Software License Management on a Grid Computing Environment

Diploma thesis
Maurice Valery Misse Ndong
Matr.-Nr. 223703
Aachen, March 3rd, 2011

Supervised by
Dipl-Inform. Joachim Götze
Chair of Integrated Communication Systems,
Technical University Kaiserslautern

First thesis supervisor:
Prof. Dr. Dr. h. c. Otto Spaniol
Chair of Computer Science 4,
RWTH Aachen University

Second thesis supervisor:
Prof. Dr. Paul Müller
Chair of Integrated Communication Systems,
Technical University Kaiserslautern
ACKNOWLEDGEMENT

The writing of this dissertation has been one of the most important and significant academic challenges I have ever had to face. This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study. It is to them that I owe my deepest gratitude.

First and foremost, my ultimate gratitude to Prof. Dr. Dr. h. c. Otto Spaniol, former Head of Chair of Computer sciences 4, RWTH-Aachen whose sincerity and encouragement I will never forget. Prof. Spaniol has been my inspiration as I hurdle all the obstacles in the completion of this research work until the day of his retirement.

Prof. Dr. Paul Müller, Chair of Integrated Communication Systems, Technical University Kaiserslautern who had kind concern and consideration regarding my academic requirements and accepted to give me a chance when I needed it the most.

Dipl.-Inform. Joachim Götze who undertook to act as my mentor despite the fact that I have to leave Germany to the United States and his many other academic and professional commitments. His wisdom, knowledge and commitment to the highest standards inspired and motivated me weeks after weeks.

Shanda Misse, my wonderful wife, without whom this effort would have been worth nothing. You kept pushing me to finish my school, encouraging and believing in me. Your love, support and constant patience have taught me so much about sacrifice, discipline and compromise.

Ismet Aktas and Uta Christoph who helped me by giving me the hardest critics, which allow me to finish and correct my thesis according to the requirements.

Maurice and Agnes Ndong Misse, my parents, who have always supported, encouraged and believed in me, in all my endeavors and who so lovingly and unselfishly cared for my son Xavier.

Xavier Anthony Misse, my son, who was born before this thesis was completed and who spent many days with relatives to allow me to focus. I am deeply sorry for the time we spent apart.
Richard Misse Ndong and his Wife Tatjana, who opened their home and fed me when I was coming here in Aachen for my school.

Franck Hermann Misse Ndong, my brother and my sister Amalea Misse Ndong for their continual support and presence.

Gator and Judy Allmond, my parents-in-law, for mentoring me and looking after Xavier at a moment’s notice and for all your encouragement and profound understanding.
Affidavit

I hereby declare that the following Diploma thesis "Software License Management on a Grid Computing Environment" has been written only by the undersigned and without any assistance from third parties.

Furthermore, I confirm that no sources have been used in the preparation of this thesis other than those indicated in the thesis itself.

Aachen, March 3rd, 2011

____________________ ______________________
Place, Date Signature  Maurice Valery Misse Ndong
Abstract

The purpose of this thesis is to explain and show how software licenses acquisition on a Grid computing environment is processed. Which processes have to be filled to obtain a license, and which legal factors are required to enable scientific researches all of this on a Grid Computing environment. Another important aspect of this research will be accounting of software licenses, which plays a major role on a software management point of view. In this work a study will be also made on accounting and how to manage the financial part of License in a Grid environment.

The difference will be focused on accounting in computer science and accounting in companies in this context. The goal to achieve here is to find out why using software licenses and how is it managed in this environment. Another reason will be to develop a way to make software available to more users, try to see how to find a solution to facilitate sharing resources in a grid environment. The thesis will try to find solutions and ideas to improve software management issues on a Grid level.
# Table of Contents

ACKNOWLEDGEMENT .................................................................................................................. 3

Abstract ........................................................................................................................................ 8

1 Introduction .................................................................................................................................. 14

2 State of the art ................................................................................................................................. 17

2.1 Parallel computing ...................................................................................................................... 17

2.1.1 Definition and roles ............................................................................................................... 17

2.1.2 Advantages of parallel computing ....................................................................................... 17

2.2 Cluster computing ..................................................................................................................... 18

2.2.1 Definition and roles ............................................................................................................... 18

2.2.2 Advantages of cluster computing ....................................................................................... 18

2.3 Grid Computing ......................................................................................................................... 19

2.3.1 What is a Grid? ..................................................................................................................... 19

2.3.2 Definition of Grid computing ............................................................................................ 21

2.3.3 Components .......................................................................................................................... 22

2.3.4 Grid infrastructure and Constitution ................................................................................ 25

2.3.5 Pros and cons of grid computing ....................................................................................... 30

2.4 Accounting in Grid computing .................................................................................................. 31

2.4.1 Definition and roles of accounting ...................................................................................... 31

2.4.2 Different types of accounting in a grid computing environment ....................................... 32

2.5 Software licenses ....................................................................................................................... 41

2.5.1 Definition .............................................................................................................................. 41

2.5.2 Definition of a License ......................................................................................................... 42
1 Introduction

The expansion of technology nowadays generates also the creation and invention of new applications and software. The main reason that motivates this master thesis is software licenses which are one the most important components to have when it comes to using applications or services. Without agreements or rights of utilization it is difficult to use a product and get a service from a third party.

The purposes of this thesis are:

1. First of all to understand and define exactly the concept of software licenses
2. Learn how to differentiate this concept from other types of licenses
3. Study how to manage and distribute them on a modern Grid computing environment.
4. Finally find new features to record and process grid accounting transactions

The most important aspect of this thesis is to solve the issues encountered with software licenses in Grid computing due to the lack of accounting mechanisms on a software management point of view. In this work a study is also how to manage the technical part of License in a Grid environment.

In this paper, we also determine what requirements and processes have to be fulfilled in order to obtain a license, and which legal agreements are required to enable scientific researches in a Grid Computing environment. The thesis will try to elaborate new features for accounting methods for software licenses. Our focus is first on trying to define and learn how to use Accounting in a Software License environment to understand exactly what is trying to be achieved here. Previous studies have been made on different types of accounting for financial, networking and hardware purposes.

The goal to achieve here is to first find out why software licenses are used, second how they are managed in a Grid environment and third how to control their distribution. Microsoft already demonstrated that it exists 2 modes of managing Licenses which are per server licensing and per seat Licensing.
Another goal is to develop a way to make software available to more users worldwide, try to find a solution to facilitate sharing resources in a grid environment. The thesis will find solutions and ideas to improve software License management issues on a Grid level.
2 State of the art

This section defines and introduces the concept of distributed computing to help us understand the idea and the structure of a Grid in general and a Grid computing in particular. It will also introduce the different accounting mechanisms, which will help us understand their roles.

2.1 Parallel computing

2.1.1 Definition and roles

Parallel computing is the simultaneous use of multiple computing resources to solve a computational problem

- To be run using multiple CPUs. Each CPU in a parallel computer system has access to a small amount of local memory and also global memory that is accessible by all processors. The CPU’s connection is made so that the results from one CPU directly flow into another for further processing
- A problem is broken into discrete parts that can be solved concurrently
- Each part is further broken down to a series of instructions
- Instructions from each part are executed simultaneously on different CPUs

The computer resources can include:

- A single computer with multiple processors;
- An arbitrary number of computers connected by a network;
- A combination of both.

2.1.2 Advantages of parallel computing

Save time and/or money: Parallel clusters because of his structure which is constructed from cheap components can help saving money and time by putting more resources to work at the same time. That why parallel computing’s tasks can be completed in a short run.

Solve larger problems: because of the size and the complexity of certain problems it is sometimes impossible to resolve issues on a single computer, especially one given with limited computer memory.
Provide concurrency: A single compute resource can only do one action at a time. Multiple computing resources can be doing many actions simultaneously.

Use of non-local resources: The connection’s art can also play a big role here. Having computer resources on a WAN\(^1\) connection for example. Network wide computer resources can also be utilized in scarcity at local resources (WifiNotes, 2011).

2.2 Cluster computing

2.2.1 Definition and roles

Cluster computing is a form of computing in which a group of computers are linked together so that they can act like a single entity. The problems are the ones, which can be replaced into small subtasks The Architecture of Computer hardware and systems software, Irv Englander.

Clusters are typically used for High Availability (HA) to assure greater reliability or High Performance Computing (HPC) to provide greater computational power than a single computer can provide.

2.2.2 Advantages of cluster computing

One of the most important reasons to use cluster computing is the desire to create redundancy in a computer network to ensure its availability and correctness.

A common application for cluster computing is in hosting web sites, with the cluster distributing the load of the visitors across an array of computers so that many visitors can be accommodated. This technique is also used for gaming servers used by large groups, to avoid lag and logon problems.

Some computations are extremely complex and they require multiple computers, which have to communicate fast with each other, to function properly, as changes at one location can change the entire system. For example, the simulations used to test theories in meteorology are often run on computing clusters. Without a cluster, the calculation might be impossible to do, or it might take a very long time to process (Smith, 2003-2011).

\(^{1}\) Wide Area Network (WAN): a computer network that covers a broad area (i.e., any network whose
2.3 Grid Computing

Grid computing is basically using computer resources, which come from independent computer units via advanced exchanging networks to different user types.

One of the most important goals of Grid computing is to execute computing tasks in a distributed set of resources instead of one central resource.

But before we can define what the term “Grid computing”, we have to understand what a “Grid” is.

2.3.1 What is a Grid?

There are so many different arts to define a Grid. Some works on the subject challenged this aspect of not having a proper definition of this term. A Grid was finally defined after studies made on its uses and characteristics (Grindshaw, 2002).

According to (Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis, and Eduardo G´omez-S´anchez, 2002) on Grid characteristics, to be called a grid an entity must be able to accommodate a large amount of data and information from 0 to millions. So in simple words it has to have a large scale of a Grid infrastructure to pack all these resources. For example Platform Computing, it is the pioneer and longtime leader in HPC² workload management software for large-scale grid infrastructure. It is the backbone of mission-critical grids deployed by the world’s leading businesses and government organizations (Corporation, 2009).

A Grid can also be heterogeneous as it can host different kind of hardware and software entities (data, files, software components, programs, computers, supercomputers and networks, etc….). It has to be capable of managing all these objects.

A grid must possess a resources sharing capability that means different organizations are allowed to mutually access and use resources according to each other’s policies and securities. This leads to a variety and difference of administrations. These resources have to be controlled and managed so that they can be compatible with the environments where they will be used or needed.

---

² High-performance computing (HPC) uses supercomputers and computer clusters to solve advanced computation problems. Today, computer systems approaching the teraflops-region are counted as HPC-computers.
The Grid concept refers to the virtualization of computing resource in the sense that end-users should have the illusion of using a single source of computing power" without knowing the locality of the computation (Grid computing and eScience, 2005-2006). This is a transparent aspect of the Grid because a Grid should be seen as a virtual computer (Grid Characteristics and Uses: a Grid Definition, 2002).

