### 2.3.2 Grid-enable Weka

This is developed by Xin Zuo under the University College Dublin [Rinat, 2004], [Zeng, 2012]. Grid-enable Weka consists of two main components, which are predicated on Weka-parallel. One is the Weka-server and another one is the Weka-client. On the server side the original Weka is configured and the cognition task and distributing work are done on the client side. This implement is much more convenient for a person who works with a machine learning task, but not much habituated with parallel computing and Grid technology.

### 2.3.3 Weka G

It is a system that usages client server architecture. In this system data mining algorithm is implemented through the server and the grid accommodations are implemented through the client side. The architecture of Weka G is predicated on the Weka and Globus toolkit 4, is called Data Mining Grid Architecture (DMGA). Weka G support parallel algorithm and additionally provides a user interface like as Weka [Pérez et al., 2005].

### 2.3.4 GridWeka2

GridWeka2 is a modified version of WEKA. It is written in Java and modified by Eibe Frank under the University of Waikato in New Zealand. It has facilitated to run cross-validation within the Weka Explorer and Experimenter in parallel and conjointly within the distributed over several machines. GridWeka2 has some constraints [Altorf, 2007]. There are as follows:

- It is a research prototype and under development
- It does not have any authentication mechanism
- Parallelization does not work command line interface
- No valid timing information is stored in the output files

### 2.3.5 Weka4WS

Weka4WS [Talia et al., 2005], [Zheng et al., 2010] is developed by utilizing Java software with WSRF library that is provided by the Globus Toolkit (GT4). Web services Resource Framework (WSRF) is utilized by the Weka4WS to facilitate the GRID environment. Weka4WS usages GT4 services for implementing the standard grid functionality which includes security, data management, etc. It provides distributed data mining in grid environments. The architecture of Weka4WS is enclosed with three kinds of nodes:

- Storage nodes - contains the data sets
- Compute nodes – data mining algorithms are run on remote nodes
- User nodes – local machine of users

**Table 2 - Difference between several projects of extend Weka**

| Feature/system | Weka | WekaG | Grid-enabled Weka | Weka Parallel | GridWeka 2 | Weka4WS |
|---|---|---|---|---|---|---|
| **Architecture** | Single | DMGA | C/S | C/S | C/S | C/S |
| **Open source** | Yes | No | No | No | Yes | Yes |
| **Grid enabled** | Yes | Yes | Yes | Yes | Yes | Yes |
| **Extensibility** | Yes | No | No | No | No | Yes |
| **Platform Independence** | Yes | Yes | Yes | Yes | Yes | Yes |

Note: C/S – Client / Server, DMGA – Data mining grid architecture.

## 2.4 Distributed computing systems

It has been spectacular amelioration recent years in computer technology like additional potent machines, impressive storage capacity, high connection speed. However, most of the time users utilize a fraction of this potency. With some few installation of free software by few clicks, users can involve their powerful machine with distributed systems network that would sanction astronomically immense progress in several areas of public and academic scientific research.

Distributed computing systems, are designed to provide computational power that is accessible in the same manner that electricity is offered from the electricity grid, we have a propensity to merely plug into it and do not have to be compelled to worry regarding wherever the power is emanating from or how it got there. The thought of Distributed Computing systems is just like the electricity metaphors, if supplemental computing power is needed, need not to fret regarding it; spare CPU cycles on other computers are utilized. This suggests that the super-computer type power is accessible without expending immensely prices of buying a super-computing power. On the opposite hand, the idle CPU cycles are placed into good use.

Distributed Computing is not simply a conception for discussing, however, are some things else that is authentically used daily. There are several Distributed Computing systems around the world, and many researchers investigating how to improve Distributed Computing systems.

### 2.4.1 Condor

The condor is a software system that is developed by Miron Livny´s research group at the University of Wisconsin [Thain, 2005]. It was disclosed to the public in 2003. The condor is termed high throughput distributed computing systems. It is a batch system [Bradley, 2011], that has a high ability to manage job scheduling, policy, priority, resources management and monitoring. High throughput computing and opportunistic computing are infrequent in other batch systems. The following are unique and potent implemented in the Condor system.

- ClassAds
- Job checkpoint and migration
- Remote system calls

These implements sanction Condor to manage efficaciously the PC on the network and search and manage the idle CPU power from the workstations [Zach, 2010]. The condor is organized in such a way that it only can run jobs on workstations once the keyboard and CPU are idle. Running jobs are ceased and migrate to different workstation only when user back in his work and start working with his keyboard. The remaining task of that running job is sifted in different workstation until the task has finished.

A simplified diagram is shown in Figure 4, how Condor can work as a grid in both ends such as front end and back end.

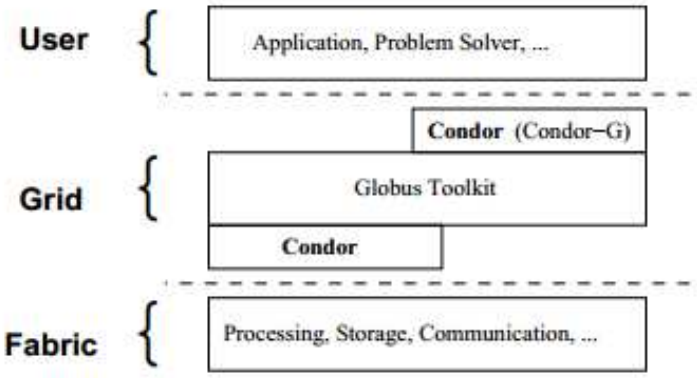



**Figure 4 - Condor technology [Thain, 2005]**

The above figure shows a simplified architecture of high throughput computing systems of Condor.

**User:**

Users can submit their jobs with their categorical requisite of operating systems and categorical requisite of amount of memory to run their jobs. Users can submit their job through a script.

**Grid:**

- **Condor-G** – is used as the reliable submission and a job management service for one or more sites

- **Condor** – High Throughput computing system, Condor is utilized as the fabric management service as a grid generator for one or more sites, and
- **Globus Toolkit** – is open source software that is utilized to build up Grid systems and application. [Mohan, 2012], [Sotomayor, 2006] Globus toolkit provides the grid security infrastructure (GSI) which sanctions collaborators to apportion resources without blind trust. Globus toolkit provides grid resource allocation management (GRAM) which is responsible for grid job management and additionally responsible for remote process queuing and execution of several potent features such as vigorous security, file transfer etc. Globus toolkit additionally provides GridFTP which provides the high-performance; reliable data transfer and is the FTP predicated protocol, which is an Open Grid Forum (OGF) recommended data movement protocol. Globus includes programs such as:

- **GRAM** (Globus Resource Allocation Manager) - convert a request for resources into commands that local computers can understand

- **GSI** (Grid Security Infrastructure) - authenticates users and determines their access rights

- **MDS** (Monitoring and Discovery Service) - collects information about resources such as processing capacity, bandwidth capacity, type of storage, and so on

- **GRIS** (Grid Resource Information Service) - queries resources for their current configuration, capabilities, and status

- **GIIS** (Grid Index Information Service) - coordinates arbitrary GRIS services

- **GridFTP** (Grid File Transfer Protocol) - provides a high-performance, secure and robust data transfer mechanism

- **Replica Catalog** - provides the location of replicas of a given dataset on a grid

- **Replica Management system** - manages the Replica Catalog and GridFTP, allowing applications to create and manage replicas of large data sets.

Globus toolkit is utilized as a bridge in between the Condor-G and Condor.

**Fabric:**

The Condor high throughput computing system is utilized as the fabric management services (a grid generator) for one or more sites and Globus toolkit is utilized here to make a bridge in between them.

With the astronomically extensive feature of the Condor, sanctions users to take the advantage of the idle resources of their workstation to put them to good use. Condor provides consequential features are as follows:

- **Distributed submission** – Condor allows users to submit their jobs from many machines

- **Job priorities** – Condor allows users to assign their priority on their own submitted jobs
- **User priority** – Condor allows administrators to assign priority to the users
- **Job dependence** – Condor allows enforcing dependency to handle easily the job dependencies between jobs.
- **Support for multiple job models** – Condor allows serial jobs and parallel jobs incorporating PVM, dynamic PVM and MPI.
- **ClassAds** – Condor allows match making by resource requests (jobs) with resource offers (machines).
- **Job checkpoint and migration** – Condor allows checkpoint that gives a snapshot for a job complete state and left off jobs or uncompleted jobs migrate to another machine.
- **Periodic checkpoint** – Condor also allow a periodic checkpoint for their jobs.
- **Job suspend and resume** – Condor allow a job could be suspended and to resume it later.
- **Remote system calls** – Condor´s remote systems call functionality allows communicating in between master and workstations.
- **Pools of machines can work together –** Pools are connected machines in a network that sharing their resources.
- **Authentication and authorization** - Condor provide strong authentication and authorization mechanisms, encryption of data sent across a network and data integrity.
- **Heterogeneous platforms** - Condor support heterogeneous platforms to execute different types of applications.
- **Grid computing** - Condor is used as a grid computing because it is integrated with the Globus toolkit. Flocking technology allows Condor to create a grid computing environment.

In the next Figure we have described concerning the Condor-G system, that leverages software from Globus and Condor to provide users for harnessing multi-domain resources [Frey J., et al. 2002]. It is

shown through the graphical presentation concerning the structure of Condor-G and conjointly has shown the way to handle job management, resource selection, security and fault tolerance.

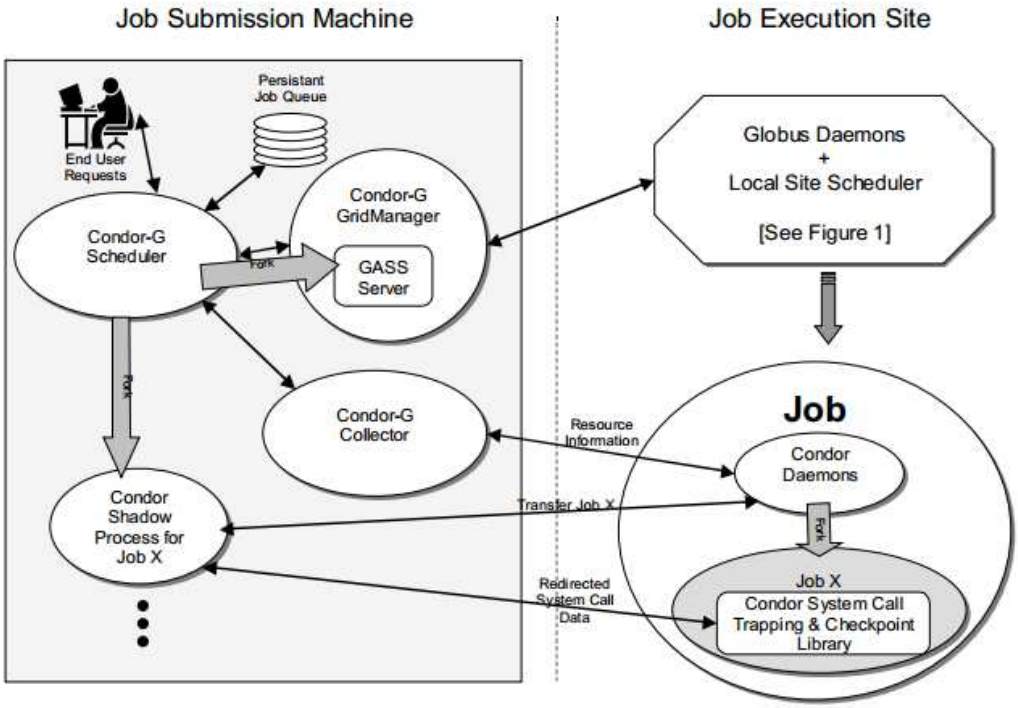



**Figure 5 - Job submission and execution in Condor-G technology [Frey et al., 2002]**

### 2.4.2 BOINC

The Berkeley Open Infrastructure for Network Computing (BOINC) [Ries et al., 2012] is an open source middleware software system for the grid computing infrastructure. It is developed to support for the project of SETI@home [Korpela, 2012]. SETI@home project presently running with about 1 million volunteer computers. There are other projects that are using BOINC.

- Predictor@home – [Michigan, 2014] studying protein behavior at the Scripps Research Institute at the university of Michigan.

- Folding@home – [Stanford, 2014] studying protein folding, misfolding, aggregation and related diseases at the Stanford University.

- Climateprediction.net – [Oxford, 2014] studying in long-term climate prediction at Oxford University.

- Climate@home – [NASA, 2014] is based at NASA and studying climate prediction same as Climateprediction.net under the collaboration of researchers at NCAR, MIT, UCAR, Rutgers, Lawrence Berkeley Lab and U.C. Berkeley.

- LHC@home – [CERN, 2014] is a CERN project and studying the behavior of the LHC (Large Hadron Collider), is situated in Geneva, Switzerland.

- Einstein@home – [Einstein, 2014] studying to detect certain types of gravitational waves under the collaboration of researchers at the University of Wisconsin, U.C. Berkeley, California Institute of Technology, LIGO Handford Observatory, University of Glasgow and the Albert Einstein Institute.

- UCB/Intel study of Internet resources – [Berkeley, 2014] studying the structure and performance of the consumer Internet under the collaboration of researchers at the U.C. Berkeley computer sciences department and the Intel Berkeley Research Laboratory.

BOINC has two main features which include:

- Multiplatform – MAC OS X, MS Windows, GNU/Linux.

- Open source - is available under the URL at http://boinc.berkeley.edu.

BOINC works as master-slave architecture. The slaves are the workstations and master is the main server that distributes the work among the workers.

- BOINC client – download a BOINC project and installs the BOINC client on a workstation are becoming a BOINC client. A registered user, through a BOINC project website, is identified by a unique email id. Through the unique e-mail ID a user can become a part of a many project by attaching project. A user can allocate percentage of time for each project. It will determine how the computer resources are divided into the projects. Jobs on the client are processed in FIFO style. Clients communicating with the server via using the Hypertext Transfer Protocol (HTTP). The client is available for Windows and Linux on Intel X86 architectures, for Mac OS X on PowerPC and Solaris on SPARC architectures. The BOINC client operates in several modes:

- As a screensaver – display graphics of running application
- As a Windows service – run application when no user is logged-in
- As an application – provides a tabular view of projects, work, file transfer, and disk usage.
- As a Unix command-line program – through stdin, stdout, and terror

BOINC also provides tools for remote clients to participate by remotely install the client software on a large number of machines.

- BOINC server – is a place where the scientific research project is hosted. The server serves the following:
  - Hosting scientific project
  - Create and distribute jobs
  - Assimilate and validate the results

A BOINC project consists of the following components:

- A database
- A directory structure
- A configuration file

A BOINC project is included with a set of daemons:

- Work generator – creates work units and corresponding input files
- Feeder – creates a shared-memory segment used to pass database records in the CGI scheduler process
- Transitioner – handles state transitions of work units and results
- Validator – it compares redundant results
- Assimilator – it handles work units

- File deletion – deletes input and output files when they are no longer needed
- Database purging – removes work related database entries when they are no longer needed

The arrow in black-ink is representing the communication in between the elements. The red-arrow is indicating the flow of the primary computation. The multiple - arrow is indicating several parallel instances that can be run simultaneously.