Examples of this virtualization are the use of digital certificates to access systems on behalf of the user, third party file transfer between machines authenticated via certificates, client tools for workflow composition with the workflow being consigned by agents such as brokers (Grid computing and eScience, 2005-2006).

A grid must provide also services to the users. So it has to be accessed easily according to the standards and requirements of each user. It has to be dependable and consistent. That means it must provide a high quality service and insurance every time it is used.

A grid can allow an extended access to its environment so that resources can be available and where their failure can be minimized. But this process must only be possible according to Grid’s policies and rules.

A grid cannot be only qualified as being a computing paradigm because of its role of provider for computational resources for huge and important applications. But it is also an infrastructure that makes globally remote and different resources work together and become one entity in order to provide computing support to a large amount of applications. Another aspect of the grid definition results from its uses and purposes. The computing support part of a grid can be categorized according to their utilities and the grid architecture’s demands.

That why for example distributed supercomputing support will be used to modify the completion time of a required job so that the application is made faster by coupling multiples computational resources (Klaus Krauter†, Buyya, R., and Maheswaran, M. , 2001). Distributed supercomputing support can also be utilized to solve huge issues on a single system (Foster, 1998). The only problem with this technic is the double schedule on the use of highly expensive resources and the scalability of protocols and algorithms of a large number of nodes. (Klaus Krauter†, Buyya, R., and Maheswaran, M. , 2001)

For On-demand computing support applications will be using the Grid to recover lost resources that cannot be locally located and cost effective. For
example a financial application allowing users to perform accurate stock market analysis and price prediction employing their home desktop computer (Klaus Krauter†, Buyya, R., and Maheswaran, M. , 2001).

Data-intensive computing support gives applications throw the Grid the rights to combine new information from distributed data storage, digital libraries and databases. For example creating a new database using data mined from a number of online databases

Collaborative computing support lets applications use the Grid to enable and enhance human to human interactions (Klaus Krauter†, Buyya, R., and Maheswaran, M. , 2001) in a synchronous and asynchronous way (Ian Foster, Carl Kesselman, Steven Tuecke, 2001) via a virtual space. For example grids are groupware applications and multiconferencing applications.

Multimedia computing support allows applications to use grids to present contents assuring end-to-end Quality of Service (Klaus Krauter†, Buyya, R., and Maheswaran, M. , 2001). Videoconference applications are a typical example of application requiring multimedia-computing support (Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis, and Eduardo G´omez-S´anchez, 2002).

According to the list of grid characteristics, uses and different categories of support given here or extracted from literature, a “Grid” can be defined as a large-scale geographically distributed hardware and software infrastructure composed of heterogeneous networked resources which are owned and shared by multiple administrative organizations coordinated to provide transparent, dependable, pervasive and consistent computing support to a wide range of applications. These applications can perform distributed computing, high throughput computing, on-demand computing, data-intensive computing, collaborative computing or multimedia computing (Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis, and Eduardo G´omez-S´anchez, 2002).

2.3.2 Definition of Grid computing

After studying and defining the terminology of a Grid it will be easier to grasp the meaning of Grid computing. Grid computing (or the use of a computational grid) is applying the resources of many computers in a network to a single problem at the same time - usually to a scientific or technical problem that requires a great number of computer processing cycles or access to large amounts of data. Grid computing is changing the way the world is doing science, as well as business, entertainment, social science and more. Enabling Grids for E-science (EGEE) is the largest European grid in particular and in the
world in general bringing together more than 120 organizations to provide scientific computing resources to the European and global research community. EGEE comprises 250 sites in 48 countries and more than 68,000 CPUs available to some 8,000 users, 24 hours a day, 7 days a week, but there are many other grid projects and activities taking place around the world (Cafe, 2010).

A well-known example of grid computing in the public domain is the ongoing SETI (Search for Extraterrestrial Intelligence) a Home project in which thousands of people are sharing the unused processor cycles of their PCs in the vast search for signs of "rational" signals from outer space. A radio telescope at Arecibo, Puerto Rico scans the sky for signals. An entire sky survey returns about 39 TB of Data for processing. All these examples are known under the name of “Volunteer Computing” (Irv Englander, 2001)

Grid computing can also be defined as being a method of exploiting the power of many computers in a network to solve problems requiring a large number of processing cycles and involving huge amounts of data. So instead of using a network of computers simply to communicate and transfer data, grid computing taps the unused processor cycles of numerous sometimes thousands of computers (planet, 2011).

So far grid computing systems have had limited commercial application, although they have been used successfully to solve large-scale scientific problems.

2.3.3 Components

2.3.3.1 Resources

A Resource is a representation of a physical or logical entity (software applications, hardware, operating system, clusters), which offers different interfaces (API\(^3\)) for login, management, and surveillance of this process.

Here are the different types of resources:

---

\(^3\) Application Programming Interfaces
Computation

It is the most common resource in the Grid computing cycles provided by the processors of the machines on the grid. The processors can vary in speed, architecture, software platform, and other related factors, such as memory, storage, and connectivity.

There are three primary ways to exploit the computation resources on a grid computing environment:

1. Control and run an existing application on an available machine on the grid rather than locally.
2. Use of an application designed to split its work in such a way that the separate parts can execute in parallel on different processors.
3. Run an application to execute many times on many different machines in the grid to distribute resources or provide services.

“Scalability” is a measure of how efficiently the multiple processors on a grid are used.

Storage

The second most important resource used in grid computing is data storage. The Grid provides an integrated view of data storage, which is sometimes called a
“data grid.” Each machine on this environment usually provides some quantity of storage for temporary use. Storage can be memory attached to a processor or it can be “secondary storage” using hard disk drives or also other permanent storage media. Memory attached to a processor usually has very fast access but is volatile and is therefore non-persistent too. It would best be used to cache data to serve as temporary storage for running applications. Secondary storage in a grid can be used in interesting ways to increase capacity, performance, sharing, and reliability of data. Using the storage on multiple machines with a unifying file system can increase capacity. (Berstis, 2002)

**Communications**

Another important resource of a grid is its data communication capacity with connections within the grid and outside of the grid. Communications within the grid are important because it is responsible for sending jobs and their required data to points within the grid. Some jobs require a large amount of data to be processed. But the problem may not always reside on the machine running the job. The bandwidth available for such communications can often be a critical resource that can limit utilization of the grid. External communication access to the Internet, for example, can be valuable when building search engines. Redundant communication paths are sometimes needed to better handle potential network failures and excessive data traffic (Berstis, 2002).

**Software and licenses**

Most of the software installed on a Grid is may be too expensive to install on every grid machine. Using a grid, the jobs that require the use of this software are sent to the particular machines on which this software happens to be installed. When the licensing fees are less expensive, this approach can save significant expenses for an organization. Some software licensing arrangements authorize the software to be installed on all of the machines of a grid. But it can also limit the number of installations that can be simultaneously used at a time. License management software keeps track of how the software is being used by an end-user. That means to know how many concurrent copies of the software are being used and to prevent more than that number from executing at anytime. The grid job schedulers can be configured to take software licenses into account, optionally balancing them against other priorities or policies.

**Capacities, architectures, and policies**

Platforms on the grid will often be differently setup like architectures, operating systems, devices, capacities, and equipment. Each of these items represents a
different kind of resource that can be used in a Grid computing environment as criteria for assigning jobs to machines. Some software may be available on different architectures, for example, PowerPC and x86, as software is often designed to run on a particular type of hardware and operating system only. These attributes must be considered when assigning jobs to resources in the grid.

2.3.3.2 Service

**Service:** (e.g. CPU-Cycle) is the use of different distributed resources (*CPU and memory*). On an economic view, a service looks like a one of the foundation concept of Grid computing as far as principal characteristics.

**Jobs and applications**

Even if a lot of resources on the grid may be shared and used, they are usually accessed via an executing “application” or “job.” The term “application” is usually used as the highest level of a piece of work on the grid. However, sometimes the term “job” is used to define the equivalent action. Applications may be broken down into any number of individual jobs, in turn, can be further broken down into “sub jobs.” The grid industry uses other terms, such as “transaction”, “work unit”, or “submission”, to mean the same thing as a job. Jobs are programs that are executed at an appropriate point on the environment. A grid application that is organized as a collection of jobs is usually designed to have these jobs execute in parallel on different machines in the grid.

**Scheduling**

The grid computing is responsible for sending a job to a given machine to be executed. The easiest part of a Grid computing Environment is the fact that the user may select a machine suitable for running his job and then execute a grid command that sends the job to the selected machine. More advanced grid systems would include a job “scheduler” of some kind that automatically finds the most appropriate machine on which to run any given job that is waiting to be executed. Schedulers react to current availability of resources on the grid. A service transforms resources that are not accessible to private club goods by a work overload of resources

**2.3.4 Grid infrastructure and Constitution**

In a Grid computing environment the architecture is made first of all of 5 different layers: Figure 1 depicts the architecture of a layered Grid computing environment which will be described in details in the following sections.
Figure 2 Layered Grid computing Architecture

2.3.4.1 Fabric Layer

The Fabric layer acts like a provider by granting access to resources like computational resources, storage systems, catalogues, network resources, sensors, etc. One of these resources is computational resources which are resources used by some mathematical models in computational science to study the behavior of a complex system by computer simulation. In this case, it can be the number of steps necessary to solve a problem. Fabric Layer functions are also known as being Resource Layer implementations of protocols like APIs and SDKs to access and control local resources. For example, National Center for Supercomputing Applications (NCSA), which is an American state-federal partnership to develop and deploy national-scale cyber infrastructure, that advances science and engineering. The Dell Intel 64 Cluster (Abe) is a shared resource that is 60% allocated through the National Science Foundation allocation process, with the remaining time allocated at the discretion of the NCSA leadership to serve state of Illinois, University of Illinois strategic initiatives, and NCSA’s Private Sector Program Partners.

Another Grid Resource can also be Storage systems. For example, IBM’s storage system, which is a flexible, policy-based hierarchical storage management developed by IBM. It can enable a user to utilize cluster, LAN, and/or SAN technology to aggregate the capacity and performance of many
computers, disks, and tape drives into a single virtual file system. Catalogues that are sort of huge ensemble of libraries for a database in other words are metadata which definitions of database objects such as base tables, views (virtual tables), synonyms value ranges, indexes, users, and user groups are stored. They also belong to Grid Resources group.

Network resources, which are resources allocated to connections, are also Grid Resources. For example Network resource management, which uses broadband networks to keep track of the way the link of resources allocation to connections. In other words Network resource management Network keeps track of the bandwidth and controls the allocation of capacity to connections when requested as part of the connection setup process (ATM process).