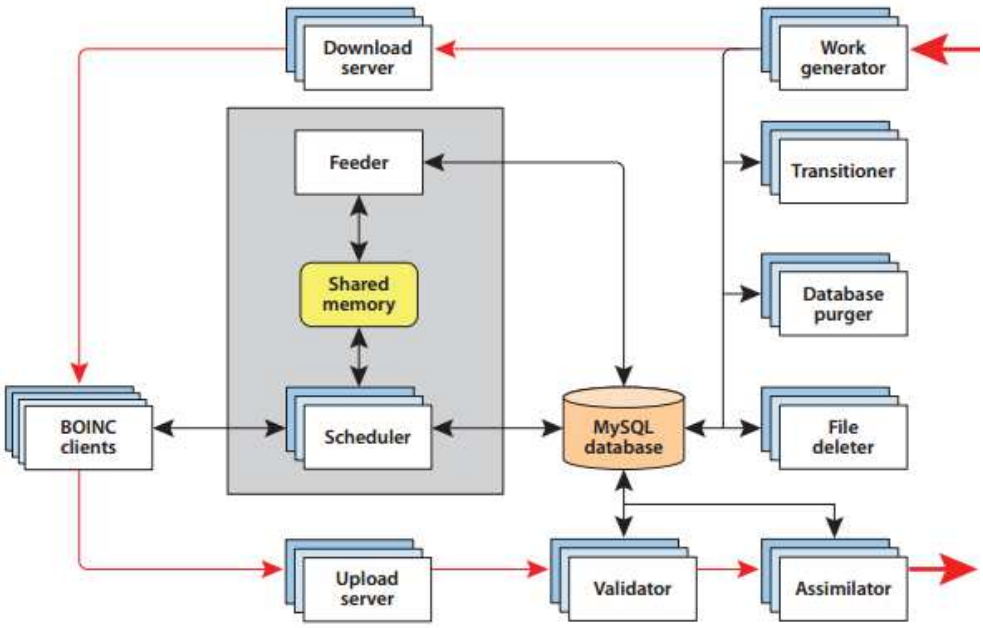



**Figure 6 - Berkeley Open Infrastructure for Network Computing [Korpela, 2012]**

### 2.4.3 Difference between CONDOR and BOINC system

There is an immensely difference between CONDOR and BOINC at an abstract level. CONDOR is denoted to run very heterogeneous jobs inside the boundaries of an organization and BOINC is supposed to run very homogeneous jobs outside the boundaries of an organization. One allegation of this is that CONDOR trusts the resources it utilizes, whereas BOINC does not. Because of the lack of trust, BOINC does redundant computing to endeavor and eschew cheating and it signs executables to amend the sercurity since it is in operation on the Internet, whereas CONDOR does nothing of this kind albeit it can be configured to authenticate entities and encrypt all communication. A consequential technical difference lies in the way communications are done. By default CONDOR uses port 9618 and 9614 to communicate. These are not prominent ports, as designated in RFC793, and are therefore infrequently open in firewalls. CONDOR utilizes the push method of distributing jobs, i.e. the central resource manager connects to the execution machines.

This scheme cannot be utilized by BOINC since many of its users sit behind firewalls, and asking them to transmute their firewall configurations is not liable to prosper. Therefore, BOINC uses only port 8020 for communication and users pull jobs from the server intricate, so connections are established from the client to the server in lieu of the other way around. Because of its matchmaking and multiple universes, many different job types can be run on the CONDOR making it much more multifarious than BOINC. BOINC requires specially designed server side functions tailored to the categorical application, as well as linking the application with the BOINC API. This places a sizably voluminous obstruction in the way when it comes to running jobs under BOINC. Because porting applications to BOINC requires quite marginally of effort, this effort only pays off when the application has to be run many times, whereas in CONDOR fundamentally any application with binary compatibility can be run under the Vanilla universe with very little effort. If we consider the standard universe, the migration and check pointing sanctions for very long running jobs, as opposed to BOINC, which does not have direct built-in support for these features. Check pointing can be done in BOINC, but it requires considerable work by the application categorical backend.

BOINC strive more to magnetize volunteers and regale them by having graphics and progress designation of jobs as well as facile access to information about running and buffered jobs. All of this should make it more facile to magnetize users to a BOINC predicated project, as opposed to a CONDOR predicated project. BOINC fortifies Public Resource Computing more preponderant than CONDOR in terms of getting reliable results, protecting users and overcoming firewalls. BOINC is a true Public Resource Computing platform, whereas CONDOR is more suited for intra-organizational use.

**Table 3 - Comparing BOINC vs CONDOR**

| **Items** | **BOINC** | **CONDOR** |
|---|---|---|
| **Application** | Applications for public distributed computing | Compute-intensive programs |
| **Operational Systems** | MS Windows, OS X, Linux/x86, Solaries/SPARC | MS Windows, OS X, Linux, AIX |
| **Grid computing opportunistic** | Yes | Yes |
| **Open source** | Yes | No |

| | | |
|---|---|---|
| **Free binary** | Yes | Yes |
| **Object-oriented Implementation** | Yes | No |
| **Support multiple applications** | Yes | Yes |
| **Support communication among peers** | No | Yes |
| **Support communication between servers** | Yes | Yes |
| **Support parallel applications** | No | Yes |
| **Interoperability** | No | Yes |
| **Architecture** | Centralized | Centralized (Central manager) |
| **Fault Tolerance** | Supports separate schedulers and has an intelligent algorithm to prevent customers from overloading the server after a down-time. | Checkpointing |
| **Security Mechanism** | use of encryption and authentication for public keys | Remote access to a restricted login account |

## 2.5 Monitoring systems

The monitoring system is the important part of management of distributed system activity. The main requirements of a monitoring system are to quickly identify the potential failure and to transmit messages to the user. The key design features of monitoring systems include scalability, extensibility, portability, robustness, manageability, reasonable overhead and secure. The basic usages of a monitoring system are

- Troubleshooting and recovery of operating system components,
- Performance analysis and enhancements of system operations,

- Support to the component performing job scheduling,
- Support to the component performing resource management
- Collection of information on application
- Security threats and hole detection

After surveying some other systems we have decided to select the Nagios open source program for our implementation. We have discussed the advantages and disadvantages of various monitoring systems as follows:

### 2.5.1 Ganglia

The ganglia monitoring system [Matt, et al., 2012] is a scalable distributed monitoring system suitable for high performance computing systems like clusters and Grids. The features of the Ganglia monitoring systems are:

- Hierarchical data model
- Hierarchical Dynamic Shared Object (DSO) plug-in
- Interactive Gmond/Gmetad
- Trigger plug-in
- Rich ganglia query system
- Gmetad and Web front-end
- XML data compression

The Ganglia monitoring system is build-up with three main components:

1. Ganglia Monitoring Daemon (gmond) – is a lightweight service is installed on every machine.
2. Ganglia Meta Daemon (gmetad) – is a service to collect data from other gmetad and gmond sources and it also has a simple query mechanism for collecting specific information about groups of machines.
3. Ganglia PHP web front-end – provides a graphical information to the users by using PHP.

### 2.5.2 Cacti

Cacti [Urban, 2011] is an open source software, facilities monitor and graph – CPU load, network bandwidth utilization, network traffic monitor. RRDtool is used in Cacti for making the graph and also for storing polled data. RRDtool has some limitation for making graphs, such as there has no option for 3D, Pie or Scatter chart. RRDtool has the facilities only to make line chart, area chart and combination of both. The feature of the Cacti is included are as follows:

- Unlimited graph items
- Auto-padding support for graph
- Graph data manipulation flexible data sources
- Data gathering on a non-standard time span
- Custom data-gathering scripts
- Build-in SNMP support
- Graph templates
- Data source templates
- Host templates
- Tree, list and preview views of graph data
- User-based management and security

### 2.5.3 Hawkeye

Hawkeye [Hawkeye, 2014] is a distributed monitoring tool, is designed for grid and distributed application. It provides automated detection, it is very flexible, is easy to deploy and also provide timely alerts of problems. Hawkeye has built on Condor technology, such as uses ClassAds and match-making. Hawkeye is used for:

- Monitoring system load, I/O, usage, etc.
- Watching for run-away processes
- Monitoring the pool of a distributed system
- Watching the health of the grid site

Hawkeye architecture consists of four major parts:

- Hawkeye pool – cluster nodes, includes Hawkeye agents and manager
- Hawkeye module – that perform actual node monitoring
- Hawkeye monitoring agent – node monitoring daemons
- Hawkeye manager – node information collector

### 2.5.4 Nagios

Nagios is a powerful network monitoring tool that provides features such as alerting, event handling and reporting [Josephsen, 2013]. The feature of the Nagios is included are as follows:

- Powerful monitoring engine – to provide users with efficient, scalable monitoring of nearly any network infrastructure.
- Updated web interface – provides users with a high level overview of hosts, services and network devices.
- Advanced graphs and visualizations – allows users to quickly view the status.
- Performance and capacity planning graphs – allows organizations to plan for infrastructure upgrades.
- Configuration wizards – configure network devices quickly and easily with UI-based configuration wizards.
- Advanced infrastructure management capabilities – allows users to manage large infrastructures quickly and efficiently.
- Configuration snapshot archive – provides users with an archive of past configuration snapshots.
- Monitoring server integration – to create an IT management system
- Advanced user management – easily setup and manage user accounts
- Service-level agreement (SLA) reports – provide users the ability to measure effectiveness of specific hosts, services and business processes.

**Table 4 - Showing the different monitoring systems comparison**

| Monitoring Systems | Advantages | Disadvantages |
|---|---|---|
| **Gangila** | Scalable architecture, Graphical support, Scalability, Robustness, Reasonable Overhead, Portability | Writing data in XML form requires computationally demanding parsing and cause significant network workload, Complicated system settings |
| **Cacti** | Excellent graphic displays, Web management interface, Manageability, Robustness | Poor extensibility, No notification mechanism, Overhead |
| **Hawkeye** | Notification mechanism, Multiplatform, Possible custom-made sensors, Scalability, Extensibility | Poor front-end, The system is under-developed, Overhead |
| **Nagios** | Excellent extensibility, Notification mechanism, Low overload, Scalability, Robustness, Security, Number of available plug-ins | Can be difficult to setup, Configuration are not user friendly |

Nagios is widely used by big enterprises like Yahoo, Sony, McAfee, DHL, Siemens and Texas Instruments, meaning it is a used, known and feasible tool and it is actively developed. Plug-ins are developed in a regular peace through the open-source community and are free to use to add in Nagios environment. Nagios has a web interface that provides a good administration control, with updated information on every sensor and service configured, being possible to extend it to show graphs and statistics about the information we decide is important to our work.

# 3 Data Mining and Distributed Computing Systems

In this chapter, we briefly introduce Data Mining systems like WEKA, is a collection of machine learning algorithms for data mining tasks. We also discuss briefly the distributed computing systems like Condor and BOINC, how a computational task is distributed through the remote nodes. Actually, distributing a computing task is not a straightforward process, eventually depends on the nature of each computing problem.

## 3.1 WEKA

WEKA is a collection of Machine Learning algorithms for data mining tasks [Bouckaert, 2010], [Witten et al., 2011]. The algorithms can either be applied directly to a data set or called from own Java code. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules and visualization. The WEKA, machine learning / data mining software, is written in Java and is distributed under the GNU Public License. WEKA is used for research, education, and applications. WEKA uses of text files to describe the data. WEKA can work with a wide variety of data files including its own arff format and C4.5 formats [Bouckaert, 2013]. Data can be imported from a file in various formats such as ARFF, CSV, C4.5 binary. Data can also be read from a URL or from an SQL database by using JDBC.

### 3.1.1 ARFF format

For WEKA to analyze data sets, it needs a format so that it understands the structure of the data. The ARFF format is what WEKA uses and is very simple. There are only a few tags to be aware of, which all begin with the @ symbol. A line beginning with the header section of an ARFF file is very simple and merely defines the name of the data set along with the set of attributes and their associated types. The @relation < name > tag is declared at the beginning of the file, where <name > is the name of the relation wish to use. Attributes are defined in the following format:@attribute < attribute_name > < type > [Bouckaert, 2013]. An attribute can be one of four types

Numeric - Can be a real or integer value.

Nominal-specification - Where the value must from a pre-defined set of possible values.

String - Textual values.

Date - For storing dates.

The optional date format argument instructs WEKA on how to parse and print the dates. However, this type will not be very useful for Natural Language Processing (NLP) tasks. Example header: @relation employees @attribute empName string @attribute empSalary numeric @attribute empGender male, female @attribute empDob date "yyyy-MM-DD"

Attributes are case-sensitive, so the label "Name" is different to "name". The second half of an ARFF file is the data itself. The @data tag signifies to WEKA that the instance data is about to commence. Each line after the tag is a set of values (separated by commas) that represent a single instance. It should be obvious that WEKA will expect the order of the values to be in the same order in which the attributes were declared. @data 'AndrewRoberts', 50000, male, 1980-11-09 'PhilSpace', 20000, male, 1976-04-12. String values and nominal values are case sensitive.

One way of using WEKA is to apply a learning method to a data set and analyze its output to extract information about the data. Another is to apply several learners and compare their performance in order to choose one for prediction. The learning methods are called classifiers. They all have the same command-line interface and there is a set of generic command-line options as well as some scheme-specific ones. The performance of all classifiers is measured by a common evaluation module. Implementations of actual learning schemes are the most valuable resource that WEKA provides. But the tools for preprocessing the data are called filters. Like classifiers, filters have a standardized command-line interface and there is a basic set of command-line options that they all have in common. The main focus of WEKA is on classifier and filter algorithms. However, it also includes implementations of algorithms for learning association rules and for clustering data for which no class value is specified.

### 3.1.2 Tools

WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, attribute selection and visualization. Tools or functions in WEKA include [Bouckaert, 2013]:

- Data pre-processing (Data Filters)
- Classification (BayesNet, KNN, c4.5 Decision Tree, Neural Networks, SVM)
- Regression (Linear Regression, Isotonic Regression, SVM for Regression)
- Clustering (Simple K-means, Expectation Maximization (EM))
- Association rules (Apriori Algorithm, Predictive Accuracy, Confirmation Guided)
- Feature Selection (Cfs Subset Evaluation, Information Gain, Chi-squared Statistic)
- Visualization (View different two-dimensional plots of the data)

### 3.1.3 Weka Knowledge Explorer

The Weka Knowledge Explorer [Bouckaert, 2013] is particularly facilitating with graphical user interface. The user can interact with the weka software through the user interface component called Knowledge Explorer. Each of the major weka packages Filters, Classifiers, Clusterers, Associations and Attribute Selection is included inside the Knowledge Explorer along with a Visualization tool which allows datasets and the predictions of Classifiers and Clusterers to be visualized in two dimensions.


**Figure 7 - WEKA graphical user interface**

### 3.1.4 Preprocess Panel

The preprocess panel is the start point for knowledge exploration. From this panel it is allowed to load datasets, browse the characteristics of attributes and apply any combination of Weka's unsupervised filters.

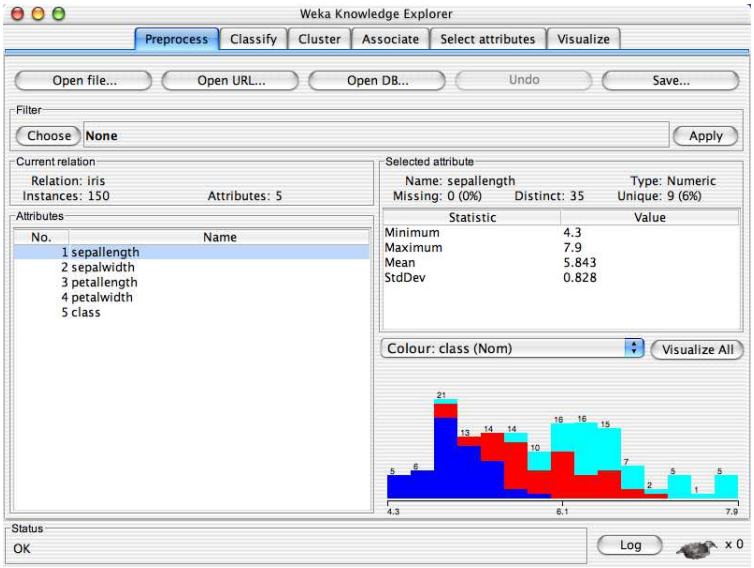



**Figure 8 - Process panel of WEKA knowledge explorer**

### 3.1.5 Classifier Panel

The classifier panel allows configuring and executing any of the weka classifiers on the current dataset. It is allowed to perform a cross validation or test on a separate data set. Classification errors can be visualized in a pop-up data visualization tool. If the classifier produces a decision tree it can be displayed graphically in a pop-up tree visualizer.

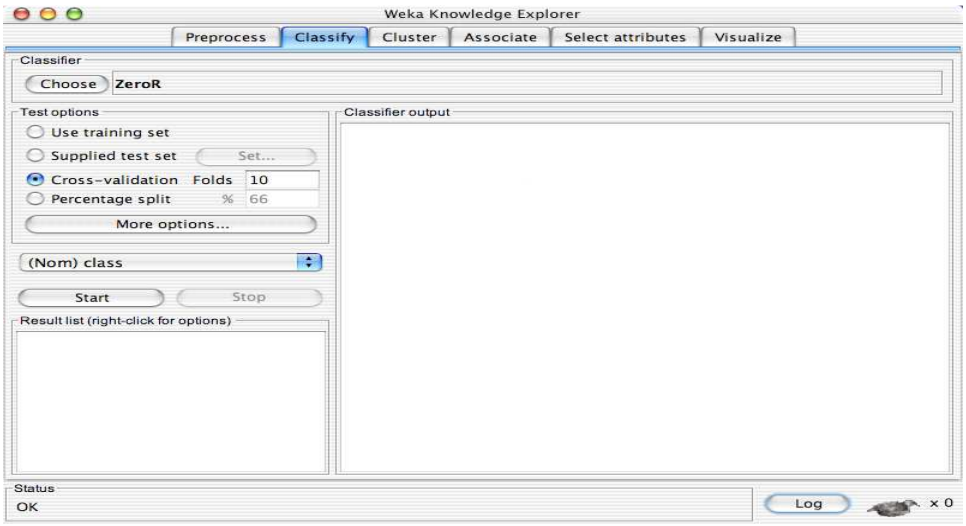



**Figure 9 - Classifier panel of WEKA knowledge explorer**

### 3.1.6 Cluster Panel

From the cluster panel it allows to configure and execute any of the weka clusters on the current dataset. Clusters can be visualized in a pop-up data visualization tool.

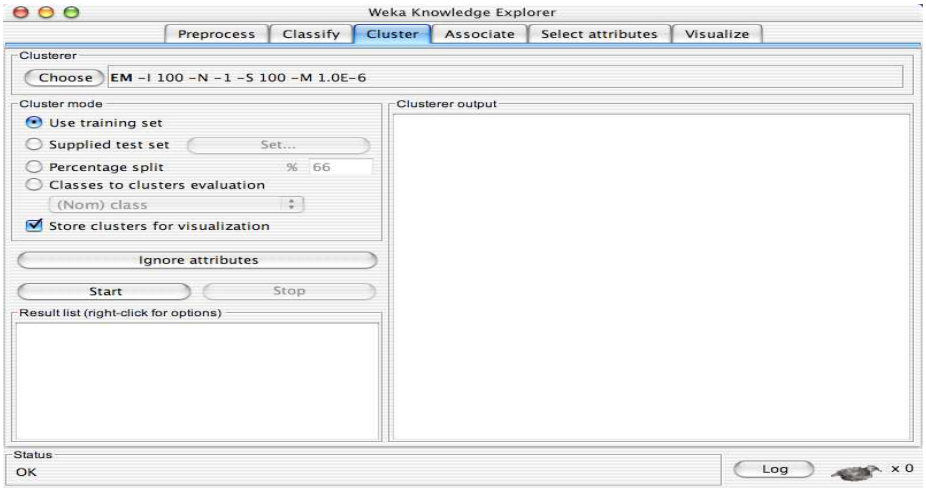



**Figure 10 - Cluster panel of WEKA knowledge explorer**

### 3.1.7 Associate Panel

From the associate panel it allows to mine the current dataset for association rules using the Weka Knowledge Explorer.

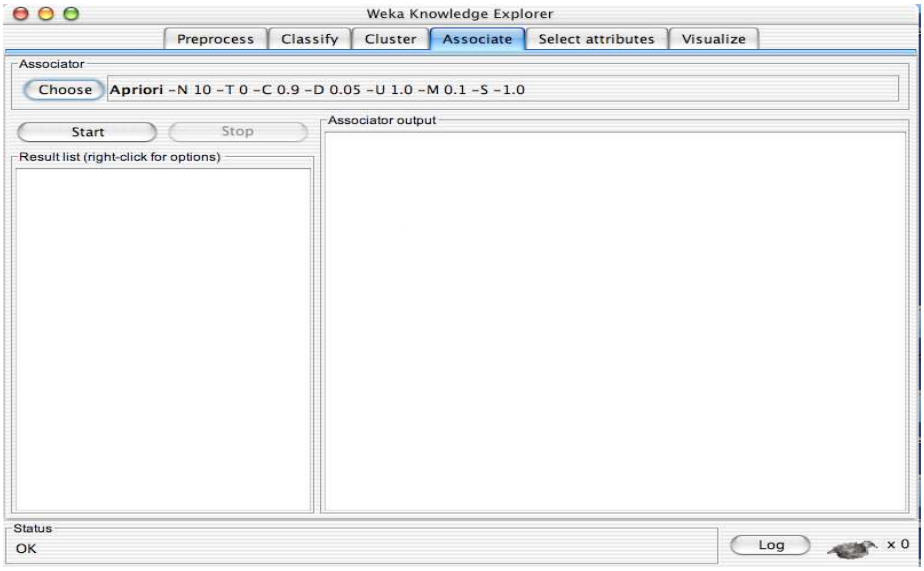



**Figure 11 - Associate panel of WEKA knowledge explorer**

### 3.1.8 Select Attributes Panel

This panel allows configuring and applying any combination of Weka attribute evaluator and searching method to select the most pertinent attributes in the dataset. If an attribute selection scheme transforms the data, then the transformed data can be visualized in a pop-up data visualization tool.

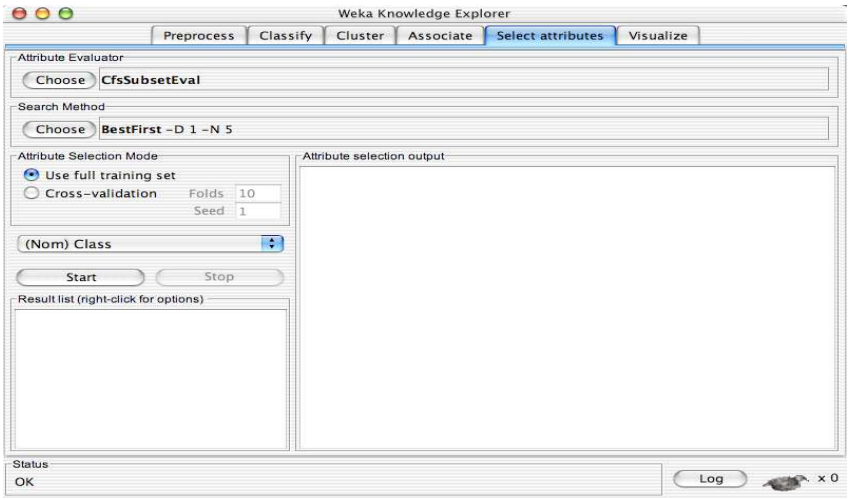



**Figure 12 - Select attributes panel of WEKA knowledge explorer**

### 3.1.9 Visualize Panel

This panel allows to display a scatter plot matrix for the current dataset. The size of the individual cells and the size of the points they display is possible to adjust by using the slider controls at the bottom of the panel. The number of cells in the matrix is changed by pressing the "Select Attributes" button and then can be chosen those attributes to display. When a dataset is very large, plotting performance is possible to improve through displaying only a subsample of the current dataset. This panel allows to visualize the current dataset in one and two dimensions. When the coloring attributes are discrete, each value is displayed as a different color. When the coloring attributes are continuous, a spectrum is used to indicate the value. Attribute bars provide a convenient summary of the discriminating power of the attributes individually. This panel can also be popped up in a separate window from the classifier panel and the cluster panel is allowed to visualize predictions made by classifiers/clusters. When the class is discrete, misclassified points are shown by a box in the color corresponding to the class predicted by the classifier. When the class is continuous, the size of each plotted point varies in proportion to the magnitude of the error made by the classifier.

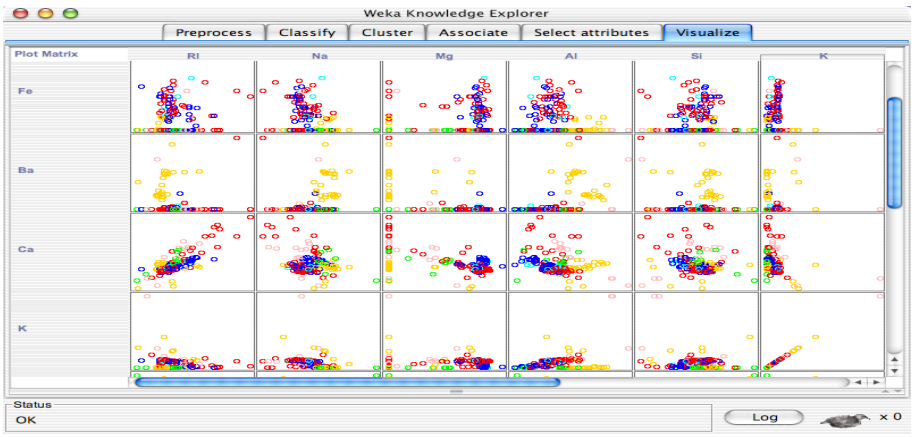



**Figure - Vizualize panel of WEKA knowledge explorer**

### 3.1.10 Knowledge Flow

Knowledge flow is a visual drag and drop GUI for Weka. It has the facilities to provide visual design for the KDD process. Filters, Classifiers, clusters, Associations, loader and savers are included inside in the Knowledge flow along with some extra tools.

The following features are included in the knowledge flow:

- Intuitive data flow style layout

- Process data in batches or incrementally
- Process multiple batches or streams in parallel
- Chain filters together
- View models produced by classifiers for each fold in a cross validation
- Visualize the performance of incremental classifiers during processing
- Plug-in facility for allowing

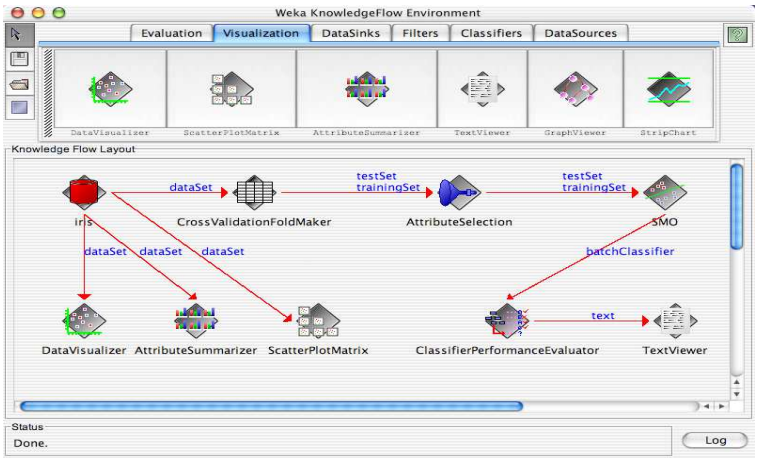



**Figure 13 - Knowledge flow component**

### 3.1.11 An interactive decision tree construction

Weka has a novel interactive decision tree classifier (weka.classifiers.trees.UserClassifier). Through an intuitive, easy to use graphical interface, UserClassifier allows the user to manually construct a decision tree by defining bi-variate splits in the instance space. The structure of the tree can be viewed and revised at any point in the construction phase.

## 3.2 CONDOR

The condor is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to Condor, Condor places them into a queue, chooses when and where to run the jobs based upon a policy, monitors their progress, and informs the user upon completion. Condor effectively utilizes the computing power of workstations that communicate over a network. Condor can manage a dedicated cluster of workstations. Its power comes from the ability to effectively harness non-

dedicated, pre-existing resources under distributed ownership [Mateescu et al. 2011], [Caton et al. 2012], [Mandal et al., 2013].

The condor is one of the leading High-Throughput Computing (HTC) systems in worldwide use.

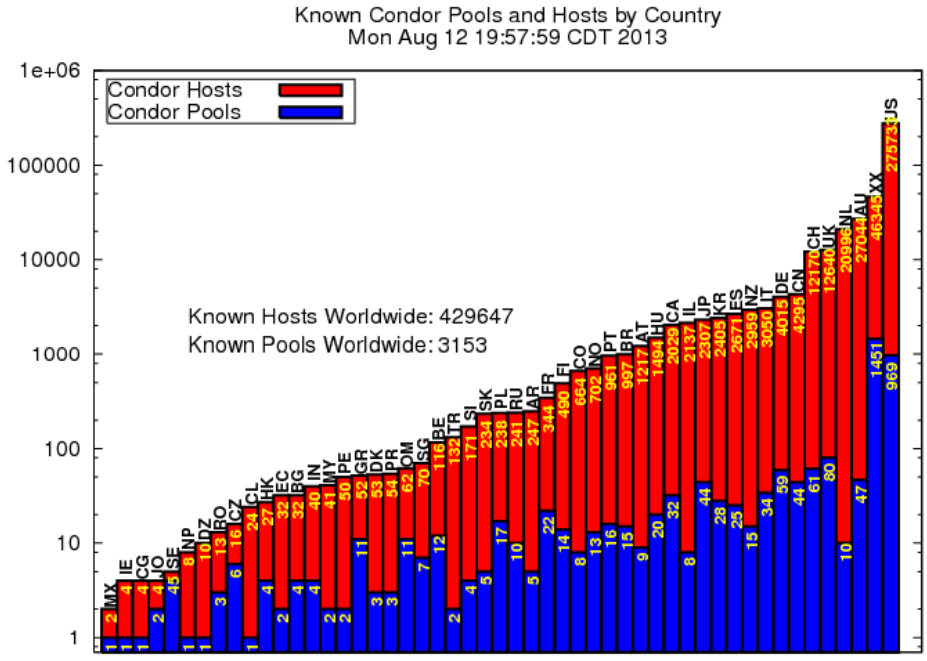



**Figure 14 - Condor pools and hosts by Country [CONDOR, 2014]**

The Condor is the research project under the university of Wisconsin-Madison (UW-Madison). The condor is available through the http://www.cs.wisc.edu/condor. The condor was first installed as a running system at the UW-Madison Department of Computer Sciences in 1988.

As grid computing becomes a reality, Condor is already there [Schwiegelshohn et al., 2010]. The technique of glide-in [Parag et al., 2012] allows jobs submitted to Condor to be executed on grid machines in various locations worldwide. A user submits a job to Condor. Condor finds an available machine on the network and begins running the job on that machine. The condor has the capability to detect that a machine running a Condor job is no longer available (perhaps because the user of the machine came back and start typing or working with the machine). It can checkpoint the job and

move (or migrate) the jobs to a different machine which would otherwise be idle. Condor continues job on the new machine from precisely where it left off. Condor does not require a shared file system across machines, if no shared file system is available, Condor can transfer the jobs data files on behalf of the user, or Condor may be able to transparently redirect all the jobs I/O requests back to the submitting machine. As a result, Condor can be used to combine all of an organization's computational power into one resource. Condor allows almost any application that can run without user interaction to be managed. This is different from systems like SETI@Home and ProteinFolding@Home which are custom written. The Class Ad mechanism in Condor provides a flexible and expensive framework for matching resource requests (jobs) with resource offers (machine). Jobs can state both job requirements and job preferences. Likewise, machines can specify requirements and preferences about the jobs they are willing to run. These requirements and preferences can be described in powerful expressions, resulting in Condor's adoption of the desired policy. Condor can be used to build a Grid-style computing environments that cross administrative boundaries. Condor's Flocking Technology allows multiple Condor compute installations to work together. Condor incorporates many of the emerging grid-based computing methodologies and protocol. For instance, Condor-G [Frey et al., 2002] is fully interoperable with resources managed by the Globus Toolkit [Rani, 2013].