The last element of the Grid Resources group is Sensors, which are devices that measure physical quantities and convert them into signals, which can be read by observers or by instruments. For example the way thermometer works. A mercury-in glass thermometer converts the measured temperature into expansion and contraction of a liquid, which can be read on a calibrated glass tube, and the calibration on a thermometer is standard.

### 2.3.4.2 Connectivity Layer

Connectivity Layer defines the basic communication and authentication protocols required for Grid specific networking-service transactions. Communication protocols here enable the exchange of date between Fabric Layer resources. Its requirements are composed of Transport, Routing, and Naming. Authentication protocols are created on communication services to provide cryptographically secure mechanisms for verifying the identity of users and resources. It requires single sign on from users, delegation that means a user runs a program on his behalf so that he can access authorized resources, local security integration where each resource provider has the ability to employ all varieties of local security solutions, and the user based trust relationships where the user does not require to identify himself every time he wants to access resources.

### 2.3.4.3 Resource Layer

This layer is built on different connectivity layers protocols. First of all are Communication protocols. When two dissimilar computers systems communicate with each other they require a standard set of instructions for communicating with each other and these instructions are known as protocols. Protocols are the communications standards and the set of rules that source and
destination computers must abide by and follow in order to communicate with each other. They determine that how data will be transmitted between two computer computers.

They also define the data packet size, authentication, signaling, data compression, error checking and retransmission of the packets. They also define that how the packet information will be organized while traveling over the network. Another aspect of the resource layer is Authentication protocols. Authentication is a fundamental aspect of system security. It confirms the identity of any user trying to log on to a domain or access network resources. Authentication protocols are cryptographic protocols that means abstract or concrete protocol that perform a security-related function and applies cryptographic methods, with the purpose of authenticating entities wishing to communicate securely.

Resource layer defines protocols for secure negotiations that mean these protocols are designed to make negotiations and contract signing processes secure and confidential. For example an e-notary, which is an electronic signing of, contracts known as a type of signing protocols. These protocols include mechanisms for the establishment of a secure channel, via cryptographic key exchange, over an insecure medium. Such a channel can then be used for ensuring the confidentiality, authentication, and/or integrity of the communications between two parties (Citeseerx).

The fourth point on this part is Accounting protocols. These protocols can sometimes be used to quantify traffic to support billing, Quality of Service (QoS) assurances, and other objectives. A common way to do this, as seen in cellular data systems, is to associate traffic with specific clients on an access network and use a Network Access Server (NAS) with an associated accounting system such as a Remote Authentication Dial In User Service (RADIUS) server to maintain accounting records such as the number of bytes or packets the client exchanges with a server on the Internet (Alwyn Goodloe, Matthew Jacobs and Gaurav Shah, 2005).

These Protocols call Fabric layer functions to access and control local resources. Resource Layer only handles individual resources, which are resources with a specific required instance that includes people and computers. (Fran Berman, Geoffrey Fox, Anthony J. G. Hey, 2003) This layer also manages and provides APIs that support open source License and SDKs to each available Grid, ignores global state and -atomic actions across the resource collection pool, which is the responsibility for the collective layer. But the two primary protocols classes of the resource Layer are: Information protocols which are
used to acquire information about the structure and state of resource, for example its configuration, its current load and its usage policy, and Management protocols which are used to negotiate access to a shared resource with details information like for example resource requirements (reservations and QoS). Because of his responsibility of sharing relationship agent, Management protocols serve also as a “policy application point” for the consistency of requested protocols applications. Resource and Connectivity protocol layers form the neck of our hourglass model, and as such should be limited to a small and focused set. The management of data in a Grid computing environment requires a lot of efforts and easy access to heterogeneous and distributed data formats like. Those data are generally known as Data Collection. Data collection contains many entities that are hierarchically organized and can also have sub collections. (Fran Berman, Geoffrey Fox, Anthony J. G. Hey, 2003)

2.3.4.4 Collective Layer

While the Resource Layer manages individual resources, Collective Layer is taking care of all global resource management and interactions with collections of resources. This protocol layer implements a large variety of sharing behaviors using a small amount of resource layer and connectivity layer protocols. Collective layer protocols span the spectrum from general purpose to highly application or domain specific Collective functions can be implemented as persistent services, with associated protocols, or as SDKs (with associated APIs) designed to be linked with applications Scheduling for interactive and parallel applications on Grids.

2.3.4.5 Application Layer

The Application Layer enables the use of resources in a Grid environment through various collaborations and resources access protocols. This is where you use publicly available APIs to interface with underlying infrastructure and also the architecture is composed of the user applications that operate within a VO environment. It is a term used to categorize protocols and methods in architectural models on a computer networking environment. This is also the User Interface where the Grid application is being designed and used. In this layer Grid service compositions called workflows combine with Grid services to new Grid applications.
2.3.5 Pros and cons of grid computing

2.3.5.1 Pros of Grid computing

Grid computing is utilized for many years now and its advantages are many. Grid computing appears to be a promising trend for three reasons:

1. Its ability to make more cost-effective use of a given amount of computer resources
2. As a way to solve problems that can't be approached without an enormous amount of computing power
3. It suggests that the resources of many computers can be cooperatively and perhaps synergistically harnessed and managed as collaboration toward a common objective.

In some grid computing systems, the computers may collaborate rather than being directed by one managing computer. One likely area for the use of grid computing will be pervasive computing applications - those in which computers pervade our environment without our necessary awareness.

These are their characteristics:

- Support of Distributed commands and projects
- Support of high performance projects or researches
- Reduce time of job execution of big projects or researches
- Focus more for now on Information technologies

But there is also a remote access to high performance computing resources. The best example is the I-WAY project created in San Diego in 1995. This project was made to provide a large scale test bed for innovative high performance and geographically distributed applications deployment. (Fran Berman, Geoffrey Fox, Anthony J. G. Hey, 2003)

2.3.5.2 Cons of Grid computing

Here are the some disadvantages:

1. More memory to run applications required
2. Fast connection between resources
3. Applications stopped by Licenses servers
4. Lack of tools for software licenses management
5. Political aspects about sharing resources
2.4 Accounting in Grid computing

To this date, Grid computing has mostly been confined to scientific environments. However, Grid computing is making its way into industry. Grid technology is now starting to gain industry support and is reaching more widespread deployment in commercial settings.

2.4.1 Definition and roles of accounting

Grid Accounting is a process of identifying, measuring, and reporting financial information of an entity on a Grid computing environment. It can also be defined as a phase in which a system audits the usage of system resources. The Accounting system properly determines a confidence level in the existing accounting information and adequately address and present erroneous or missing accounting data. It also protects the privacy of the users and organizations involved. Accounting also means managing and reporting of information belonging to usage of resources. The most important goals are optimization and daily maintenance of resource usage. One of the main focuses of “Accounting” in D-Grid, which is the German Grid, is the design of a substantial D-Grid-wide System (D-Mon) for resource surveillance, user and resource management, and usage compensation (Prof. Dr.-Ing. Gabriele von Voigt, Dipl.-Ing., Wolfgang M¨uller, Dipl.-Kfm. Dr. rer. nat. Claus-Peter R¨uckemann, Dipl.-Geophys, 2006).

D-Mon’s prospective is to create a base towards the D-Grid-wide monitoring System that can utilize the monitoring information provided by different middlewares as well as external tools (Heike Neuroth, Martina Kerzel, Wolfgang Gentzsch (Ed.), 2007). Accounting protocols are used to quantify traffic to support billing, QoS\(^4\) assurances, and other objectives (Alwyn Goodloe, Matthew Jacobs and Gaurav Shah, 2005).

P. Gardfjaell (GardfjÄall, 2004) defined Grid accounting as being the process of maintaining a consistent Grid-wide view of virtual organizations members' resource utilization that means making sure that resources are utilized in a normally and that users are getting the information they need.

The Different roles of a Grid Accounting system are:

- Designing the schema for the accounting attributes

\(^4\) Quality of Services
• Ensuring the necessary collectors and sensors are in place in the resource providers,
• Defining and deploying repository and access tools for the reporting and analysis of the grid wide accounting information.
• Forming a basis for economic compensation. When usage information is available and ready to be used, the resource provider can apply a transformation to convert resource usage into some monetary transaction. Direct payments could also be envisioned, where resource usage is charged for immediately. Credit card transactions are a concrete example for this method.
• Enabling a reinforcement of Grid-wide resource allocations. Resources might only grant access to users whose current resource allocation has not been used up.
• Allowing tracking of user jobs. Users can obtain information about a submitted job, such as where it was submitted, the resources it consumed and the output it generated.
• Enabling evaluation of used resource. Resource providers need to be able to find what extent different resources have been utilized. In the same direction administrators can obtain information about what job executed on a specific resource at a certain time. Such information can be useful when debugging programs or tracking malicious users.
• Using resources to dynamically assign priorities to user requests based on previous resource usage.

2.4.2 Different types of accounting in a grid computing environment

In this section the different types of accounting models are distinguished. There are 2 main types of accounting models and they are referred as allocation based and aggregation based. According to (Gardfjäll, 2004) in an allocation based accounting each project is assigned a resource allocation called a quota. This quota may not be exceeded, unless a less strict quota enforcement policy has been specified. Also in this accounting model, quota enforcements are performed on service requests (job submissions), possibly denying resource access if the allocation has been used up. To enforce Grid-wide quotas with strict consistency guarantees, resources must share a common view of each project’s resource utilization. In order to acquire a consistent view of accounting information across the Grid, the information provided to resources must always be updated.

On the other hand accounting uses also another model called Aggregation. It is a pervasive theme in accounting research with many beneficial uses identified. Compare with Allocation based accounting, Aggregation based accounting does
not enforce project quotas. The usage information is taken as resources. The collected information can for example be used to determine a cost in real money at a later time. This means that unless strict timing guarantees are imposed on the accounting system (how fast accounting information needs to be available) an aggregation based approach can cache information locally at the resources level and only updating periodically global (i.e., Grid-wide) accounting information. This method increases performance and scalability.

Aggregation based accounting can also be viewed as a special case of allocation based accounting, in which the allocation is infinitely large in which no quota enforcement is necessary. An allocation based accounting model is set, since it represents a more general accounting model (SweGrid).

There are also some approaches used to work with accounting. Here are some of them.

- Accounting Processor for Event Logs (APEL).
- Distributed Grid Accounting System (DGAS).
- Grid Accounting Services Architecture (GASA).
- Grid Based Application Service Provision (GRASP).
- Grid Service Accounting Extensions (GSAX).
- Nimrod/G.
- SweGrid Accounting System (SGAS).