**Main features of Condor**

- Condor can be used for Normal workstation PCs -Clusters with dedicated compute nodes
- Condor supports an impressive list of platforms; Linux (2.0.x, 2.2.x, 2.4.x) - Intel x86 – Windows 2000, XP, 7, 8 - Solaris (2.5.1, 2.6, 2.7, 8) -SPARC - HP Unix 10.20 - PA RISC – Digital Unix 4.0 - Alpha - Irix 6.5 - SGI Mips
- The condor is under an open source license - Condor Public License.
- The source code of a Condor job does not have to be modified.
- PVM and MPI (MPICH) are supported.
- Condor supports check-pointing and job migration.
- Condor offers a flexible resource matching through ClassAds.

- A special Condor version (Condor-G) works together with the Globus Toolkit.
- Linking of different Condor pools is possible through Condor Flocking.

### 3.2.1 Condor Workstation

The Condor Workstation is a normal Condor machine that can submit and serve jobs depending on its configuration.

### 3.2.2 Central Manager

The Central Manager in the Condor pool holds a database about the Condor Workstations and performs the matchmaking between available machines and waiting jobs. Depending on the configuration, the Condor Manager can also submit and serve jobs. In Figure 15 showed how Central Manager works with other Execution and Submission machine.

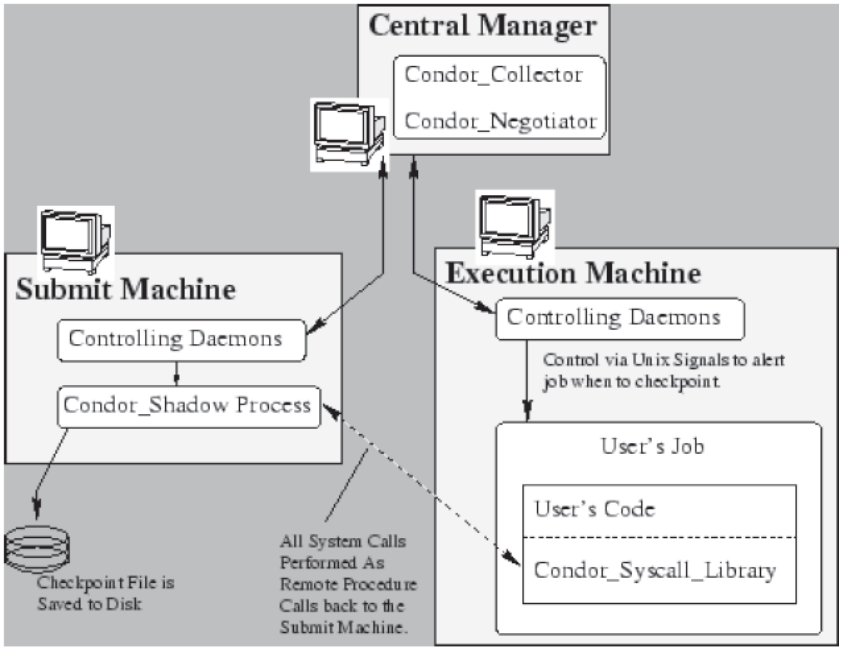



**Figure 15 - CONDOR pool diagram [CONDOR, 2014]**

### 3.2.3 What can be done

In those cases where Condor can checkpoint and migrate a job, Condor makes it easy to maximize the number of machines which can run a job. In this case, there is no requirement for machines to share file systems (for example, with NFS or AFS), so that machines across an entire enterprise can run a job, including machines in different administrative domains. Condor can be a real time saver when a job must be run many (hundreds of) different times, with perhaps hundreds of different data sets. With one command, all of the hundreds of jobs are submitted to Condor. Depending upon the number of machines in the Condor pool, dozens or even hundreds of otherwise idle machines can be running the job at any given moment. Condor does not require an account (login) on machines where it runs a job. Condor can do this because of its remote system call technology, which traps library calls for such operations as reading or writing from disk files. The calls are transmitted over the network to be performed on the machine where the job was submitted.

### 3.2.4 Checkpoint and Migration

Where programs can be linked with Condor libraries, users of Condor may be assured that their jobs will eventually complete, even in the ever-changing environment that Condor uses. As a machine running a job submitted to Condor becomes unavailable, the job can be check pointed. The job may continue after migrating to another machine. Condor periodic checkpoint features periodically checkpoints a job even in lieu of migration in order to safeguard the accumulated computation time on a job from being lost in the event of a system failure such as the machine being shutdown or a crash.

### 3.2.5 Remote System Calls

Despite running jobs on remote machines, the Condor standard universe execution mode preserves the local execution environment via remote system calls. Users do not have to worry about making data files available to remote workstations or even obtaining a login account on remote workstations before Condor executes their programs there. The program behaves under Condor as if it were running as the user that submitted the job on the workstation where it was originally submitted, no matter on which machine it really ends up executing on.

### 3.2.6 No Changes Necessary to Users Source Code

No special programming is required to use Condor. The condor is able to run non-interactive programs. The checkpoint and migration of programs by Condor is transparent and automatic, as is the use of remote system calls. If these facilities are desired, the user only re-links the program. The code is neither recompiled nor changed.

### 3.2.7 Pools of Machines can be Hooked Together

Flocking is a feature of Condor that allows jobs submitted within a first pool of Condor machines to execute on a second pool. The mechanism is flexible, following requests from the job submission, while allowing the second pool, or a subset of machines within the second pool to set policies over the conditions under which jobs are executed.

### 3.2.8 Jobs can be Ordered

The ordering of job execution required by dependencies among jobs in a set is easily handled. The set of jobs is specified using a directed acyclic graph, where each job is a node in the graph. Jobs are submitted to Condor following the dependencies given by the graph.

### 3.2.9 Condor Enables Grid Computing

As grid computing becomes a reality, Condor is already there. Glidenin is a mechanism by which one or more remote machine joins in a local Condor pool. The technique of glidein allows jobs submitted to Condor to be executed on grid machines Condor in various locations worldwide. As the details of grid computing evolve, so does Condor's ability, starting with Globus-controlled resources.

### 3.2.10 Sensitive to the Desires of Machine Owners

The owner of a machine has complete priority over the use of the machine. An owner is generally happy to let others compute on the machine while it is idle, but wants it back promptly upon returning. The owner does not want to take special action to regain control. Condor handles this automatically.

### 3.2.11 Class Ads

The Class Ad mechanism in Condor provides an extremely flexible, expressive framework for matchmaking resource requests with resource offers. Users can easily request both job requirements and job desires. For example, a user can require that a job run on a machine with 64 Mbytes of RAM, but state a preference for 128 Mbytes, if available. A workstation owner can state a preference that the workstation runs jobs from a specified set of users. The owner can also require that there is no interactive workstation activity detectable at certain hours before Condor could start a job. Job requirements/preferences and resource availability constraints can be described in terms of powerful expressions, resulting in Condors adaptable to nearly any desired policy.

### 3.2.12 CONDOR Daemon and Work Flow Diagram

The following Figure 16 showing the Condor daemon work flow diagram

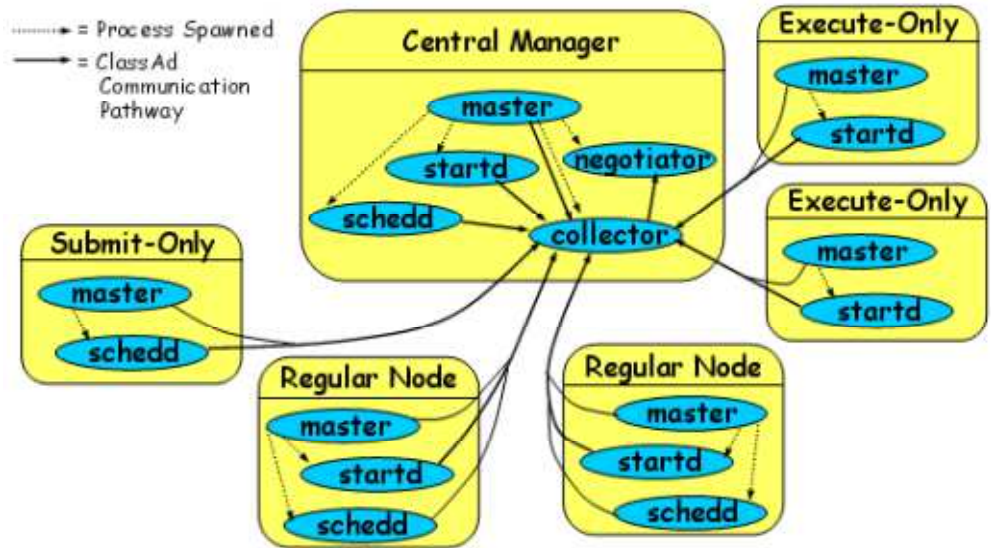



**Figure 16 - CONDOR daemon work flow diagram [CONDOR, 2014]**

### 3.2.13 Condor master

This program runs constantly and ensures that all other parts of Condor are running. If they hang or crash, it restarts them.

### 3.2.14 Condor collector

This program is part of the Condor central manager. It collects information about all computers in the pool as well as which users want to run jobs. It is what normally responds to the condor status command.

### 3.2.15 Condor negotiator

This program is part of the Condor central manager. It decides what jobs should run and where.

### 3.2.16 Condor startd

If this program is running, it allows jobs to be started up on the computer-that is, the computer act as an "execute machine". This advertises the computer to the central manager so that it knows about this computer. It will start up the jobs that run.

### 3.2.17 Condor schedd

If this program is running, it allows jobs to be submitted from this computer-that is, our computer is a "submit machine". This will advertise jobs to the central manager so that it knows about them. It will contact a condor startd on other execute machines for each job that needs to be started.

### 3.2.18 Condor shadow

For each job that has been submitted from this computer, there is one condor shadow running. It will watch over the job as it runs remotely. In some cases, it will provide some assistance.

## 3.3 BOINC

BOINC [Ries et al., 2012] stands for Berkeley Open Infrastructure for Network Computing. BOINC is an open source software platform to allow distributed computing projects, which use volunteer computer resources to run. Volunteer computing platforms such as BOINC are currently the most successful Distributed computing systems, which rely on donated computer cycles from ordinary

citizen communities. BOINC is currently being successfully used by many projects to analyze data. BOINC is open source and is available at http://boinc.berkeley.edu.

BOINC is a client server system that takes care of the most of background tasks involved in a Distributed Computing project. The project owners first create a BOINC compatible science application to process their project's work units. Then load the scientific application and work units onto their BOINC server for distribution to the project participants. The project participants install the BOINC client on their computers, and then attach to the project they are interested in running. Once this happens, the BOINC client program, then downloads the scientific application and work units from the BOINC servers. The BOINC client then starts the science application to process the work units. When done, the BOINC client uploads the completed work unit to the BOINC server, which then verifies the work done and grants credit to the participant.

### 3.3.1 The BOINC system

BOINC follows a simple network protocol, showed in Figure 17, which requires clients to initiate all the communications (because of NAT/firewall problems), and to contact the server every time a client requires more work [Ries et al., 2011].

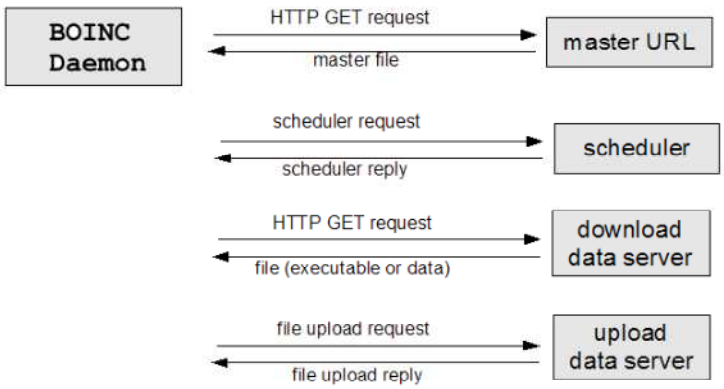



**Figure 17 - BOINC network protocol Overview [Korpela, 2012]**

Every time a client is idle and wishes to execute more work, it has to contact the main server 3 times (the first connection to get the master file is normally only needed once, when the client first attaches to a project), following these steps:

Contact the scheduler to ask for a new result to execute. The scheduler reply contains information about the result: input files and executables the client needs to download and URLs of data servers where it should download and upload files from.

Download the input files and executables from data server referred by the scheduler.

Upload the output files to data servers indicated by the scheduler.

BOINC's architecture, showed in Figure 18 below, along with the protocol describe above, shows that there is a heavy dependence on each project's central server.

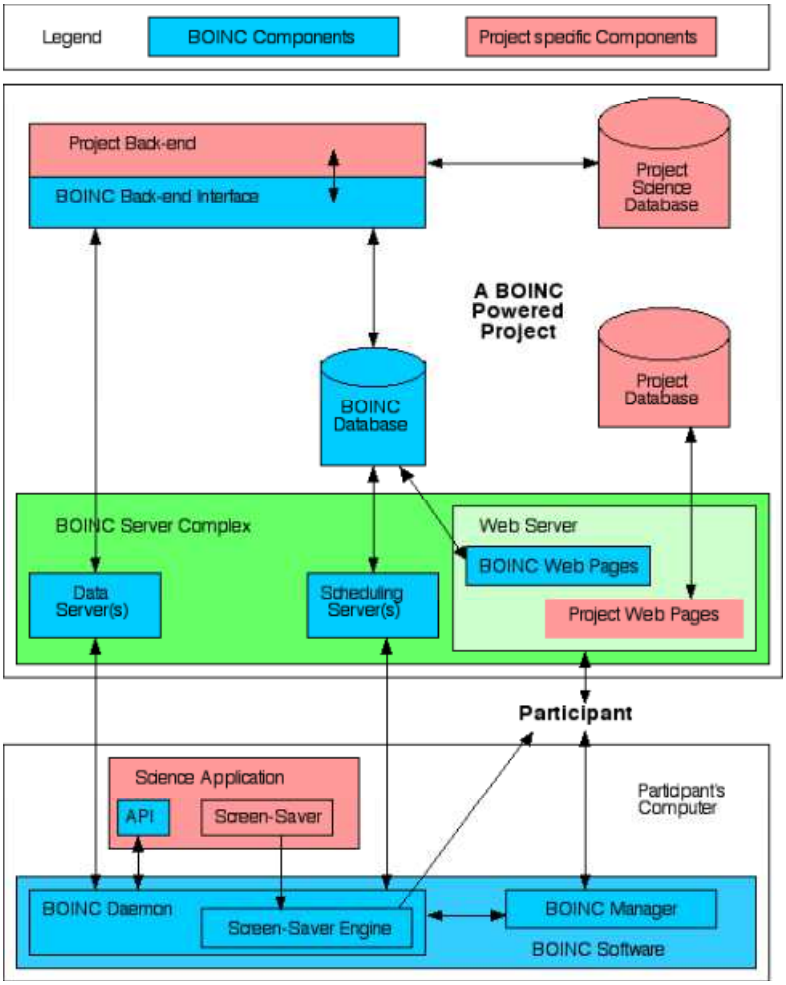



**Figure 18 - BOINC Project-components [Korpela, 2012]**

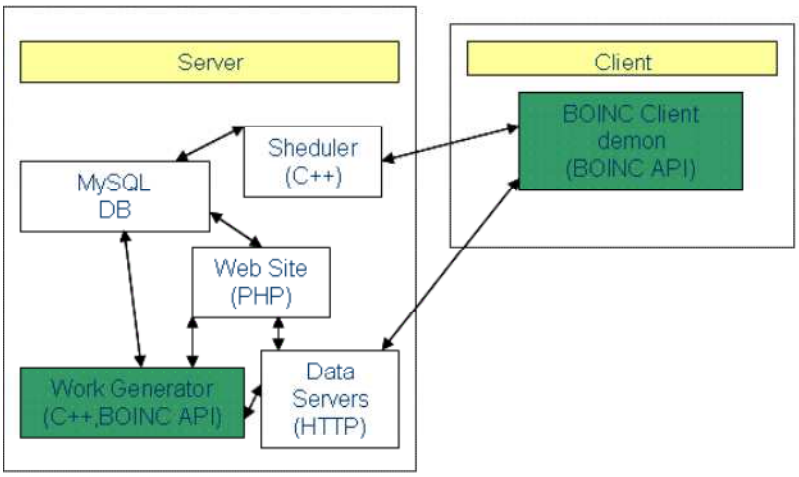



**Figure 19 - BOINC interfaces Server and Client [Korpela, 2012]**

There are two interfaces between the server and each client: the scheduler and the data server. Figure 19 shows the organization of the main server, with a central coordinating point, the database. The BOINC platform follows a simple model: each project runs a central server that runs a Master application. The application is divided into thousands of smaller tasks that are sent to machines spread over the Internet, where they execute the Worker type application. There are no communications between Workers. All communication must be from the Worker to the Master, to enable the traversal of NATs and firewalls. The heavy dependence on the MySQL database of all the other components like transitional, assimilator, etc., Limit the decentralization of the main server for the distribution of the DB bring more problems (security, reliability, intrusiveness) than advantages.