In this thesis the first 3 Grid accounting approaches are studied to understand and see their suitability in this topic to fit the requirements of our subject (Comparison of Grid Accounting Concepts for D-Grid, 2006).

### 2.4.2.1 Accounting Processor for Event Log (APEL)

APEL is a log processing application, which is used to interpret gatekeeper, and batch system logs to produce accounting records. APEL currently supports PBS and LSF batch systems but can easily be extended to support other variants. Portable Batch System\(^5\) (PBS) provides sophisticated scheduling and resource management facilities for a wide range of systems:

- Fault isolation, crashes recovery, and support for logging and accounting.
- Production hardened and demonstrated scalability—to thousands of processors.

---

\(^{5}\)Portable Batch System: The name of computer software that performs job scheduling.
• Separate, fully tunable, scheduler—production schedulers support priorities, dedicated time, and dynamic backfilling (Bayucan, 2000).

Load Sharing Facility ⁶(LSF) is a commercial computer software job scheduler sold by Platform Computing.

This is possible because of a newly developed plug-in architecture, which separates the core functionality from the actual log parsing. A complete accounting record is composed of the grid user, the job id of the submitted job and the resources used when executing the job and other elements which are not relevant to be mentioned. This information is typically dispersed between several different log file types such as those produced by the gatekeeper or batch system. For resource usage, a query is issued to the site’s GIIS (Global International Infrastructure Symposium) to lookup the CPU performance for the computing nodes where the job was executed.

APEL attempts to collect all this piecemeal information together and manages it within a database. A further process carried out by APEL then attempts to join the data together to produce a list of final accounting records with all necessary details filled-in.

APEL is then used to publish the generated accounting records into R-GMA (Relational Grid Monitoring Architecture) whereby they are collated at the Grid Operations Centre (GOC) using an R-GMA secondary producer.

R-GMA is an implementation of the Grid Monitoring Architecture (GMA) proposed by the Global Grid Forum (GGF), which models the information infrastructure of a Grid as a set of Consumers (who request information), Producers (who provide information) and a single Registry (which mediates the communication between producers and consumers). R-GMA imposes a standard query language (a subset of SQL) on this model - so consumers issue SQL queries and producers return database rows (known as tuples) in reply. R-GMA also ensures that all tuples carry a time-stamp, so that monitoring systems (which require time-sequenced data) are inherently supported. (User guide for Apel - Accounting using PBS Event Logging , 2004)

APEL provides support for republishing the complete local copy of accounting records to RGMA (in cases when the GOC was offline). It also provides a mechanism for reliable delivery using a basic integrity check to compute the

---

number of records that were last published compared with the actual count stored on the GOC. Each accounting account is unique and for each grid job is only one record available (APEL, 2005).

**Advantages of this approach**

1. The first advantage noticed when using Accounting Processor Event Logs is the support for various metrics (CPU time, wall clock time, RAM etc.). That means that APEL works like a measurement tool because it interprets or publishes the values of the gatekeeper and batch system logs to produce accounting records.
2. It is possible for APEL to evaluate different resources by putting heterogeneous resources to a normalized status.
3. APEL can also support various batch systems via plug-ins.
4. APEL is conforming to the Grid Monitoring Architecture (GMA) of Open Grid Services Architecture (OGSA) in other words APEL follows OGSA Principles that are Models, capabilities and Mechanisms.
5. APEL also supports 2 middleware called: LCG-2/ gLite. LCG-2 (Legal Certainty Group) provides experience, which operates and manages a global grid service. According to (E.Slabospitskaya, 2005) Glide middleware is a Service Oriented Grid middleware, which provides services for managing distributed computing, storage resources and the required security, auditing and information services.
6. The main Source code is available by APEL.

**Disadvantages of this approach**

1. Centralized database concept (within Grid Operations Centre, GOC).
2. Exclusive accounting of batch system jobs (currently PBS, BQS).
3. Exclusive support for homogeneous nodes (homogeneous worker nodes).
4. PBS logs, gatekeeper logs and system news needed (often no longer available).
5. Dependent on monitoring- and information system R-GMA.

### 2.4.2.2 Distributed Grid Accounting System (DGAS)

This part will show a fully distributed view of Grid usage accounting and a methodology for allocating Grid computational resources for use on a Grid computing system. For an accounting system to be functional in a Grid computing environment, it needs to be decentralized, scalable and flexible. It must have a minimum impact on local accounting and should not make any
limiting assumptions about whether accounting is done by user, group, project, or site.

Distributed Grid Accounting System (DGAS) was created within the European Union Data Grid Project and was known between 2001 and 2004 as Data Grid Accounting System®. It is only in April 2004 that the name Distributed Grid Accounting System was changed and used for the first time for the EGEE’s project. DGAS the functionality needed for the implementation of a complete stand-alone infrastructure for computing and storage accounting of national Grids and Grid virtual organizations. It provides sensors for many different Local Resource Management Systems, repositories for persistent storage of usage records at site-level or national/regional-level and it comes along with a tool, called HLRMon, which provides a user-friendly interface to display the accounting data. DGAS and HLRMon are supported by the INFN Grid and EGEE projects. The Enabling Grids for E-Science (EGEE) project is funded by the European Commission and aims to build on recent advances in grid technology and develop a service grid infrastructure, which is available to scientists 24 hours-a-day (EGEE, 2006).

The purpose of DGAS is to implement Resource Usage Metering, Accounting and Account Balancing (through resource pricing) in a fully distributed Grid environment. The focus will be made on accounting and account balancing in this part.

Accounting records are stored on a distributed infrastructure of Home Location Register (HLR) servers (for scalability). Accounting records are associated to user accounts and resource accounts identified by

- User DN (certificate subject) as user account ID
- Global resource ID (CE ID) as resource account ID

Accounting records can be forwarded from Resource/Site HLRs to User HLRs: Users /VO admins do not need to query many Site HLRs for accounting information, but only the HLR that manages the respective account (user level accounting). For the account balancing perspective, DGAS Price Authority (PA) servers:

- Responsible for setting resource prices (only CPU for now),
- Prices can be set manually or determined dynamically,
- Pricing algorithms are dynamically linked libraries and can be customized as needed.
Job cost determined by Resource/Site HLR from resource price (per unit) and resource usage (number of units). Account balancing by exchanging virtual credits between user and resource account. Resource pricing and account balancing are Optional in DGAS (Piro, 2006).

Figure 3: “this figure depicts the basic architecture of the DGAS service. It consists of a set of sensors collecting records on the computing farm and a site-local server known as HLR, which archives them on a RDBMS”. (DGAS, 2010)

**Advantages of this approach**

Like all the other types of accounting the Distributed Grid Accounting System has also some particular points that make it being unique. The Distributed Grid Accounting System (DGAS) is a full featured Grid accounting toolkit. Here are the different points that have been mentioned previously: There are 3 main advantages with using DGAS. These advantages are following:

1. Security/Reliability:
   - Resource HLR and User HLR have the capacity and the possibility of storing Accounting records.
   - In a Grid computing environment dealing with DGAS the use of temporary user accounts is common. Template user accounts have been made for this effect. User HLRs accept accounting records only for registered users and only from trusted Resource HLRs.
   - The other way around Resource HLRs accepts records only from registered resources. This transactions and transmissions for accounting record transactions between HLRs are called
asynchronous and in case of failures (e.g. temporary network problems) are retried.

- It is completely grid oriented that means it is completely distributed without having to use a central repository. It relies upon a network of independent accounting servers used to keep the accounting records.
- It enables the deployment of hierarchical servers, suitable for regional grids or virtual organizations.
- It can be used for Computational Usage Records regarding CPU, memory, storage and so on.
- It can be used for Economic Accounting, managing the cost of the jobs executed by Grid User on Grid Resources.
- The Communication aspect is made possible here via a Secure Sockets Layer (SSL) and via XML-based data format.

2. Privacy:
   - Only a Simple Logging and authorized access with central Workload Management System Server is necessary for DGAS to access accounting data (users, admins).
   - Another advantage of this method will be the fact that DGAS supports various bases of economic concepts and is encrypted based on Grid Security Infrastructure (GSI). (Piro, 2006)

3. Scalability
   - Scalability by the decentralization of a bank structure: For each Virtual Organization (VO) it exists an own Home Location Register (HLR), which implies a high degree of scalability. It relies on a modular and scalable architecture, suitable to be adopted both by little sites, with few CPUs, and up to very large facilities.
   - DGAS also support for various accounting policies and middleware like LCG-2/gLite. (DGAS, 2010)

**Disadvantages of this approach**

But there are also some issues to this system that can show another side of features.

- These issues are the following:
- There are no conformity with Open Grid Services Architecture (OGSA) and Open Grid Services Infrastructure (OGSI).
- DGAS exclusively support virtual monetary units (Grid Credits)
o It uses central administrated resource broker and only one Pricing Authority can result in scalability problems.
o There are no special provisions that avoid failures or specified data regeneration.
o There are not enough features for administration of user accounts.
o The Implementation of DGAS is tightly coupled with Data Grid or D-Grid Workload Management System (DWMS). It is kind of difficult to find an application without influence on local cluster software environment.
o DGAS also does only charging/clearing of resources because there is no real accounting of resource usage.
o There is also no integration method for resource evaluation available.
o They are a large overhead because each resource has an own Home Location Register (HLR) that has to be administrated from a central instance.
o There is Low interoperability for various middleware like e.g. UNICORE etc.

**Grid Accounting Services Architecture (GASA)**

Grid Accounting Services Architecture (GASA) also commonly called GridBank was created to allow the sharing of resources between multiples administrative domains and platforms and support unambiguous recording of user identities against resource usage (GridBank, 2003).

There are 2 definitions of Grid Bank.

Grid Bank is a secure Grid-wide accounting and (micro) payment handling system. It maintains the user (consumers and providers) accounts and resource usage records in the database.

GridBank can also be thought of as a web service for Grid accounting and payment. GridBank uses SOAP over Globus toolkit’s sockets, which are optimized for security (GridBank, 2003).

**Advantages of this approach**

- Grid Bank (GB) secures, and fully described payment- und billing system.
- It also has a simple attendance and modification because of modular architecture (implicit high degree of extensibility).
- GB supports the Quality of Service (QoS), e. g. resources, costs, deadline etc.
The User accounts are administrated on central server not at the resource providers.

GB uses Resource Usage Records (RUR) from all resources used to provide the service.

Different metrics are supported by GB on this methodology (CPU time, main memory, secondary memory, software libraries etc.).

All payment activities are made via secure connections, and they are using certificates (PKI X.509) and can access lists of content.

GB supports monetary and virtual monetary units so do different payment schemes.

GB decentralizes trade servers, which arrange prices of the resources.