### 3.3.2 The BOINC Server

The BOINC server consists of at least one web server that handles up downloads and a database server that keeps track of the state of the Work Units (WU) and their associated results. Furthermore, five different daemons periodically check the state of the database and perform any needed tasks within their area of responsibility. All these programs can be run on the same machine or they can be distributed to different servers for performance reasons. This was done for the SETI@home project [Korpela, 2012], [Javadi et al., 2011]. The server runs on Linux and Solaris, but being open source other platforms are a possibility. In below, Figure 20, is showing how the computing proceeds in a BOINC-based system.

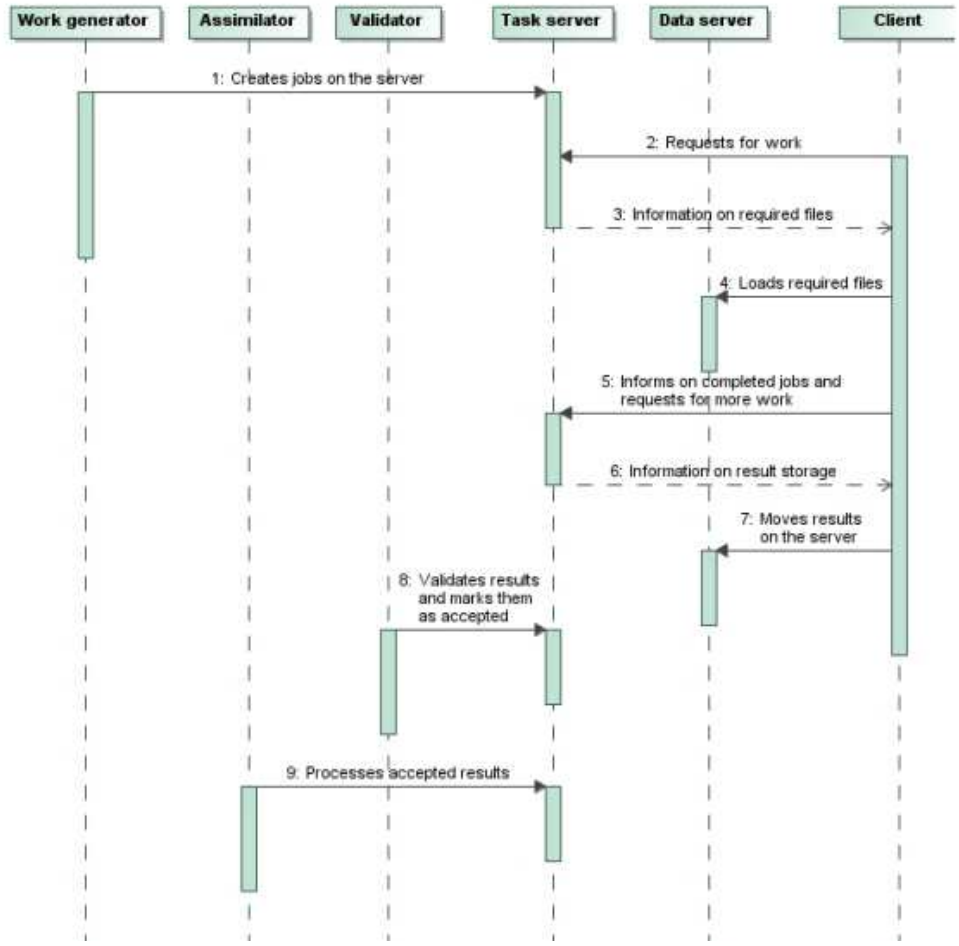



**Figure 20 - The process of BOINC-based system [Korpela, 2012]**

Below details of the five daemons and a few other components that make up the BOINC server.

### 3.3.3 The Work generator

There is one work generator per application. It creates work units and the corresponding input files. It is application-specific and uses BOINC library functions for registering the work units in the database. During testing one can create, single work units by using create work.

### 3.3.4 The Transitioner

This program is application independent. It handles the state transition of WUs and results. The transitioner checks the state of a WU in the database and updates the appropriate state fields in the database, once a WU is ready for a state transition. This is complicated by the fact that a WU does

not have an overall state, but a set of sub states. These sub states involve the states of its associated results. For instance, if they are ready for validation, and there is enough results ready for validation to grant a quorum the state of the WU is changed to "ready for validation". The transitioner generates the initial results and generates more if timeouts or errors occur. It is one of the most CPU intensive programs, it can therefore be split into many processes that each has the responsibility for a subset of WUs. These processes can then be distributed to different servers.

### 3.3.5 The Validator

The purpose of this daemon is to verify whether the returned results are valid or not. It is part of the project back end shown on Figure 20. When run, it checks the database to see if there are any new results uploaded that need validation. If there is, the validator then runs an application specific function for comparing the results. The implementer of the validator functions has to specify two functions: one that compares the two results and one that compares a set of results. The first is used to decide whether to grant credit when a new result is uploaded and a canonical result has already been found. The second is used to decide on a canonical result from a set of results. The implementer might, for example, consider results that agree to the third decimal as good enough, but require that at least four results be within this range. Furthermore, it allows the user to perform further sanity checks on the results. If we, for instance, were simulating gravity on a body dropping freely in the atmosphere, we could check if it ended up higher than it started. This should never happen so we would not accept the result. Whether the cause of this occurrence is errors in the client or in the simulation program, we would never want such a result to be validated. When a WU is created it is specified how many agreeing results are needed for that particular WU. This value could have been set on an application basis, but since different people with different needs could potentially be submitting WUs for the same application, the method chosen allows these different people to set different values. It is also possible that WUs the same person has different needs. Once there are enough agreeing results a canonical result for the WU is chosen from the set of agreeing results. The WU is hereafter aged in the database as ready for assimilation.

### 3.3.6 The Assimilator

The Assimilator regularly checks if new WUs are ready for assimilation. It is like the validator as part of the project backend depicted in Figure 20. The administrator must supply a function that

determines what is to be done with the canonical results. It could, for example, zip up the result files and email it to the person interested in it, or automatically do post processing on the data extracting the interesting parts and storing it on a magnetic tape storage device. Once the WU has been assimilated, the WU will be tagged as completed.

### 3.3.7 The File Deleter

The file deleter daemon simply checks for completing and assimilated WUs accumulated since its last run. If so, it cleans the web server of input and output files related to these WUs. It is therefore essential that the canonical result's output files are copied somewhere safe in the assimilation phase, if they are needed. It only deletes files not database records, so it is always possible to go through the database and find information on WUs, results etc. even after their completion (and file deletion).

### 3.3.8 The Feeder

The feeder loads undispatched (unsent in BOINC terms) results from the database into a shared memory segment. This prefetching of results is done to improve performance of the entire BOINC system by limiting the number of queries on the database. Especially the performance of the scheduler is improved, when dispatching results to clients it only needs to access the shared memory segment, which contains many more results than needed for one client, instead of doing a query on the database. The results loaded into the memory segment were randomly chosen in the early versions of BOINC to prevent cheating as it would make it virtually impossible to have control of say 5 clients and have them connect in rapid succession to get all results associated with a specific WU. If one user got all results for a specific WU, he could easily fool the validator. Because of performance problems the new versions of BOINC load results on a first come, first served basis, meaning results are loaded in the order their respective WUs were submitted.

### 3.3.9 The Scheduler

The scheduler is a CGI program that is run whenever a client connects to the project and asks for work. It is shown as the scheduling server in Figure 20. Instead of querying the database, it gets work from the shared memory segment loaded by the feeder. The scheduler matches results with clients,

since not all clients are identical and certain user settings differ. For instance, one user could be running a Linux version of the core client and have specified that the core client can only use 10 MB of disk space and 20 MB of RAM. While another user could be running the windows version and allowed it to use 100 MB of disk space and 100 MB of RAM. These users are clearly able to process different results. During a scheduling session, the core client also reports any finished results, which have already been uploaded, since the last scheduled session. After a scheduling session, the core client is left with a list of results to process and a list of URLs from which to get the needed files, i.e. input files and the application files, if not already on the host machine.

### 3.3.10 The Database

A MySQL database stores all information relevant to the BOINC server complex. This includes information about registered users and their associated hosts, about applications and application versions, about BOINC core clients and the versions hereof and of course about WUs and their associated results. The entire state of the server complex is stored in this database and queried by among others the above-mentioned daemons.

### 3.3.11 The BOINC Client

The client is available for Windows and Linux on Intel X86 architectures, for Mac OS X on PowerPC, and Solaris on SPARC architectures, but since it is open source it should not be too difficult to get it running on other platforms as well. A user who wants to share his computer's spare cycles to a BOINC project downloads and installs the BOINC client, also known as the core client. This client can be found at the website of every BOINC project and it is therefore usually downloaded from the first project the user signs up for. After installation of the client software, the user visits a project website and registers as a user. Shortly after having registered, the user receives an email with a unique user id. This id together with the project's URL is then entered by the user to sign his BOINC client up for the specific project. This process is called attaching the client with a project and the reverse process is called detaching. The client is attachable to many projects at the same time and allows the user to specify how large a percentage of time on average should be spent on each project. The user has a lot of control over the client via user preferences. Some of these preferences are project specific and some are not. Among other things, the user can specify whether the core client should run jobs

while the user is actively using the computer or only use it after a specified period of time without user activity.

### 3.3.12 The BOINC Applications

To get any work done by BOINC, projects must implement at least one BOINC application. Once the application has been implemented in the form of an executable, it must be registered with the BOINC server and the administrator can start creating WUs for this application. This means that WUs are nothing more than a specification of input files and command line switches to an already registered application.

If the project wants to run their computations on different platforms, a version of the specific application for each desired platform must be implemented and registered. WUs are not bound to a specific platform nor are they bound to a specific version of the application; the latest version of the appropriate platform is simply used. In order for an application to be a BOINC application, it must call certain functions from the BOINC API, which is implemented in C. BOINC applications, must call a BOINC initialization function at the beginning and a BOINC finishing function at the end. Besides these two functions, BOINC applications are encouraged to call a function, communicating the progress in percent to the core client, so that users can see the progress on their screens. Users also expect some sort of graphics to be present so BOINC applications should implement a specific function for drawing graphics, which will be called by the core client, if the user requests that the graphics be displayed.

If the application uses input files or non-temporary output files, which of course most do, the file names must be translated before opening the files. This is done by calling a function from the BOINC API and it is done because each application runs on many different input files and produces many different output files. If each of these files from different runs had the same names, each run would require a separate directory on the server and on the client to avoid names clashing. Since the application is the same between runs, it is not possible to change the file names internally in the application between runs and it would be very impractical to recompile it for every run.

All this means that the application has logical file names and these file names are translated by the core client to physical file names at run time. The physical file names are specified when the WU is

created. Check pointing can be a big advantage, but in order to support it the application must call a BOINC API function, that tells the application if it is okay to checkpoint now. This is because the user can instruct the client to only use the hard drives on the computer at certain intervals to avoid spinning up the drives frequently (on laptops for instance). If the core client allows check pointing, the application must then do all the hard work of check pointing by itself and then call another BOINC API function to tell the core client it has finished check pointing. This has to do with the fact that the core client records the CPU time spent on a result to calculate the credits. If a result is restarted, the CPU time begins counting from the time spent at the last checkpoint instead of zero.

### 3.3.13 Choosing a Server

BOINC applications can run on Windows, MacOS X, and Unix (generally Linux, but also Solaris and HPUX), but the server must be a Unix computer, generally running on Linux. Any Linux distribution will work, so long as it has the necessary software installed inside.

There are three ways of setting up a server correctly to host a BOINC project.

- First is to use Linux distribution, which very easily includes all the requisite software.
- Second is to specifically download, build and install the necessary software (or download and install binaries or packages).
- Third, one is to download and run a Linux Virtual Machine for BOINC using the VMware Player. VMware approach is a bit difficult to sorting out the problem. For this work, we have used Linux LAMP server distributions.

## 3.4 Monitoring system

When designing a cluster - monitoring system, it is conspicuous to require consideration rigorously concerning the cluster type to be used and its categorical features. During this work, the Nagios monitoring system has taken into account the foremost congruous for this type of monitoring. Nagios is a popular and puissant open source IT infrastructure monitoring software application. Nagios is incredibly stable and ready to monitor the astronomically immense spread infrastructures with thousand of devices. Nagios implementation is for monitoring availability services and amassing performance information from the infrastructure. It sanctions to trace state and find an abundance

of detail information what is happening to the services. With the automated checking of the device and repair standing and causation error report, Nagios give for more expeditious resolution of issues.

### 3.4.1 Environment Configuration

Here we have described how to install and configure the tools that we have needed for our network to achieve successfully. On our server, we have installed the Grid Computing System HTCondor, which is responsible to send the jobs we want to the PCs idled on the network. To monitor the server and its applications, there is a monitoring PC with Nagios installed, which is connected to the server through SSH. The image bellow exemplifies this.

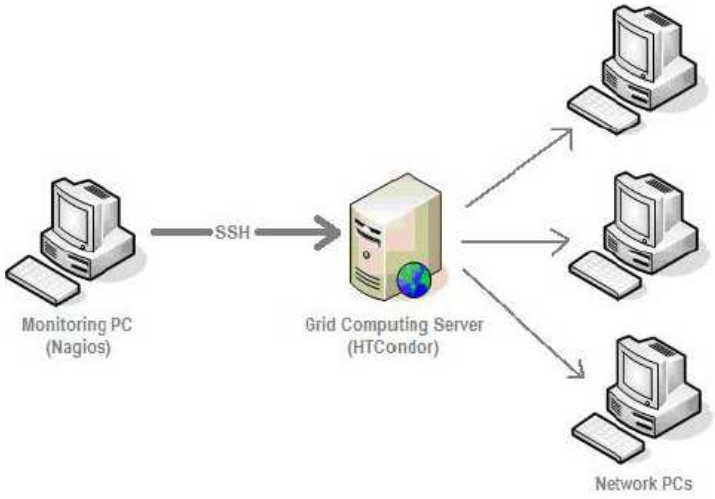



**Figure 21 - The network setup for Monitoring PC**

### 3.4.2 Openssh Server

We have installed Openssh on our Ubuntu HTCondor server and on the client PC we have installed Putty in case we had installed Nagios on the same server as HTCondor, being possible to connect to it securely through SSH. We have chosen SSH mainly because it is more secure than NRPE. In our environment, security is a very important aspect, because it deals with lots of confidential and important data. Even though the CPU overhead with SSH is larger, we believe it is not negatively worse than the other options. Another advantage of using SSH is the configuration; it imposes when compared with NRPE, for example.

### 3.4.3 Nagios

We have used Nagios-3 [Josephsen, 2013] on the Monitor PC and the configurations (services and commands). We have configured commands to check if SSH that was enabled on the server. Moreover, we have checked through SSH service that the HTCondor was running. Figure 22 shows that the Nagios is running through SSH service on the server.