GB also supports middleware like Globus Toolkit v.2 or later versions.

Source code is also available in this case. GridBank: A Grid Accounting Services Architecture (GASA) for Distributed System Sharing and Integration, Alexander Barmouta, Rajkumar Buyya

**Disadvantages of this approach**

- On the other hand the scalability of GB is really small due to registration on side of resource users and providers on central server.
- GB is not based on open standardized Grid protocols because of its low interoperability.
- GB also has no conformity with OGSA and Web-Services and is not a distributed structure.
- Local accounts in a GB are use on side of Grid Service provider (GSP) by specifying permissions on the template accounts.

**SweGrid Accounting System (SGAS)**

SGAS is a computational Grid composed of six computer clusters with a total of six hundred CPUs. The cluster sites located in GÖteborg, LinkÖping, Lund, Stockholm, Umeå, and Uppsala are connected through the GigaSunet High-performance network. The computer time of SGAS is controlled by the Swedish National Allocations Committee (SNAC), which issues computer time, measured in node hours per month, to research projects. Grid-wide node hour allocations are assigned to projects based on their scientific merits and resource requirements. These node hour allocations specify the computer time per month that each project is entitled to. The whole allocation may be used up at one cluster or in parts 7 Accounting in Grid Environments at any number of clusters.
The accounting system must provide coordinated enforcement of project allocations across all resource sites (GardfjÄall, 2004).

2.5 Software licenses
2.5.1 Definition

People often confuse a serial number with a License and think that just because they put a valid serial number in they are good to go. The truth is much different than that. In reality most of the times a License is an agreement, permission required to execute an action legally and licensing a piece of software is often referred to as assignment. The act of assigning a License is usually as complicated as pointing to a server or a piece of software with a finger and saying: he is now assigned a license. That is about all it takes. What people might not realize is these licenses are actually governed by many carefully crafted legal documents that tell users in really small print what they are allowed or not allowed to do with that license. Sometimes these are EULAs (End-User License Agreements) and often those EULAs refer back to some other legal documents with even more terms and conditions. Many people don’t actually go off and read all of those terms and conditions. But they exist and have to be followed (WiseGeek, 2010).

But some parts of the software are too expensive for one person to be owned. For this reason people only install software on one machine in a Grid computing environment. For this option where License keys are very important, having only one copy of this software License cuts down licensing costs and does not decrease the productivity. This can be done remotely. It is no need to be physically connected to the machine. The most important measure to take care off is to make sure that the machine with the software is logged on (Sze Hwei, O., 2004). In other words the machine with the software just has to be on. But this procedure has a hack because it seems to be legally not appropriate. So it has to be first of all some types of arrangements made between both parts the vendors and the clients so that the software can be used in the right way. Most of the commercial applications vendors adopt software License to control the using authorization (Sze Hwei, O., 2004).

Another way to deal with these licenses is to control the software access with a limited licensing agreement. This method limits the number of installations that may be able to run simultaneously at any moment making all other machines continued their job normally (Sze Hwei, O., 2004). For example licenses for

---

7 End-User License Agreements: http://www.webopedia.com/TERM/E/EULA.html
colleges. Colleges buy the licenses and let students use them to install the desired software so that they can limit the budget of purchasing software licenses.

2.5.2 Definition of a License

A License is a permission that grants a user an access to an application or a product. In our case a License is an authorization by the ISVs\(^8\) to use the licensed material, which is the software. Individuals on personal computers under License from the developer of that software use Mass distributed software. The most popular of all licenses is the End-User License Agreement (EULA). This License applies always when the user installed software on a computer. According to the terms of the License agreement, the user may install the software on a limited number of computers. A product key, also known as a License key, is a specific software-based key for a certain program, or a computer game or computational resources. It is used to identify that the copy of the program/game or resources is original.

2.5.3 Roles and importance of Licenses in Grid computing

On a Grid environment dealing with resources from different sources requires the use of licenses, to filter and give allowance to certain machines and deny access to others.

One of the roles of Licenses in a Grid is to make a computer/CPU/Processor available only for the users where the License was installed. A License can be described as being an assurance and maintenance of your product. The assurance part guarantees the user to most current version of the product in a long-term basis. But it does not apply to all the licenses and depends on the terms and policies of their agreement. Some of them will charge the end-users for every update and maintenance made on this software and others will just provide the maintenance but not the updates and vice versa (Microsoft, Services Providers License Agreement, 2009). For example Microsoft; Microsoft licenses process has two options that are available for acquiring licenses for customers using software services:

1. If providers offer their services, they may acquire the licenses following the Services provider License Agreement (SPLA) program and must ensure that their customers are using licensed products that match with

\(^8\) Independent Software Vendors
the Services provider Use Rights (SPUR). (Microsoft, Services Providers License Agreement, 2009)

2. The customer may receive licensed products through Licensing programs and ask a virtual organization (service organization) to manage the licensed products only if the following requirements are met:

- “The customers acquire both the server licensed product and Client Access Licenses (CALs) in the case of redistribution of services.
- These licensed products are used only for the customer and only for him. They cannot be shared and has to belong only to one owner.
- Licensed product use complies with the Product Use Rights (PUR)\(^9\).
- The core License rights under the SPLA program are:
  - The right for the licensee to provide software services.
  - The right for the licensee’s customers (end users) to access or use the licensed products (Microsoft) running on the services provider’s server.
  - The right for the licensee to provide multi-tenancy or provide to more than one customer to use the same licensed server products.
  - The right to pay for actual use on a monthly basis.
  - The right to no up-front License fees and no minimum commitments.” (Microsoft, Services Providers License Agreement, 2009)

An External Connector License is another option to the individual CALs used to allow your vendors, and other external users to access providers’ server. According to (Microsoft, Services Providers License Agreement, 2009) providers cannot use this License to distribute access to a software application that can enable their customers to access business operations. Another option could be to enable the access to a software application where access to the software application has a beneficial priority. But the External Connector License can also be used in the case of Self Hosted ISV. Providers can then be assigned an External Connector License under existing Volume Licensing agreements. (Microsoft, Services Providers License Agreement, 2009)

“Customers may acquire licensed products on their own through Volume Licensing programs and ask a services provider to manage them under the following circumstances:

1. The customers must acquire both server licenses and CALs.
2. The services provider can use these licenses only for those customers.
3. License use must comply with the PUR.” (Microsoft, Services Providers License Agreement, 2009)

2.5.4 Licenses without License Manager

The issues that we encountered are to find out who is using which License and when. We need to find out how the vendors are tracking the utilization of their products by end-users. When Licenses were used back in the early 1980s, there were many different concepts that play an important role in this process. These concepts included:

1. Licenses are fixed to a particular CPU that means if a Grid had 20 or 30 PCs according to their CPU performances that is how they were assigned a license. System administrators who were able to allow the distribution of resources to the CPU’s users then controlled these licenses. “In the early 80s when the PC became popular, PC software vendors kept this earlier view of licensing software with the exception of pricing based on CPU performance” (Mirabella, 2010). If it needs to be applied nowadays, the licenses have to be assigned to some fixed IP addresses once the software has been installed. The vendors can then see who is using the software License and when it was utilized.

2. Another concept to keep track on the License utilization was to limit the number of copies of software made to disk or tape. This is important because by applying this method the Software Vendors were able to just make to count on how many copies of software for certain amount of machines they sold. For example again with our Grid with 20 PCs. Each CPU would have to obtain a copy of the software so that it can work properly on the machine.

With this concept of software licenses system administrators were able to keep track of software by limiting the copies of software to hard disks that were directly attached to licensed CPUs. Networked file systems just demonstrated to the technology world that this view of licenses was obsolete and not 100% reliable. The huge evolution and market acceptance of networked computing in the late 1980s forced a new idea of what software licensing really means on a network of workstations, diskless workstations, servers, X-terminals and ASCII
terminals (Mirabella, 2010). Without License management technology, it is not easy for a system administrator with a network of may be 10 workstations to keep track of how many floating licenses are used or distributed (Mirabella, 2010).

Microsoft is tracking licenses this way. For example 20 local computers or within a small domain is time consuming but possible. Tracking licenses without the assistance of automated tools across an entire organization with multiple domains can be very difficult, extremely costly, and overly time consuming. (Licensing and License Manager, 2011)

2.5.5 Licenses with License Manager

In the late 80s Workstation customers called for new concepts of software licensing:

1. Software is a network resource. Licenses "float" on the network
2. Software costs are a function of how many users simultaneously runs the software. Pricing is based on users, not CPU performance
3. Value exists in the use of software, not in the number of copies on disk or tape

During this time the network computing started evolving. The new concepts of licensing were praised and required dynamic tracking of users and software licenses. That is how the new technology “License Manager” was created. License with License Manager are more efficient for software vendors because they are now able to control the execution via a License key and authorization and no more software "copy protection" used in the PC marketplace instead of copying. PC users complained against copy protection method because the practice inconveniences honest customers while giving them nothing in return. In contrast, UNIX customers pressure software developers to provide them with License management because it makes software a network resource and makes software pricing more equitable.

Buying or acquiring more licenses is much easier under License management because it eliminates the need to send out another tape or copy with the software. Now the user requires only a "License key". It is a brief string of data that describes the License plus some security codes. These License keys can be distributed by e-mail, FAX or verbally over the telephone and does not require to be posted per mail all the time. Today, most major vendors of workstation software use License managers to control their products. It is a tool that
manages and tracks licenses and usage throughout an organization can help contain these costs. (Mirabella, 2010)

2.5.6 Licenses on networking level

By Microsoft for example Network solutions generally have two major components:

- Servers that contain information and provide services
- Clients that access information and services

But the two components have separate and different licenses. Each server requires a License and each client computer accessing a server also requires a License (called a Client Access License). To study how the licenses are utilized in a network, it is important to divide the client-licensing mode in two categories: Licenses per server and License per seat.

“In per server licensing, each Client Access License is assigned to a particular server and allows one connection to that server for the use of that product” (Licensing and License Manager, 2011). It means that each client has the right to access only one server that he has been assigned to so that he can use the product that was put to his disposal. “A connection, in this case, is to a server and not to an individual share point or printer on that server” (Licensing and License Manager, 2011). If the user connects to a file named \Maurice\Apps and \Maurice\Public, that is considered as being only one connection for licensing purposes (Licensing and License Manager, 2011). But if he connects in per server mode to a server from two different computers using the same username, that is considered two separate connections and is charged twice. The owner of the License must have at least as many Client Access Licenses dedicated to a service on that server as the maximum number of client computers that will connect to that server at any point in time. (Licensing and License Manager, 2011)

If the per server option is selected, license’s owner must specify upon purchasing new Client Access Licenses, the number of Client Access Licenses (which corresponds to the number of concurrent connections) that have been purchased for that server. Also with per server licensing, once the specified limit for concurrent connections is reached, the server does not allow new users with new licenses to connect, unless enough users (including administrators) have disconnected to get below the specified limit. No more computer connections are also allowed to that server even for the administrators whose connections are also counted as part of the total number of concurrent
connections. When the limit is reached, though, administrators are still allowed to connect to manage the lockout situation. (Licensing and License Manager, 2011)

Per server option is often the most economical one for networks in which clients tend to connect to only one server. That means the clients are only using for one License for the connected server. Or occasional-use or special-purpose servers, and they do not all need to connect at the same time. If a network environment has multiple servers, each server licensed in per server mode must have at least as many Client Access Licenses dedicated to it as the maximum number of clients that will connect to it at any one time. (Licensing and License Manager, 2011)

‘With a per seat licensing; only one Client Access License is assigned to each specific computer that accesses the server” (Licensing and License Manager, 2011). It means that there are licenses for each client and there are all different to each other to access the product. After a computer acquired a License in the per seat mode, it can access any network server running that the server product sometimes at no additional charge depending on the terms and agreements of the contract. This mode requires a Client Access License for each computer that will access a particular product on any server. Once a computer is licensed for a particular product, it can be used to access that product at any computer (Windows NT Server). Multiple users with multiple licenses can also log on to that single computer. (Licensing and License Manager, 2011)

But having a valid per seat mode Client Access License does not guarantee you access to a server that is licensed in the per server mode and has reached its specified limit. Such a connection also consumes one of the licenses assigned to the pool of available per server licenses. Therefore, you can connect only if there are per server licenses available. For example, if a server in per server mode has 50 Client Access Licenses dedicated to that server and has fewer than 50 simultaneously connected clients, additional clients can connect. If, however, that server has reached its specified limit, additional clients cannot connect, even if they have a valid per seat mode License for that service.

In the other hand to obtain correct licensing results when dealing with per seat licenses, Software License providers might need to group certain users and make them members of a License group. License groups show a relationship (also known as a mapping) between users and computers and should be used only when one of the following configurations is true:
• Multiple people using one computer, such as when people share jobs or there are multiple shifts using the same computers.
• Many users are using many computers but there are still a different number of users than computers, such as in a university computer lab or in a retail store.
• One user too many computers, such as happens in many software developers' offices where they need to develop on one computer and test their applications on several different computer-hardware platforms.

A License group is composed of:

• A single descriptive name for the group to identify every user
• A specified number of per seat licenses assigned to the group to track each of them when it comes to manage them.
• A specific list of users who are members of the group. Also to make a difference between the seats and see who belongs to which group.

The number of licenses assigned should match the number of computers in the licensing group. This number does not have to match the number of users in the group. It is recommended to give more users than less. Just in case the number of users expected to connect is not exact (Licensing and License Manager, 2011).

In the first example with multiple users but only one computer, you need only one Client Access License. You are licensing the number of computers, not the number of users.

The second configuration is explained with this example, if you have 50 users accessing 10 computers, you need to purchase only 5 Client Access Licenses to cover those 10 computers. But the system administrator has to keep track of all 50 users and how often they use each computer to access various servers. The License group you create is assigned 5 Client Access Licenses and includes 50 users (Licensing and License Manager, 2011). For this case there is one user who needs multiple Client Access Licenses to fulfill the legal licensing requirements even though License Manager shows only one user. In this case, the License group can only allow one user with multiple Client Access Licenses (Licensing and License Manager, 2011).

With the per seat licensing mode, the number of licensed computers and time are not important here. Any number can be used to connect at any time to any Windows NT Server. But the user will have to purchase a separate Client Access License for each computer even if he uses client Microsoft’s operating-
system software for example or from a third-party vendor or use from another source another client software supported by Windows NT Server. The per seat option is often the most economical, the one where users save the most and the one for networks in which clients tend to connect to more than one server (Licensing and License Manager, 2011). For example in Windows 95 the Operating system package comes without Client Access License included, the License must be purchased separately in addition to the operating-system software (Licensing and License Manager, 2011).

It is also possible sometimes to have the per server and per seat modes within a single institution because it depends on how much the different server products are used in each department. The same operation can be done on a single server if multiple server products are running. But a given server product, such as SQL Server, cannot simultaneously run in two modes on the same server at the same time (Licensing and License Manager, 2011). The reason is that two default instances cannot be on the same server (Warren, 2010).

But network licensing is very expensive for companies and virtual organizations. To solve this issue Microsoft (Windows NT 4.0 server) came up with two administrative tools:

- Licensing option in Control Panel
- License Manager program

These tools are good for companies willing to cut down on licensing costs (Licensing and License Manager, 2011).

2.5.6.1 Licensing option in Control Panel

It is a tool used to control and reduce the price of software licenses in a network environment. This tool is mostly used for server purposes. The Licensing option in Control Panel can be used to change the licensing mode on a computer from per server to per seat. It allows the user also to configure licensing replication for computers and to add or delete per server licenses for each product installed on the server (Licensing and License Manager, 2011).

2.5.6.2 License Manager Program

With License Manager, administrators or License owner can (locally or remotely) change the licensing mode of their products from per server to per seat, add or delete Client Access Licenses for their products, create License groups, and view licensing information at several levels. Per seat licenses can be administered only through License Manager (Licensing and License Manager,
2.5.7 Types of licenses Management

A License Manager is a software tool used by software vendors to control where and how the licenses are able to be distributed. License managers protect the software vendor from losses due to piracy and enable him to offer a range of licensing models, such as trial licenses, subscription licenses, feature-based licenses, floating licensing and usage-based or metered licensing from the same software package they provide to all users. License Manager is independent of local specific LMS (License Management Service), and suitable for various and dynamic License resources management in Grid environment (Feng Guofu, Wang Yinfeng, Guo Hua, Dong Xiaoshe, 2006)

2.5.8 Definitions and roles of 3 types of licenses management: GenLM, SmartLM, FlexLM

2.5.8.1 GenLM

GenLM is a License management framework that allows ISVs to manage their License usage in a distributed world (for example Grid computing environment). The main idea is to attach the License not to a node or a person but make licenses available for a specific task or issue. This way, a user can buy a License for each job (per-job license) and run the job on any suitable resource. To secure the licensing framework and prevent tampering cryptographic primitives are usually used. In the next section we will introduce the GenLM approach, describe the overall design, Technical evaluation and implementation of GenLM. GenLM supports at least 3 main licensing techniques:
**Support for lifetime License agreements**

In this part the user cannot break the contract made with the ISVs. It is a sort of digital signature as soon as he agrees to use the license. In most cases money paid for the use of this License is not refundable. That means even if the user decides not to continue with the software or deletes it. It will still be charged and will be always assigned to him or her. The Software may automatically deactivate and become non-operational only at the end of the Service Period, and you will not be entitled to receive any feature or content updates to the Software unless the Service Period is renewed. But before that it is impossible to stop it. A good example will be the Norton License agreements policies where it explained the rights and obligations of the user when it comes to changes during the subscription period (GenLM: License Management for Grid and Cloud, 2009).

**On-Demand licenses**

On-Demand licensing is basically the fact that the user will be charged only for the time or period they used or using the product not more. So this method is like a metered usage of software. The usage can be managed by the ISVs so that the user saves more on the service. This technique is user friendly and can be one of the improvements in a grid computing environment. The Software Tsoft is an Example of this method where the software License can be put on timeout if the user is not working on it for a while. But the break is determined by the ISVs (GenLM: License Management for Grid and Cloud, 2009)

**Mobility of licenses**

This method mostly applies to software licenses for certain server applications and all external connector (EC) licenses. This does not apply to software licenses for the Windows Server operating system, Client Access Licenses (CALs), or Management Licenses (MLs); also not for Software licenses that comes from other retail sources that means if it is not a License proper to a software it is not applying to it (GenLM: License Management for Grid and Cloud, 2009).

One of a good example to show a License file’s characteristics is the Token Licenses. A token License provides maximum flexibility in the usage of a suite of MSI products while limiting the total concurrent usage of those products. The token License specifies the number of tokens available for concurrent use. Each product is enabled by a finite number of tokens. When a process is executed,
this number of tokens is checked out. Tokens are checked in when the program or process exits, or when they are automatically or manually checked in (GenLM: License Management for Grid and Cloud, 2009).

- The dongle solution where the user needs hardware to execute the program properly. The software checks which features of the software want to be used and fulfilled the requirements (a license) and then give access based on the communication with a dedicated hardware token (GenLM: License Management for Grid and Cloud, 2009). For example MIL stands for Matrox Imaging Library. A run-time License takes the form of either a software or hardware key (dongle). A run-time License enables one to run a MIL-based application only for the package(s) (groups of modules) purchased (GenLM: License Management for Grid and Cloud, 2009).

- The system fingerprinting solution where the user requires a fingerprint reader. It is a hardware that gives user’s information to software vendors. This process is called fingerprints identification. The user will be scanned out of a database and checked if it matches the information on this database. This step is fingerprints verification process. The fingerprint is just hardware with identification numbers and other identifiers components. When this process is done the ISV provides License keys for each user. These keys are made of numbers and characters and are used to unlock the needed software. It can also be something else than a fingerprint it just depends on the Hardware recognition capability to read licenses on the user (GenLM: License Management for Grid and Cloud, 2009). For example if you have a safe in a bank or at home with this system. The safe will play its role that means locking and opening. But if the user wants a safe with a fingerprint option he will have to pay more because the price of the licenses may be included in the sale prices or he will have to pay extra for the license. That means a normal safe with a fingerprint feature but without software to make it work properly. Another example can be the MIL License Manager utility.

Advantages and inconveniences of these types of licenses

GenLM is like a border patrol that controls the border and only admits people with the right paperwork. GenLM acts the same in real world by giving only access to users who fulfilled certain requirements. GenLM verifies carefully all the little details on the batch jobs before granting permission to anybody from outside the grid. Just like a border patrol GenLM needs a passport to allow the entry in the country. The passport in this case will be a License token. The advantages for a resource Manager and is that:
• Applications can be utilized even without a valid license
• Different user with different types of licenses can still access with a License agreement to the same resources
• The ISVs can create on top of the License token an On demand License mechanism that can

But the disadvantage with this operation is that the License Token can be found on the input data and an incomplete job can be restarted (Mathias Dalheimer, 2008).

Improvements

An integration process is made by the vendors for the software development of GenLM to allow users to be autonomous. SDK has been especially developed to solve GenLM issues (Dalheimer, 2008).