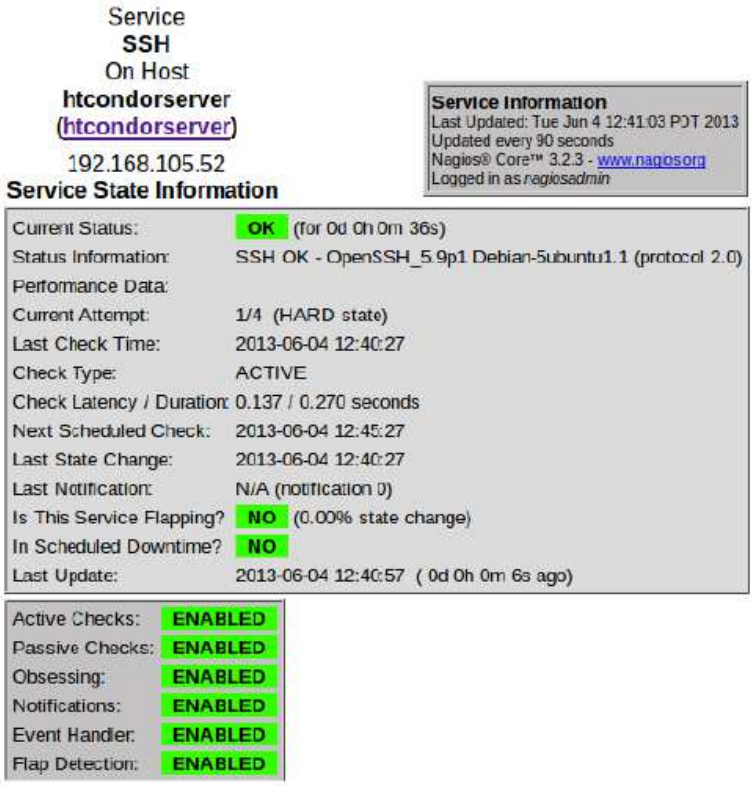



**Figure 22 - SSH service is running on htcondorserver**

In this work we have successfully identified, installed, configured, and managed Nagios for the distributed computing systems.

# 4 Implementation

In this chapter, we have implemented data mining tasks through WEKA, in the distributed computing systems and we have discussed the output results briefly.

## 4.1 HTCondor

We have installed HTCondor on Ubuntu machine. Ubuntu 12.04 is used to install HTCondor, a complete one-machine pool. The details of installation and configuration of HTCondor are described in the appendix A. The first outlook of the HTCondor system is to observe whether the condor_master daemon is running or not in the condor master PC. In the Condor master PC we have executed the ps command to see the current status of the Condor. The output found from the ps command in the HTCondor master PC as shown in the Table 5, and as long as it lists all of those Condor programs, we assume that the HTCondor is running well. HTCondor master PC running the condor_master that spawns all the other condor daemons and monitors them, restarting them if necessary.

| PID | TTY | STAT | TIME | COMMAND |
|---|---|---|---|---|
| 15652 | ? | S | 0:00 | sshd:anis@pts/0 |
| 15654 | pts/0 | S | 0:00 | -bash |
| 15824 | ? | S | 0:00 | condor_master |
| 15825 | ? | S | 0:00 | condor_collector –f |
| 15826 | ? | S | 0:00 | condor_negotiator –f |
| 15827 | ? | S | 0:00 | condor_schedd –f |
| 15828 | ? | S | 0:00 | condor_started –f |
| 15846 | Pts/0 | R | 0:00 | ps –x |

**Table 5 - Output from the running Condor master**

| PID | TTY | STAT | TIME | COMMAND |
|---|---|---|---|---|
| **15652** | ? | S | 0:00 | Sshd: anis@pts/0 |
| **15654** | pts/0 | S | 0:00 | -bash |
| **15824** | ? | S | 0:00 | Condor_master |
| **15825** | ? | S | 0:00 | Condor_collector –f |
| **15826** | ? | S | 0:00 | Condor_negotiator –f |
| **15827** | ? | S | 0:00 | Condor_schedd –f |
| **15828** | ? | S | 0:00 | Condor_started –f |
| **15846** | Pts/0 | R | 0:00 | ps –x |

The HTCondor master Ubuntu machine is dedicated to run as a Condor master PC and as a job submitter and executable machine in the pool. HTCondor is using ClassAd to match jobs with systems. If the submitted job requirements like 512MB RAM, HVM CPU etc. ClassADs are used to fulfill that job requirements and are searching for that preference in the pool. When a job is submitted to be executed on a system, job priorities play an important role in the Condor system. The system gives a higher priority to a job based on who owns the job or what groups they belong to. Once the job is complete, condor systems, is no longer running a job and so is available to potentially run another job.

### 4.1.1 Dataset

We have chosen Internet traffic dataset to make some experiments by executing WEKA knowledge discovery tasks over Condor. Internet traffic is playing an important role in network management. It enables network operators with information of various management activities, like capacity planning and provisioning, traffic engineering, fault diagnosis, application performance, anomaly detection and pricing. In this work to identify Internet traffic, we have used a supervised classification system based on the Naïve Bayes method implemented by using Weka through the Condor system.

The data set is used in this work is collected from Andrew W. Moore and Denis Zuev [Andrew and Denis, 2014]. We have downloaded the data set in the arff format from the website of the Computer Laboratory Research Centre of University of Cambridge. The dataset is available on the web URL http://www.cl.cam.ac.uk/research/srg/netos/nprobe/data/papers/sigmetrics/. The flows were recorded in arff format to form the dataset. The experiment is conducted using dataset entry12.weka.allclass.arff through cross validation. The data set is fed into Naïve Bayes classifier to train the model and then test and evaluate the performance. We have used 10-fold cross-validation method which means that all data from the dataset is selected in the training data and testing set consecutively. The dataset is broken down into 10 sets of size n/10 and train on 9 datasets and test on 1 dataset and repeat up to 10 times and take a mean accuracy [Labovitz, 2011], [Zhang, 2013]. We have fed the same data set into the J48 classifiers and as well as for the K means clustering algorithm.

### 4.1.2 Submitting Data mining tasks to the Condor

HTCondor has the facility to submit a job through a script file. The script file is mentioned in the appendix for the further explanation of the work. In the weka script file we have mentioned the universe = java, output = weka.output, error = weka.error, log = weka.log file, should_transfer_files = yes, when_to_transfer_output = on_exit, executable = usr/share/java/weka.jar, jar_files = usr/share/java/weka.jar, transfer_input_files =/home/anishur/ internettraffic.arff, arguments = weka.classifiers.bays.NaiveBays –t /home/anishur/internettraffic.arff and finally end the script file with, queue. The submission to the Condor systems is done through the condor_submission weka.condor command. Whereas, weka. condor is the weka script file and condor_submission is the condor command is used to submit a script file into the Condor system.

### 4.1.3 Output from the HTCondor using Naïve Bayes Classifier

HTCondor produces the results through weka.output, weka.log and weka.error. Weka.output contains the output results from the Weka. The following results show the Naïve Bayes Classifier.

Naive Bayes Classifier
---
Time taken to build the model: 22.94 seconds
Time taken to test the model on training data: 111.33 seconds

=== Error on training data ===

The correctly and incorrectly classified instances are defined as the case where the instances are used as test data.

The Kappa statistic is the measure of agreement normalized for chance agreement.

Where, Kappa, K = P (A) – P (E) /1 - P (E); whereas, P (A) is the percentage agreement, P (E) is the chance agreement. K = 1 is indicating the perfect agreement and where as K = 0 is indicating the chance agreement.

The mean absolute error is the sum of all the instances and their absolute error per instance is divided by the number of instances in the test set with an actual class label.

Mean Abs Error = Sum (Abs Error Per Instance)

The root relative squared error is computed by dividing the root mean squared error by the root mean square error obtained by just predicting the mean of target values and then multiplying by 100.

**Table 6 - Summary of the results of Error on training data**

| | | |
|---|---|---|
| **Correctly Classified Instances** | **18293** | **93.208%** |
| **Incorrectly Classified Instances** | 1333 | 6.792% |
| **Kappa statistic** | 0.8152 | |
| **Mean absolute error** | 0.0151 | |
| **Root mean squared error** | 0.1216 | |
| **Relative absolute error** | 18.9433% | |
| **Root relative squared error** | 61.0419% | |
| **Total Number of Instances** | 19626 | |

=== Confusion Matrix ===

**Table 7 - Confusion matrix**

| a | b | c | d | e | f | g | h | i | Classified as |
|---|---|---|---|---|---|---|---|---|---|
| **15158** | 11 | 0 | 10 | 0 | 83 | 0 | 0 | 335 | a = WWW |
| **8** | 681 | 0 | 0 | 1 | 0 | 0 | 4 | 1 | b = FTP-PASV |
| **0** | 0 | 120 | 0 | 0 | 0 | 0 | 1 | 0 | c = SERVICES |
| **0** | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | d = INTERACTIVE |
| **227** | 0 | 0 | 0 | 980 | 341 | 8 | 228 | 15 | e = MAIL |
| **1** | 0 | 0 | 1 | 3 | 227 | 0 | 7 | 0 | f = FTP-CONTROL |
| **2** | 0 | 0 | 0 | 0 | 0 | 287 | 6 | 0 | g = DATABASE |
| **11** | 0 | 0 | 0 | 1 | 0 | 0 | 293 | 0 | h = P2P |
| **4** | 4 | 2 | 0 | 2 | 0 | 0 | 8 | 509 | i = FTP-DATA |

=== Stratified cross-validation ===

**Table 8 - Stratified cross-validation**

| | | |
|---|---|---|
| **Correctly Classified Instances** | **18311** | **93.2997%** |
| **Incorrectly Classified Instances** | 1315 | 6.7003% |
| **Kappa statistic** | 0.8178 | |
| **Mean absolute error** | 0.0149 | |
| **Root mean squared error** | 0.1208 | |
| **Relative absolute error** | 18.7334% | |
| **Root relative squared error** | 60.639% | |
| **Total Number of Instances** | 19626 | |

=== Confusion Matrix ===

**Table 9 - Confusion matrix**

| a | b | c | d | e | f | g | h | i | Classified as |
|---|---|---|---|---|---|---|---|---|---|
| **15161** | 9 | 0 | 10 | 78 | 0 | 0 | 339 | 0 | a = WWW |
| **8** | 681 | 0 | 0 | 1 | 0 | 0 | 4 | 1 | b = FTP-PASV |
| **2** | 0 | 118 | 0 | 0 | 0 | 0 | 1 | 0 | c = SERVICES |
| **0** | 2 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | d = INTERACTIVE |
| **216** | 0 | 0 | 0 | 1004 | 332 | 7 | 224 | 16 | e = MAIL |
| **1** | 0 | 0 | 0 | 4 | 227 | 0 | 7 | 0 | f = FTP-CONTROL |
| **2** | 0 | 0 | 0 | 0 | 0 | 285 | 8 | 0 | g = DATABASE |
| **13** | 0 | 0 | 1 | 4 | 0 | 2 | 276 | 1 | h = P2P |
| **4** | 4 | 2 | 0 | 2 | 0 | 0 | 9 | 508 | i = FTP-DATA |

### 4.1.4 Results from standalone windows PC using the Naïve Bayes classifier

In this work we have used a standalone windows PC to check and validate our results that have found from HTCondor as an output. The following results have found as an output for a Naïve Bayes classifier by using a 10-fold cross-validation.

Run information:

Scheme : weka.classifier.bayes.NaiveBayes

Relation: filelist.weka.allclass.csv

Instances: 19626

Attributes: 249

Test mode: 10-fold cross-validation

Naïve Bayes Classifier

Time is taken to build the model: 0.61 seconds

**Table 10 - Summary of the results**

| | | |
|---|---|---|
| **Correctly Classified Instances** | **18311** | **93.2997%** |
| **Incorrectly Classified Instances** | 1315 | 6.7003% |
| **Kappa statistic** | 0.8178 | |
| **Mean absolute error** | 0.0149 | |
| **Root mean squared error** | 0.1208 | |
| **Relative absolute error** | 18.7334% | |
| **Root relative squared error** | 60.639% | |
| **Total Number of Instances** | 19626 | |

### 4.1.5 Details on the accuracy by class

The following results are showing the details accuracy by class and 10-fold cross validation is conducted on the entry12.weka.allclass.arff dataset.

All the parameters are defined as follows:

- The True Positive (TP) rate is the proportion of positive cases which were correctly identified.
- The False Positive (FP) rate is the proportion of the negative cases which were incorrectly classified as positive.
- The Precision is the proportion of the truly positive which were correct.
    - Precision = TP/(TP+FP)*100%
- The Recall = TP/(TP+FN)*100%

- The F-Measure = 2* Precision* Recall/( Precision + Recall); this is a combined measure of Precision and Recall.

- Weighted Avg. – is weighting the results based on the sample sizes for each class.

**Table 11 - Detailed accuracy by class**

| TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---|---|---|---|---|---|---|
| **0.972** | 0.061 | 0.984 | 0.972 | 0.978 | 0.974 | WWW |
| **0.98** | 0.001 | 0.978 | 0.98 | 0.979 | 0.996 | FTP-PASV |
| **0.975** | 0 | 0.983 | 0.975 | 0.979 | 0.988 | SERVICES |
| **0.25** | 0.001 | 0.083 | 0.25 | 0.125 | 0.623 | INTERACTIVE |
| **0.558** | 0.005 | 0.918 | 0.558 | 0.694 | 0.979 | MAIL |
| **0.958** | 0.017 | 0.455 | 0.958 | 0.617 | 0.978 | FTP-CONTROL |
| **0.966** | 0 | 0.969 | 0.966 | 0.968 | 0.993 | DATABASE |
| **0.929** | 0.031 | 0.318 | 0.929 | 0.474 | 0.959 | P2P |
| **0.96** | 0.001 | 0.966 | 0.96 | 0.963 | 0.988 | FTP-DATA |
| **0.933** | 0.05 | 0.959 | 0.933 | 0.938 | 0.975 | Weight Avg. |

### 4.1.6 Confusion matrix

A confusion matrix describes the accuracy of the solution for a classification problem. A confusion matrix holds information about actual and predicted classifications that is done by a classification system. The performance of such systems is commonly evaluated using the data in the matrix. The confusion matrix of 10-fold cross-validation on data set entry12. weka. all class. arff is presented in the below table.

**Table 12 - Confusion matrix**

| a | b | c | D | e | f | g | h | i | Classified as |
|---|---|---|---|---|---|---|---|---|---|
| **15161** | 9 | 0 | 10 | 78 | 0 | 0 | 339 | 0 | a = WWW |
| **8** | 681 | 0 | 0 | 1 | 0 | 0 | 4 | 1 | b = FTP-PASV |
| **2** | 0 | 118 | 0 | 0 | 0 | 0 | 1 | 0 | c = SERVICES |
| **0** | 2 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | d = INTERACTIVE |
| **216** | 0 | 0 | 0 | 1008 | 332 | 7 | 224 | 16 | e = MAIL |
| **1** | 0 | 0 | 0 | 4 | 277 | 0 | 7 | 0 | f = FTP-CONTROL |
| **2** | 0 | 0 | 0 | 0 | 0 | 285 | 8 | 0 | g = DATABASE |
| **13** | 0 | 0 | 1 | 4 | 0 | 2 | 276 | 1 | h = P2P |
| **4** | 4 | 2 | 0 | 2 | 0 | 0 | 9 | 508 | i = FTP-DATA |

### 4.1.7 Evaluation the accuracy

Recall that we have tested Naïve Bayes classifier using 10-fold cross-validation and percentage split with different percentages (10%, 66% and 90%). These have given us a different number of observations. According to their accuracy 10-fold cross-validation technique is proven with higher accuracy than others.