A Software Development Kit is a piece of copy protection code that has been developed for a specific application environment. SDK's are harder to implement than wrapper technology because you must be a developer with the tools that built the original application. They can nevertheless be relatively simple to implement if the correct solution is chosen. SDK's are stronger and harder to hack than wrappers and much cheaper and more flexible than dongles. They also tend to have more features and integrate with applications much more tightly, allowing for features such as custom screens, for example. (Nalpeiron, 2009)

A good example of a SDK will be LicenseGen SDK. It is a software development kit that provides functions to manage License of the protected application. With this SDK, users can build their own customized License management tool (similar to License Key Manager) but with expanded capability, or can also be integrated into sales and support system environments. This SDK can be used to develop a website too that can issue the License Key on-demand and automatically by Mathias Dalheimer.
2.5.8.2 SmartLM

In this part of the thesis shows why SmartLM is clearly focusing on helping new industrial opportunities based on the creation of the emerging Service market. Users will be dealing with service-oriented infrastructure as a means to deliver new software services in great many fields like mechanical industries, Finance, Entertainment, Retail, Pharmaceutics, etc. SmartLM is contributing to the technology convergence (virtualization, Grid, SOA, etc.) and interoperability with focused contributions to Web Services standards and specifications of the Open Grid Forum. The role of SmartLM is to provide a new generic licensing virtualization framework according to a standard utilization, and integrate it in major Grid middleware solutions. The main goals of SmartLM are:

- To Understand the requirements of licenses for Grid and Cloud
- To be used and deployed in the commercial environment where software vendors, application service providers, resource providers and end users are involved.
- To identify service-oriented business models for distributed scenarios across organizations.
- To design and build a secured platform-independent licensing management framework.
- To provide models and technologies for accounting and billing of licenses.
- To provide a generic and flexible licensing virtualization technology.
- To enable and validate the licensing management tools with commercial applications deployed in Grids.
- To overcome technical limitations by supporting the use of licensed applications across organization boundaries.
- To Implement licensing technology that supports distributed use of licenses
- To support commercial ASP environments, e-Science Grids and Clouds
• To use open standards to create a licensing virtualization framework.
• To find a common ground between users and ISVs
• To adapt a number of widely-used license-protected commercial applications to be executed under control of the new licensing mechanisms (IT-Tude, 2010).

Figure 5: SmartLM

Advantages and inconveniences of these types of licenses

We will show that SmartLM provide a new generic licensing virtualization framework which is based on standards, integrated in major Grid middleware solutions, ready to be use on a Cloud environments and Service Oriented Architectures. SmartLM is focusing on developing both licensing technology suitable for the use in distributed environments like Grid and - together with ISVs - new business models for the use of licensed software in Grid or SOA environments. But SmartLM is not focusing only on enabling use of existing License technologies for Grids without addressing new License mechanisms or new business models. That can be seen for some as being a disadvantage for this License Management type (Towards SLA Based Software License Management in Grid, 2008).

Improvements

The major innovations of SmartLM are:

• Licenses may be used to run applications in Grid and Cloud environments no matter whether during the application run there is network
connectivity to access the site that host the License server that issued the license.

- SmartLM provides access to and management of all licenses owned by a site.
- SmartLM allows the definition of local policies for License usage addressing the site-specific needs.
- With SmartLM an accurate, user-specific price is calculated beforehand based on a large number of configurable parameters, like the time of usage, the features, the history of usage, local policies.
- In SmartLM budget limitations are checked and enforced when a user request a license.
- SmartLM realizes a number of sophisticated, state-of-the-art security mechanisms that render illegal use almost impossible.
- SmartLM offers re-negotiation of License terms at run-time, e.g. giving up a License before the reservation period is over, trying to extend a reservation period, or adding new features.
- Through SmartLM an Application Service provider (ASP) can temporally host the customer’s licenses allowing the execution of applications using the customer’s own licenses.

Moreover the above mentioned technical innovations, SmartLM introduce a new business model ‘Extension of License’ that enables users to extend their License on demand. Users can use the software and only pay per what they use. This is a crucial benefit for end users, especially SMEs, who usually are limited because they cannot afford to buy expensive licenses. Prototype will be ready the next spring; the consortium is willing to offer free trials to the first users, be one of them! Software licensing practices are limiting the acceleration of grid adoption. (Kent, 2005) Current technology is not sufficiently flexible to support Grids that access resources beyond the current administrative domain.

2.5.8.3 FlexLM

Today software developers rarely write their own License manager. The licenses commercially sold License Managers, such as FLEXlm (FLEXible License Manager) from GLOBETROTTER software (Mirabella, 2010). FLEXlm is the most famous License Manager used in the software industry. FLEXlm is also best known for its ability to allow software licenses to be available (or float) anywhere on a network, instead of being tied to specific machines. Floating licensing helps both users and License administrators. The reason is that users can make more efficient use of fewer licenses by sharing them on the network. License administrators can control who uses the licensed application and the node(s) where the licenses will be available. Molecular
simulations use the FLEXlm package from Globetrotter Software Inc. to License its products. Since this is a third party product, it is possible that you could receive software from another vendor who uses the same License manager. Keep in mind that certain topics (such as password encryption) are vendor-specific and proprietary so users cannot document them in any detail. Moreover FLEXlm does not enforce a particular licensing strategy; each vendor's implementation can have subtle differences (Welcome to FLEXlm, 2000). The four main components of FLEXlm are:

- **The License daemon** controls all of the License manager's functions. It reads the License file and launches all required vendor daemons. This License daemon is called LMgrd. It is important to use the highest version available. This release is currently using FLEXLM v5.12 LMgrd (Welcome to FLEXlm, 2000).

- **Vendor daemons** control the licenses for products from individual vendors. The molecular simulations vendor daemon is named MSI; you may have other vendor daemons at your site (Welcome to FLEXlm, 2000).

- **License file** contains all of the License information for all machines at your site (in the case of a floating or token license) or for a particular machine (in the case of a node-locked license). The Molecular Simulations License file is named msilicense.dat. Your License file may be e-mailed to you from Molecular Simulations or may be extracted during installation from the media. The system software is composed of License Manager, Execution Environment Manager, Visualization Support and Security (Welcome to FLEXlm, 2000).

- **Application program** using FLEXlm is linked with the program module (called the FLEXlm client library) that provides the communication with the License server. For example, the module used on Windows is called lmgrxxx.dll, where xxx indicates the FLEXlm version. The application program communicates with the vendor daemon, which controls the licenses for products from individual vendors, to request a License while the execution phase is going on (Application program using FLEXlm, 2010).

GLOBETROTTER estimates that FLEXlm has more than an 80% share of this market. The License file contains all site-specific information required by FLEXlm. This information includes:

- Server names and host IDs
- Vendor names and paths to vendor daemon executable
- Feature information
In general, the License file, or a copy of it, must be accessible to every machine that runs a FLEXlm-licensed application, and each machine designated as a License server. If the License file contains counted (also called "floating") licenses, before you can use the application you have to start the License Manager daemon (lmgrd) using the following syntax:

```
    lmgrd -c license_file_path -l debug_log_path
```

Where `license_file_path` is the full path to the License file and `debug_log_path` is the full path to the debug log file. Products are activated by a License file obtained through a registration, or more accurately a request-and-receipt process.

**Advantages and inconveniences of these types of licenses**

FLEXlm has become since 1989 the de facto UNIX industry standard for License management and the dominant License Manager due to its flexibility in creating licensing policies. Most of software vendors who use License management licensed FLEXlm. Over 11,000 products use FLEXlm today. The application program using FLEXlm is linked with the program module (called the FLEXlm client library) that provides the communication with the License server. On Windows, this module is called `lmgr.xxx.dll`, where `xxx` indicates the FLEXlm version. The application program communicates with the vendor daemon to request a License while executing the task.

**Improvements**

Currently there is no authentication in License management products for example with FlexNET. The BEinGRID project (Business Experiments in GRID) deals with licenses issues through a dedicated horizontal activity in one of its service clusters and tackles this problem. Other than the SmartLM solution, BEinGRID is targeting on enabling use of existing License technologies for Grids without addressing new License mechanisms or new business models. The approach is based on a proxy solution that transparently tunnels the communication between the remotely executed application and the FLEXnet server while at the same time making the FLEXnet server believe that it is talking to an application running locally.

**2.6 Analysis of processes**

This part of the thesis will focus on find a common ground between users and ISVs using one of the software licenses management types: SmartLM.
SmartLM is focusing on developing both licensing technology suitable for the use in distributed environments like Grid and - together with ISVs - new business models for the use of licensed software in Grid or SOA environments (Ziegler, 2009). The initial main goal for the accounting system will be to track VO members’ resource usage and to present that information in a consistent Grid-wide view, focusing in particular on Software License, CPU and Disk Storage utilization (Ziegler, 2009). SmartLM aims at fixing mechanisms for managing and using software licenses in a more fair and flexible way. SmartLM licenses may be used seamlessly in local cluster environments, as well as in local or remote Grid and Cloud environments, and under circumstances that the SOA concept presents (Ziegler, 2009). Just imagine 3 separate corporations working on the same researches but different one for another.

2.7 Requirements and importance of software licenses and licenses management

The License models in Grids should be up to date in order to allow a better transition from the current practice dealing with the challenges from technical, business and cost aspects. The following points identify these aspects in detail:

- **Different administrative domains:** License management may involve more than a single administrative domain. That why issues like e.g., Firewalls, remote usage control should be considered. Also, different usage policies might be defined for different administrative domains.
- **Transparent management:** Licenses should be transparently managed as part of the Grid job management.
- **Co-allocation of different resources:** Co-allocation of computing resources, Data and software licenses should be supported. Software licenses should be co-scheduled together with other kinds of services or resources.
- **Remote License enforcement:** Jobs may be executed remotely while the validity of the licenses has to be guaranteed at the same time.
- **Virtual organizations (VOs):** In the Grid virtual organizations are often dynamically built and their members or users need temporal access rights for specific software suites from different domains, so flexible that obtaining temporal licenses should be available.
- **Dynamicity:** The ability to suspend, preempt and resume the License use should be supported.
- **Interoperability:** License management should be integrated with the common Grid middleware. For example Glite (middleware stack for grid computing used by the CERN LHC experiments and a very large variety
of scientific domains), UNICORE (Uniform Interface to Computing Resources), which offers a ready-to-run Grid system including client and server software. Licenses and License management should be built on standards instead of proprietary solutions.

- **Costs:** The scalability of the Grid influences the costs using the existing License models. Models like paying per-CPU, per-Seat, per-Job may quickly become expensive in Grid environments, concurrent floating licenses across the Grid are also too expensive for the software users (Kent, 2005). Moreover, even if the price can be agreed upon, some issues remain, e.g., how the License usage will be audited and instrumented.