**Table 13 - Results of Naïve Bayes classifier**

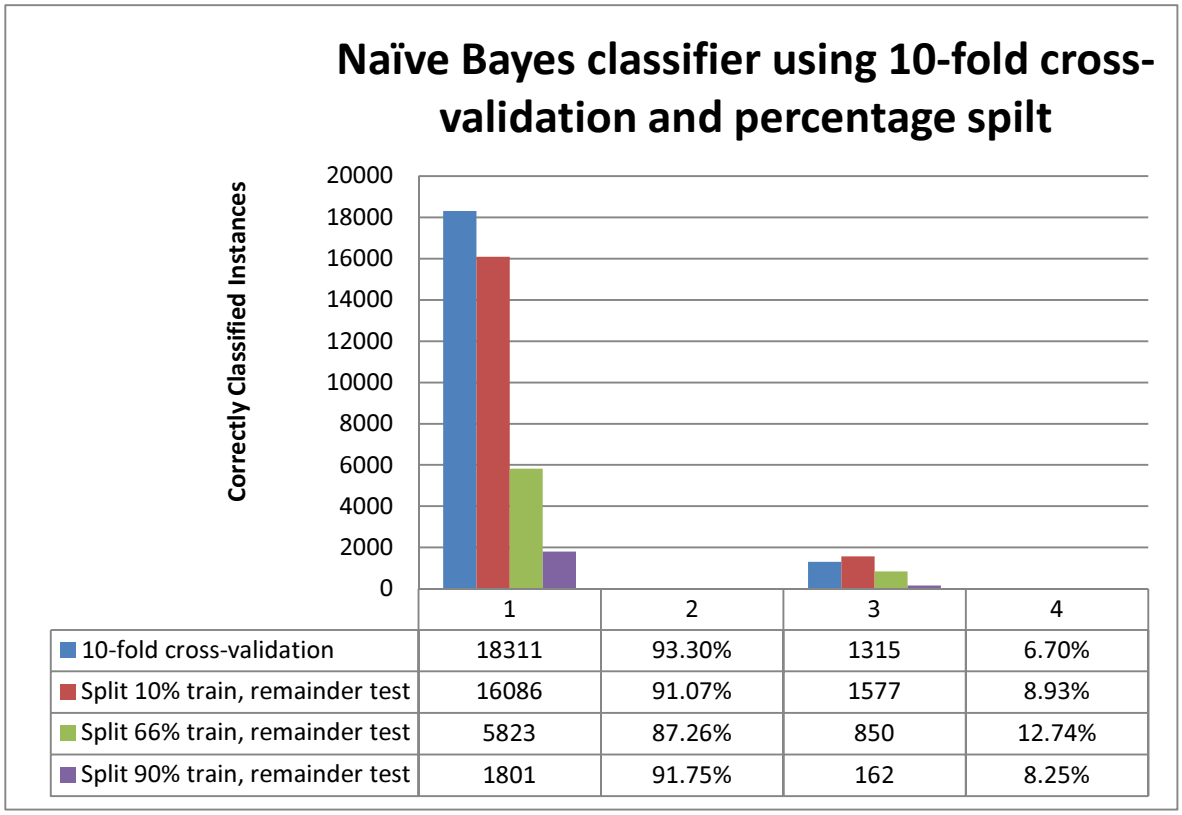
| Test Mode | Correctly Classified Instances | In percentage (%) | Incorrectly Classified Instances | In percentage (%) |
|---|---|---|---|---|
| **10-fold cross-validation** | 18311 | 93.2997% | 1315 | 6.7003% |
| **Split 10% train, remainder test** | 16086 | 91.0717% | 1577 | 8.9283% |

| | | | | |
|---|---|---|---|---|
| **Split 66% train, remainder test** | 5823 | 87.2621% | 850 | 12.7379% |
| **Split 90% train, remainder test** | 1801 | 91.7473% | 162 | 8.2527% |

### 4.1.8 Graphical presentation of the accuracy

The graphical presentation illustrates the results in the following figure. The column 1 is showing the correctly classified instances, which is for 10-fold cross-validation the correctly classified instances 18311 is equal to 93.30%. On the other hand, for split 10%, 66%, 90% the correctly classified instances are 16086, 5823, 1801 respectively. Moreover, column 3 representing the result of the incorrectly classified instances for 10-fold cross-validation is 1315 equal to 6.70% and for split 10%, 66%, 90% the incorrectly classified instances are 1577, 850, 162 respectively.

**Table 14 - Result of Naïve Bayes classifier in Graph**




| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 10-fold cross-validation | 18311 | 93.30% | 1315 | 6.70% |
| Split 10% train, remainder test | 16086 | 91.07% | 1577 | 8.93% |
| Split 66% train, remainder test | 5823 | 87.26% | 850 | 12.74% |
| Split 90% train, remainder test | 1801 | 91.75% | 162 | 8.25% |

### 4.1.9 Output from the HTCondor using J48 classifiers

The following results have been found from the output of HTCondor as a weka.output. The weka.output containing the results of j48 classifiers that has been applied to internettraffic.arff dataset through Weka.

**J48 pruned tree**

**------------------**

8 <= 20: FTP-DATA (529.0)
8 > 20
| 4 <= 53
| | 4 <= 23
| | | 4 <= 21: FTP-CONTROL (289.0)
| | | 4 > 21: INTERACTIVE (4.0)
| | 4 > 23

```
|   |   |   4 <= 25: MAIL (1640.0)
|   |   |   4 > 25: SERVICES (120.0)
|   4 > 53
|   |   109 <= 0
|   |   |   110 <= 0: WWW (38.0)
|   |   |   110 > 0: FTP-PASV (692.0)
|   |   109 > 0
|   |   |   95 <= 25
|   |   |   |   4 <= 443
|   |   |   |   |   4 <= 85
|   |   |   |   |   |   167 <= 55: WWW (35.0)
|   |   |   |   |   |   167 > 55: P2P (3.0)
|   |   |   |   |   4 > 85
|   |   |   |   |   |   73 <= 0: WWW (4.0)
|   |   |   |   |   |   73 > 0
|   |   |   |   |   |   |   149 <= 2
|   |   |   |   |   |   |   |   135 <= 0.095111: MAIL (159.0)
|   |   |   |   |   |   |   |   135 > 0.095111: SERVICES (2.0/1.0)
|   |   |   |   |   |   |   149 > 2: WWW (3.0)
|   |   |   |   4 > 443
|   |   |   |   |   128 <= 0.024205: P2P (237.0/3.0)
|   |   |   |   |   128 > 0.024205: DATABASE (293.0)
|   |   |   95 > 25
|   |   |   |   4 <= 1621: WWW (15517.0/1.0)
|   |   |   |   4 > 1621
|   |   |   |   |   73 <= 0: FTP-PASV (2.0)
|   |   |   |   |   73 > 0: P2P (59.0/1.0)

Number of Leaves  :     18

Size of the tree :      35


Time taken to build model: 685.44 seconds
Time taken to test model on training data: 67.07 seconds
```

=== Error on training data ===

**Table 15 - Error on training data**

| | | |
|---|---|---|
| **Correctly Classified Instances** | **19620** | **99.9694%** |
| **Incorrectly Classified Instances** | 6 | 0.0306% |
| **Kappa statistic** | 0.9991 | |
| **Mean absolute error** | 0.0001 | |
| **Root mean squared error** | 0.0079 | |
| **Relative absolute error** | 0.1555% | |
| **Root relative squared error** | 3.9451% | |
| **Total Number of Instances** | 19626 | |

=== Confusion Matrix ===

**Table 16 - Confusion matrix**

| a | b | c | d | e | f | g | h | i | Classified as |
|---|---|---|---|---|---|---|---|---|---|
| **15596** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | a = WWW |
| **0** | 694 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | b = FTP-PASV |
| **0** | 0 | 121 | 0 | 0 | 0 | 0 | 0 | 0 | c = SERVICES |
| **0** | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 | d = INTERACTIVE |
| **1** | 0 | 0 | 0 | 1799 | 0 | 0 | 0 | 0 | e = MAIL |
| **0** | 0 | 0 | 0 | 0 | 289 | 0 | 0 | 0 | f = FTP-CONTROL |
| **0** | 0 | 0 | 0 | 0 | 0 | 293 | 2 | 0 | g = DATABASE |
| **1** | 0 | 1 | 0 | 0 | 0 | 0 | 295 | 0 | h = P2P |
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 529 | i = FTP-DATA |

=== Stratified cross-validation ===

**Table 17 - Stratified cross-validation**

| | | |
|---|---|---|
| **Correctly Classified Instances** | **19615** | **99.944%** |
| **Incorrectly Classified Instances** | 11 | 0.056% |
| **Kappa statistic** | 0.9984 | |
| **Mean absolute error** | 0.0002 | |
| **Root mean squared error** | 0.0118 | |
| **Relative absolute error** | 0.2611% | |
| **Root relative squared error** | 5.9357% | |
| **Total Number of Instances** | 19626 | |

=== Confusion Matrix ===

**Table 18 - Confusion matrix**

| **A** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **Classified as** |
|---|---|---|---|---|---|---|---|---|---|
| **15596** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | a = WWW |
| **0** | 694 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | b = FTP-PASV |
| **0** | 0 | 120 | 0 | 1 | 0 | 0 | 0 | 0 | c = SERVICES |
| **0** | 0 | 0 | 3 | 1 | 0 | 0 | 0 | 0 | d = INTERACTIVE |
| **1** | 0 | 0 | 0 | 1798 | 0 | 0 | 0 | 0 | e = MAIL |
| **0** | 0 | 0 | 0 | 0 | 289 | 0 | 0 | 0 | f = FTP-CONTROL |
| **0** | 0 | 0 | 0 | 0 | 0 | 293 | 2 | 0 | g = DATABASE |
| **1** | 0 | 0 | 0 | 1 | 0 | 0 | 293 | 0 | h = P2P |
| **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 529 | i = FTP-DATA |

### 4.1.10 Output from the HTCondor using K-Means clustering algorithm

The following results are showing the output from the HTCondor output as a weka.output. The weka.output containing the results of KMeans cluster that has been applied to internettraffic.arff dataset through Weka.

kMeans
======

Number of iterations: 8
Within cluster sum of squared errors: 90826.42514000868
Missing values globally replaced with mean/mode

=== Clustering stats for training data ===

Clustered Instances
0 9685 ( 49%)
1 9941 ( 51%)

## 4.2 BOINC

We have been successfully installed and configured the BOINC server as mentioned details within the appendix B. To date, we have got not enforced BOINC applications because of this could need in-depth and extensive erudition of application source code, that we do not have. In the massive quantity of project, append to the BOINC server requires a real time scientific project and needed an enormous quantity of volunteer contribution, base on it BOINC architecture designed for. In such scenario we have got some constraint on our time and lack of resources. In our work we have only contributed our effort to build up the BOINC server and made ready our BOINC server to append any plausibly real time scientific project and able to affix volunteer contribution.

Our expertise with the BOINC is astronomically positive and it is always inspiriting from the high caliber of interest and dedication for the BOINC.

# 5 Conclusions

In fact, networks with workstations have incremented in great numbers in recent years. These networks represent potent computing environments for the user and their processor is mostly idle for the mundane use. Using idle capacity, it could be exploited by cumulating computers into a grid where this idle capacity could be acclimated to solve computational quandary. The aim of this thesis is to study two such grid systems, BOINC and CONDOR and compare in between them. The system performance is carried out by data mining tasks and making a difference table by the observations. BOINC system is designed for astronomical public projects where millions of volunteers PC can join in solving a computing project. Condor, on the other hand, is designed for projects that can be easily utilized subsisting resources of the organization that carries out the computing. In a situation of distributed computing systems environment, users could be able to integrate computing tasks very easily and could be able to get the results as quickly as possible in a most facile way.

The most sizably voluminous difference for users, such as researchers from the university, it is possible to integrate computing tasks from their own computer in a profoundly facile way in the CONDOR systems and easily can be downloaded their results. In BOINC it is not possible in an easier way by default, only BOINC administrator can do this job. It is needed to integrate incipient applications for their users. The applications handle the results of BOINC projects.

CONDOR strengths, including cycle scavenging works, very configurable and adaptable, support vigorous security methods, interoperates with many types of computing grids, manages both dedicated CPUs (clusters) and non-dedicated resources (desktops), fault-tolerant such as can survive crashes, network outages, any single point of failure, consistently visually examine the jobs and keep posting on the job progress, implement policy on the execution order of the jobs, authenticate on job´s activity.

BOINC solution targets volunteer resources and not enterprise desktops or workstations. General architecture fortifies multiple applications, currently being utilized by several Internet projects. Such as, climateprediction.net – study climate change, Einstein@home – search for gravitational signals emitted by pulsars, LHC@home – amend the design of the CERN LHC particle expeditor, Predictor@home – investigate protein cognate diseases, Rosetta@home – avail researchers to

develop remedies for human diseases, SETTI@home – probing for radio evidence of extraterrestrial life, Cell Computing – biomedical research, World Community Grid – advance our Knowledge of human disease.

It is very arduous to quantify one system is more preponderant than the other; it has all depended on the intended use. Comparing in between BOINC and CONDOR, is likely comparing volunteer computing and intra-organizational computing. CONDOR is most felicitous for academic purposes such as for a researcher at the research center or at the University for their Research Work.

Why use desktop grids:

The main purposes of culling desktop grids, including desktop grid solutions are typically consummate and stand alone, facile to setup and facile to manage, facile to utilize subsisting, but underutilized resources, power of grid grows over times as incipient more expeditious desktops are integrated. Typically, immensely numbers of resources on desktop grids enable incipient approaches to solving quandaries. Notably, aforetime it was only available to users at institutions with supercomputers. With the implementation of the Condor or BOINC system according to their work merit, users can expand their capacity to that of the entire computing networks.

## 5.1 Future Work

Considering thousands of jobs have submitted through distributed systems. It is discovered that some of the submitted jobs have failed to compute the results. The certain failure might be input error, incompatibility with certain OS, shortage of resources to execute the job or uncertain power failure. It is not feasible to examine one or even a handful of jobs in details. To attack this type of problem, the future plan is to use data mining techniques, especially, classification techniques could be useful. Classification technique can be used to gather information regarding the properties of machines, jobs, and the execution environment that is correlated with success or failure.

# Bibliography


[Ahmed et al., 2010] Ahmed M., Zeeshan H. S., and Rizvi S.M.A, Knowledge grid based knowledge discovery in distributed environment. Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on. IEEE, 2010.

[Andrew and Denis, 2014] Andrew W. M., and Denis Z., http://www.cl.cam.ac.uk/research/srg/netos/nprobe/data/papers/sigmetrics/ [Access 2014]

[Altorf, 2007] Altorf M., Data mining on grids.Universiteit Leiden. August (2007).

[Berkeley,2014] Berkeley, http://boinc.berkeley.edu/projects.php/[Access 2014]

[Bouckaert, 2010] Bouckaert R., et al. WEKA---Experiences with a Java Open-Source Project. The Journal of Machine Learning Research, 11 p. 2533-2541, 2010.

[Bouckaert, 2013] Bouckaert R., et al., WEKA Manual for version 3-7-8, 2013.

[Bradley, 2011] Bradley D., et al., An update on the scalability limits of the Condor batch system. Journal of Physics: Conference series, v. 331, no. 6, p. 062002, http://stacks.iop.org/1742-6596/331/i=6/a=062002, 2011.

[Congiusta et al., 2008] Congiusta, A., Domenico T., and Paolo Trunfio. "Service-oriented middleware for distributed data mining on the grid." Journal of Parallel and Distributed Computing 68.1, p.3-15, 2008.

[Cesario, 2012] Cesario, Eugenio, et al. A Visual Environment for Designing and Running Data Mining Workflows in the Knowledge Grid. Data Mining: Foundations and Intelligent Paradigms. Springer Berlin Heidelberg, p. 57-75, 2012.

[CERN, 2014] CERN, http://lhcathome.web.cern.ch/[Access 2014]

[CONDOR, 2014] CONDOR, http://www.cs.wisc.edu/condor [Access 2014]

[Caton et al. 2012] Caton, Simon, and Omer Rana. Towards autonomic management for cloud services based upon volunteered resources.Concurrency and Computation:

Practice and Experience 24.9 (2012): 992-1014.

[David and Ian, 2013] David N. M. and Ian H. W., An open-source toolkit for mining Wikipedia. Artificial Intelligence, 194, p. 222-239, 2013.

[Eibe et al., 2005 ] Eibe F., Mark A. H., Geoffrey H., et al., Weka – A machine learning workbench for Data mining. The Data mining and Knowledge discovery handbook, Springer, p. 1305 -1214, 2005.

[Engel, 2014] Engel T. A., et al., Performance improvement of Data mining in Weka through GPU Acceleration. Procedia Computer Science, 32, p. 93-100, 2014.

[Einstein, 2014] Einstein, http://www.einstein-online.info/spotlights/EaH/ [Access 2014]

[Fayyad, 1996] Fayyad U., Piatetsky-Shapiro G., Smyth P., Uthurusamy R., Advances in Knowledge Discovery and Data Mining, MIT Press, Cambridge, MA , p. 471–494, 1996.

[Frey et al., 2002] Frey J., Tannenbaum T., and Livny M., et al., Condor-G: A Computation Management Agent for Multi-Institutional Grids, Cluster Comput., v. 5, p. 237-246, 2002.

[Gantz and David, 2011] Gantz J., and David R., Extracting value from chaos. IDC iview 1-12, 2011.

[Green and Neil, 2010] Green, Samuel B., and Neil J. Salkind. Using SPSS for Windows and Macintosh: Analyzing and understanding data. Prentice Hall Press, 2010.

[Hall et al., 2009] Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., and Witten I.H. The WEKA data mining software : an update. ACM SIGKDD explorations newsletter 11 (1), p. 10-18, 2009.

[Hawkeye, 2014] Hawkeye, A. Monitoring and Management Tool for Distributed Systems. http:// www.cs.wisc.edu/condor/hawkeye/. [Access 2014]

[Hofmann and Ralf, 2013] Hofmann M., and Ralf K., RapidMiner: Data Mining Use Cases and Business Analytics Applications. CRC Press, 2013.

[Han, 2011] Han J., Kamber M.,and Pei J., 2011. Data Mining. Morgan Kaufmann. 3rd edition.

[Javadi et al., 2011] Javadi B., et al. Discovering statistical models of availability in large distributed systems: An empirical study of seti@ home. Parallel and Distributed Systems,

IEEE Transactions on 22.11, p.1896-1903, 2011.

[Josephsen, 2013] Josephsen, D., Nagios: Building Enterprise-Grade Monitoring Infrastructures for Systems and Networks. Prentice Hall Press, 2013.

[Korpela, 2012] Korpela E. J., SETI@ home, BOINC, and volunteer distributed computing. Annual Review of Earth and Planetary Sciences, 40, 69-87, 2012.

[Korpela, 2012] Korpela, E. J., SETI@ home, BOINC, and volunteer distributed computing. Annual Review of Earth and Planetary Sciences 40, p. 69-87, 2012.

[Liu, 2010] Liu, H., Instance selection and construction for data mining. Springer-Verlag, 2010.

[Larose, 2014] Larose D. T., Discovering knowledge in data: an introduction to data mining. John Wiley & Sons, 2014.

[Labovitz, 2011] Labovitz C., et al., Internet inter-domain traffic. ACM SIGCOMM Computer Communication Review, 41.4, p.75-86, 2011.

[Mateescu et al. 2011] Mateescu, Gabriel, Wolfgang Gentzsch, and Calvin J. Ribbens. "Hybrid computing—where HPC meets grid and cloud computing." Future Generation Computer Systems 27.5 , p.440-453, 2011.

[Mandal et al., 2013] Mandal, Anirban, et al. "Evaluating I/O aware network management for scientific workflows on networked clouds." Proceedings of the Third International Workshop on Network-Aware Data Management. ACM, 2013.

[Michigan, 2014] University of Michigan, http://predictor.chem.lsa.umich.edu/ [Access 2014]

[Mohan, 2012] Mohan R., and Narendhira R.C., A STUDY OF DATA TRANSFER IN A GRID USING GLOBUS TOOLKIT. Journal of Information Systems & Communication 3.1, 2012.

[Mehemed, 2011] Mehemed K., Data mining: concepts, models, methods, and algorithms. John Wiley & Sons, 2011.

[NASA, 2014] NASA, http://climateathome.com/climate@home/ [Access 2014]

[Olson and Shi, 2006] Olson D., Shi Y., Introduction to Business Data Mining. New York, NY, McGraw-Hill, 2006.

[Oxford, 2014] Oxford University, http://www.climateprediction.net/ [Access 2014]

[Parag et al., 2012] Parag M., Zachary M., Rajkumar K., et al., End-To-End Solution for Integrated Workload and Data Management using GlideinWMS and Globus Online. Journal of Physics: Conference Series, v. 396, I. 3, 2012.

[Pérez, et al., 2005] Pérez, María S., et al. Adapting the weka data mining toolkit to a grid based environment.Advances in Web Intelligence. Springer Berlin Heidelberg, P.492-497, 2005.

[Rinat, 2004] Rinat K., Xin Z., and Nicholas K.,Grid-enabled Weka: A Toolkit for Machine Learning on the Grid, ERCIM News, No. 59, October 2004.

[Rani, 2013] Rani P., Middleware and Toolkits in Grid Computing. International Journal of Computer Applications (IJCA), 65.21, p. 13-18, 2013.

[Ries et al., 2012] Ries C.B., Schröder C., and Grout V., Model-based Generation of Workunits, Computation Sequences, Series and Service Interfaces for BOINC based Projects. In Proc. International Conference on Software Engineering Research and Practice (SERP'12), part of WORLDCOMP'12, USA, Las Vegas, 2012.

[Ries et al., 2011] Ries C.B., Schröder C., and Grout V., A UML Profile for the Berkeley Open Infrastructure for Network Computing (BOINC). In Proc. IEEE Conference on Computer Applications and Industrial Electronics (ICCAIE 2011), Malaysia, Penang, December, 2011.

[Sotomayor, 2006] Sotomayor B., and Childers L., Globus Toolkit 4: Programming Java Services, Morgan Kaufmann. 2006.

[Shearer, 2000] Shearer C., The CRISP-DM model: The New Blueprint for Data mining. Journal of Data Warehousing, v. 5, no. 4, p. 13-22, 2000.

[Stanford, 2014] Stanford University, http://folding.stanford.edu/ [Access 2014]

[Schwiegelshohn et al., 2010] Schwiegelshohn U., et al. Perspectives on grid computing. Future Generation Computer Systems, 26.8, p. 1104-1115, 2010.

[Thain et al., 2005] Thain D., Tannenbaum T., and Livny M.. Distributed computing in practice: the HTCondor experience. Concurr. Comput. : Pract. Exper., 17(2-4), p.323–356, Feb. 2005.

[Talia and Paolo, 2010] Talia D., and Paolo T., How distributed data mining tasks can thrive as knowledge services. Communications of the ACM 53.7, p. 132-137, 2010.

[Talia et al., 2005] Talia D., Trunfio P., and Verta O., Weka4WS: a WSRF-enabled weka toolkit for distributed data mining on grids. In Proceedings of the 9th European conference on principles and practice of knowledge discovery in databases. Porto, Portugal, p. 309–320, 2005.

[Thain, 2005] Thain D., Tannenbaum T., and Livny M., Distributed computing in practice: the Condor experience. Concurrency - Practice and Experience, v. 17, no. 2-4, p. 323-356, 2005.

[Urban, 2011] Urban, T., Cacti 0.8 beginner's guide. Packt Publishing Ltd, 2011.

[Witten et al., 2011] Witten I., Frank E., and Hall M., Data Mining Practical Machine Learning Tools and Techniques. Elsevier, Burlington, Massachusetts, 2012.

[Zhang, 2013] Zhang J., et al. Internet traffic classification by aggregating correlated naive bayes predictions. Information Forensics and Security, IEEE Transactions on 8.1, p.5-15, 2013.

[Zhuge, 2002] Zhuge H., A Knowledge grid model and platform for global knowledge sharing. Expert systems with Applications. 22.4, p. 313-320, 2002.

[Zeng, 2012] Zeng, Li, et al. Distributed data mining: a survey. Information Technology and Management 13.4, p. 403-409, 2012.

[Zheng et al., 2010] Zheng, Shi-Ming, et al. Research on distributed clustering algorithm Weka4WS-based in grid environment. Application Research of Computers 11: 017, 2010.

[Zach, 2010] Zach M., Dan B., and Todd T., et al., Flexible session management in a distributed environment. Journal of Physics: Conference Series, v. 219, no.4, p. 042017, http://stacks.iop.org/1742-6596/219/i=4/a=042017, 2010.

# Appendix A

# Implementation

In this chapter we have briefly introduced the installation process of CONDOR.

### A.1 How to Install

The installation file name is condor-6.7.19-Linux-x86-glibc23-dynamic. tar. gz. This is the complete Condor distribution 6.7.19. The version number x86 is Compiled for the 32 bit Intel x86 architecture glibc23 Linked against glibc 2.3. This is particularly important when we talk about the standard universe. The executables are dynamically linked. We can get static, but they are usually not preferred. We have used tar and gz to package this. RPMs are also available.

To install and start up Condor, there are three steps.

### A.2 The First step

Need to unpack condor from the file condor-6.7.19-Linux-x86-glibc23-dynamic. tar. gz. All of the Condor binaries are contained in the release. To install condor into a subdirectory named condor, need to run condor configure (NOT condor installs).

### A.3The Second step

Need to run condor configure

- Condor configure -install -make-personal-condor
- The condor has been installed into /home/users/Anis/condor-6.7.19

In order for Condor to work properly must set CONDOR CONFIG environment variable to point to the Condor configuration file:

- /home/users/anis/condor-6.7.19/etc/condor config before running condor commands/daemons.

There are bin and sbin directories for Condor. The configuration files in etc directory. Condor will read all of condor config, and then read all of condor config.local. Anything in condor config.local overrides what is in condor config. In general, it doesn't matter where putting configuration variables. So need to tell Condor where to find the configuration file, and need to tell the shell how to find the Condor binaries. If using a shell other than bash, the commands may be different.

- export CONDOR CONFIG=/home/users/anis/condor-6.7.19/etc/condor config
- export PATH=/home/users/anis/condor-6.7.19/bin:(PATH)
- export PATH=/home/users/anis/condor-6.7.19/sbin:(PATH)

To make sure it worked, we can check as follows:

- echo condor config

Now the Condor is installed and now need to run. Now running the condor master and then check the Condor is running perfectly.

# Appendix B

# Implementation

In this chapter, we have briefly introduced the installation process of BOINC.

### B.1 Installing the BOINC server

The basic components of a BOINC server are as following:

1. A database server (MySQL)
2. The BOINC daemons (called the `scheduler`, `feeder`, `transitioner`, `validator`, `assimilator` and `file deleter`)
3. A web server (Apache)

Basically need to build the BOINC code for the daemons, and install and configure all these components. There is also some supporting software needed such as Python, PHP, Open SSL and libcurl.

Making sure that has installed the following packages:

- Php5-cli
- php5-gd
- phpmyadmin
- python-mysqldb
- m4
- make
- autoconf
- automake1.9
- build-essential
- libssl-dev

- libmysql++-dev
- libtool
- subversion
- pkg-config
- mutt
- ant
- python-dev
- python-libxml2

A missing package could be the reason if the project not working properly. This is the software required to set up a server.

**B.2 Setting up user Administration - for BOINC**

There needs to be a BOINC user administrator. This user will be responsible for all BOINC server related activities. To set this user up need to add the user to the system

- *sudo adduser boincadmin boinc123*

and add the web server application owner to the BOINC server administrator group which can be done using the command

- *sudo usermod -G boincadmin -a www-data.*

**B.3 Setting up MySQL Database Server**

It is possible to run the database server on the same machine which serves as the web server and runs the BOINC daemons. The default for MySQL is to use */var/lib*. BOINC removes Results and work units from the database after have completed (configuring the expiration time). The database is only used to manage active work (along with keeping track of all participating users and machines).

**B.4 MySQL UNIX Accounts**

Need to create a separate UNIX account and group just for MySQL, and run the server daemon under that account and group rather than as the superuser (`root`). If there are any security problems which allow an attacker to break in via the database system, he can only do what the database user can do, but will not have full `root` powers. The BOINC server stores everything related to the BOINC application in a database. The main BOINC database needs to be set up. The root password for MySQL database.

- *mysqladmin -h localhost -u root password mysql123*
- *sudo MySQL -h localhost -u root -p*

From root user

- *MySQL -u root -p*

Need to do the following step

- MySQL>grant all on *.* to boincadmin identified by `boincadmin-passwd`;
- MySQL>grant all on *.* to `boincadmin`@`localhost`;
- MySQL>grant all on *.* to boincadmin identified by `passwd`;
- MySQL>SET PASSWORD FOR `boincadmin`@`localhost`=`put-the-passwd`;
- MySQL>FLUSH PRIVILEGES;

If there is, no need to set password could follow the above procedures. Otherwise, can follow the below procedures.

- MySQL>SET PASSWORD FOR `boincadmin`@`localhost`=password (`boinc123`);
- MySQL>SELECT * FROM mysql.user WHERE user=`boincadmin`;
- MySQL>quit

### B.5 Configuration File

MySQL database configuration is controlled by the file */etc/my.cnf*

#### B.5.1 Initialization

The first time starts the database server the startup script */etc/rc.d/Inuit. d/mysqld* will run the database initialization command

- *mysql install db*

This will create some initial database tables and the master database account. The user name on the master account is `root`, and it is not password protected.

#### B.5.2 Automatic Startup

Configure the database daemon to start automatically when the system is booted. The command to do that on Red Hat/Fedora; which must be run as `root` are

- *chkconfig mysqld on*
- *service mysqld restart*

### B.6 Building BOINC on UNIX

One now needs to get the BOINC source code by using

- sudo aptitude install subversion

May be asked to confirm with a Y during this process.

Then run

- svn -version

To edit this file write

- vim home /.subversion/servers

Need to change two lines in the [Global] section so that they read:

- http-proxy-host = proxy.aims.ac.za
- http-proxy-host = 3128

Save the file with: w and quit the file with: q command.

Now use subversion to fetch the source, compile and test it. This can be done by using the repository on Berkeley server. This is done using

- svn co http://boinc.berkeley.edu/svn/trunk/boinc boinc trunk

Install automake with the following command:

- sudo aptitude installs automake1. 9

**B.7 BOINC Server Installation**

Then need to run cd home/boinc trunk ./autosetup. After successfully completed the setup, it is necessary to run configuration file through./configure command from command line. After running configure demon on it the client and the server will build up automatically by itself. But it is necessary to build the server first, so to disable the client part it is necessary to disable the client using ./configure - -disable-client command through command line.

That will do a whole bunch of things; including check the system is capable of building BOINC.

- --configure BOINC 6.3.10(Release)
- --Build components : (Server only)

After that, it is necessary to run *make daemon* that will put a lot of random stuff on the screen, followed by.

- sudo make install

**B.8 Checking the Server is running**

BOINC runs heavily on the Apache web server so should need to check the web server is running on the server. To do this should connect from a client using a web browser (e.g. Mozila Firefox or Internet Explorer) and type in http://192.168.52.50 but also need to check that the PHP interpreter is working. That is done by creating a PHP test file and putting it in the root of the web server. The root of the web server is */var/www/*, so go to *cd /var/www/* and then make a file by typing

- sudo vim phptest.php

This is dummy file and anything can be put in.

1. <?
2. php echo
3. "Thanks to all those help to make BOINC possible!!!!!!"
4. phpinfo();
5. ? >

By checking through the web browser, in the url bar type http*://<server address>/phptest.php.* It should be presented through a web page, where the first line contains the comment "Thanks for all those help to make BOINC possible!!!!!!".

Now we have a BOINC server.

**B.9 Run the make project script**

Once the make command finishes successfully want to run the make_project script, which is in the tools subdirectory of the source distribution. It is best to run this script as the Unix root user, if it is possible, because then there will be no problem creating the proper directories or accessing the database.

# Appendix C

# Job Submission

In this chapter, we have introduced the submission of the Weka file to the Distributed systems.

Submission of WEKA files into Condor High throughput Computing systems:

**C.1 First job submission on Condor**

```
##############################
# entry12. arff
# weka.classifiers.bayes.NaiveBayes
##############################
universe = java
output = weka.output
error = weka.error
log = weka.log
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
TRANSFER_FILES = ALWAYS
executable=/home/anishur/weka-3-6-10/weka.jar
jar_files =/home/anishur/weka-3-6-10/weka.jar
arguments=weka.classifiers.bayes.NaiveBayes  –t  entry12.arff
transfer_input_files= entry12.arff
queue
```