- **Support for workflows:** With the adoption of web services, service oriented architectures and BPEL, complex applications often are composed as workflows, which makes the License management more difficult. For instance, when executing a workflow, different applications may be used in different phases of the workflow. Finding a way to retrieve and reserve the right licenses for the applications in advance is one of the issues that need to be addressed.

- **Support for virtualization:** Licenses and License management for environments with virtualized resources or based on multicore technology is required.

- **Software is a service:** SaaS and on demand use as well as the utility pricing mechanisms should be considered both for the License models and License management in Grid environments. (Towards SLA Based Software License Management in Grid, 2008)
3 Accounting of software licenses in a Grid computing Environment

This section proposes methods for acquiring information about License usage for consumers and not to reduce software licenses prices to satisfy all user groups and categories. This method gives out also how to calculate the standardization technology cost of the usage of a job requiring software licenses and how to translate this standardization technology cost into currency cost. Furthermore, we analyze the demands of an accounting system in a computational economy based grid. Architecture of charging and accounting system and its support system is designed in this paper.

3.1 General Description

Grid computing requires the use of software that can divide and farm out pieces of a program to as many as several thousand computers. Grid computing can be thought of as distributed and large-scale cluster computing and as a form of network-distributed parallel processing. It can be confined to the network of computer workstations within a corporation or it can be a public collaboration (in which case it is also sometimes known as a form of peer-to-peer computing). Grid networks are used to compute extremely large data sets. While defining what large means in respect to data sets is continually evolving, there is an easy way to explain the basic need. Institutions that are currently working on cutting edge research in physics, engineering, and medicine are required to process huge amounts of data more quickly. In a Grid computing environment there are specifications and standards. There are 3 fundamentals that qualify this environment:

a) The resources which are the fundamental component of a Grid because of its ability to make more cost-effective use of a given amount of computer resources

b) The jobs on that resource which are fundamental consumers of resources because of its way to solve problems that can't be approached without an enormous amount of computing power

c) Jobs may be batched or interactive because it suggests that the resources of many computers can be cooperatively and perhaps synergistically harnessed and managed as collaboration toward a common objective.
The role of this section is to describe how software licenses are being used with the Grid resources and how to control their usage in a global point of view. This part of the study will try to find and propose solution for the use of Licenses in a grid computing environment. There will be 2 types of studies according to whether or not there is a License Manager in play. In order to provide an integrated and widely adopted approach to grid accounting and usage monitoring in real production Grids using software licenses, there must be continued efforts both on standards and the implementation, and also continued research and development activities in these areas. At present, there is significant diversity in the solutions adopted by different service providers and License owners, many adopting standards where they can, but also taking a pragmatic approach to provide the best solution for their own requirements. Once effort has been devoted to produce a customized solution, there will be reluctance to re-engineer even if new developments provide an improved framework, unless there is a clear benefit in doing so. Thus it is of great importance to prioritize and modularize the tasks to be performed, so that progress can be achieved incrementally, vendors can then decide when it is appropriate to do their own customization, and there is the flexibility to exploit new developments as they arise (Review of Accounting and Usage Monitoring, 2007).

There will be 3 separate entities here: The End user or client where the software is used, the vendor who owns the licenses and the provider who distributed the licenses and resources to the end user workstation. Let say there are 4 Companies which are connected to a supercomputer in a Grid environment. The ISV gets the software licenses which are resources for this purpose and sell them to the provider. The provider in return has a server and 4 CPUs in charge that he has to nourish with the data he acquires from the ISVs.

3.1.1 Scenarios without License Manager

Scenario #1 Individual Licenses
For instance, if 3 different researches institutions (N=3) were manipulating an huge access database stored on a file server in a supercomputer owned by a Biocomputing Central Lab using Microsoft office 2010 to make researches on AIDS and cancer. The institutions W1, W2 and W3 would need to have licenses L1, L2, L3 to use resources from the supercomputer. But in this particular case of a Grid environment, the licenses have to be assigned by a Grid administrator to fixed IP addresses once the software has been installed. If W1 shares his License L1 with a random institute W4 which is also in the same network, the License will not work because of the IP4 which does not match the Admin database. Another case will be also if W1 shares his License L1 with W4, and
W4 uses the L1 before W1, W1 will be denied access to the resources and W4 will be granted access (First Come, First Serve principles).

Scenario #2 Floating License/Group License
Going with the same example than above, the Institutions W1, W2, and W3 are only required to retrieve a huge access data file and report the actions taken by the users (Licensees). But in this case in order to control the utilization of license, system Administrator can distribute a limited number of copies of the License made to disk or tape. W1, W2, W3 have all the same copy of the License given by the system administrator. That is why it is called Floating License. The 3 Institutions are granted the right to use the software with this unique license. If a company W4 was not listed on the Biocomputing central lab’s group License database and a copy of the given License have been made, the access will be denied to W4. With this concept of software licenses system administrators were able to keep track of software utilization by limiting the copies of Licenses to hard disks that were directly attached to licensed CPUs.

Scenario # 3 Token license
In this scenario 3 Corporations W1, W2, W3 want to use resources from software called Flash Gordon Decompiler to work on Flash interactive content (.swf). Each of them purchases a License from a provider. Licenses L1, L2, L3

Figure 6: Scenario with Group License
are registered in a middleware Grid with a firewall. To use this software licenses on W1, W2, W3, a token can be used to digitally sign and/or encrypt SOAP messages on each the Institutions. A License proper to each of them is then assigned to each user. This means no exchange of licenses. Once provided to the institute, the License cannot be used again by on another machine in another institution. If W1’s system crashes with L1 in it, unless the hard drives have been formatted, it will still be available to the Licensee. Another case using the token security is the fact that W1, W2, W3 do not have any Licenses on none of them. In this case a firewall requirement is not playing an important role. L1, L2, L3 are on the Grid. If user1 from W1 want to use Flash Gordon information, he has a card with encryption, which is set to sign him in and also activate his License L1 on the Grid.

### 3.1.2 Scenarios with License Manager

**Scenario #1 per server License**

In this case the 3 corporations W1, W2 and W2 are used again. Servers and clients, the two components that have been brought up in section 3.4.6, will be used again. Each corporation possesses a server with different licenses. So that means W1 has a Location Home Register with a server S1 with a License L1, W2 has S2 with L2 and W3 has S3 with L3. These institutions have also clients C1, C2 and C3. Microsoft called them Client Access Licenses. Each Client C_i (i=3) computer can access a server. In this scenario each client requires a License. These corporations need some information regarding researches again aids and cancer. Therefore they have to use the CERN’s (European Organization for Nuclear Research) supercomputer and access their resources to allow their search application to fulfill its tasks. The supercomputer possesses a really huge and efficient server S that is divided in particular servers with a License Manager to control users sharing. If C1, 1, C1, 2 from W1 are connected one after another at different time of S’s servers via S1, the License L1, which is actually assigned to S1, is then identified and granted access to CERN’s resources.

**Scenario #2**

C1, 1 and C1, 2 from W1 are using 2 different applications with 2 totally different licenses for 2 similar jobs requiring same resources and same data sharing. C2, 1 and C2, 2 from W2 are also using the same principles. But S1 and S2 from 2 different organizations W1 and W2 are 2 separate servers. Both servers have a group License because there are partners. S accepts only one server request at the time and per resource. When C1, 1 and C1, 2 are requesting the use of resources straight from CERN’s supercomputer, they are granted access to information and are share them according to the License agreement
given by the License Manager on S. C₂₁ and C₂₂ are all W₂’s client and can also be allowed to use of the same license. But if S₁ and S₂ require from CERN supercomputer only one License L and S₁ and S₂ are connecting at the same time to the Grid, it will just recognize them as being 2 separate applications using 2 different licenses and will charge each of the companies individually.

Scenario #3
The Third case represents the limitation of License for a company. W₁, W₂ and W₃ are connected to CERN’s Network resources. The Grid License Manager allows only 3 Computers from Each Corporation to connect and use the resources according to the number of licenses acquired or bought. So C₁₁, C₁₂, C₁₃, C₁₄ are connected and using the CERN’s data. But the License Manager recognizes and grants only L₁, L₂, L₃, 3 accesses to the resources. The first clients to connect will be approved and the 4ᵗʰ one, L₄ will be denied. The specified limit for concurrent connections is reached; the server does not allow new users C₁₄ with new licenses L₄ to connect, unless enough users (including administrators) have disconnected to get below the specified limit. Or C₁₄ will fall under a new License agreement with 3 accesses too.

![Figure 7: Scenario with License manager and Clients](image)

Scenario #4: per seat licensing
Another case can be what Microsoft calls per seat licensing. Here there is only one License L from CERN’s Grid distributed to the 3 institutions W₁, W₂ and W₃. The 3 institutions are using the same License to access and share data from the grid. One Client License is assigned to each specific computer and memorized each client’s IP addresses that accessed the server. After C₁₁ a
computer acquired a License in the per seat mode, it can get into any network server running that the server product sometimes at no additional charge depending on the terms and agreements of the contract.

Scenario #5: Group License
If C1, 1, C1, 2 C1, 3 C1, 4 from W1 are using one computer and they are sharing this computer because of their shifts. CERN’s License Manager recognizes only the number of computers N1 and N2 used in W1 and not the number of users (4) accessing the resources from the Grid. But it can also be that C1, 1, C1, 2 C1, 3 C1, 4 are using N1, N2, N3, N4 computers in a computer lab U1. The License Manager will only allow the request on the machines and not on the users because their access to the resources has been facilitated. So C1, 1, C1, 2 and C1, 3 can share the same resources on the same computer and the U1 will just be charged or granted for one machine. The reverse is also possible when a user C1, 1 utilizes many computers at once. One machine N1is for the development of a software and N2, N3, N4 are different computer platforms for testing the application.

Scenario #6:
Another issue encountered in a grid environment is the fact that 3 institutions W1, W2, W3 for example having all 3 different Licenses L1, L2 and L3 and the CERN grid with his own License L which also different from the 3. CERN server has a firewall also to protect sensitive resources and data; CERN has a system administrator too who assigns rights and permissions for different tasks. In this case the firewall requirement is playing an important role. L1, L2, L3 are on the Grid. If C1, 1 from W1 or C2, 1 from W2 wants to use an application which needs CERN’s information, he will not be able to get to the resources and complete his task because the firewall will not allow different type of licenses than what the grid already has. It will create a conflict between both licenses L1 and L.
Figure 8: Scenario with Grid Admin and License Manager

Scenario #7: Use of two License modes
A company W1 uses per seat Licensing mode and the clients $C_{i,j}$ ($i=3$, $j=3$) are trying to utilize resources from the CERN’s Grid. CERN possesses a server and a License manager. CERN falls under the per server Licensing mode. The two mode used in this work are represented here. The fact that W1 has a per seat License does not guarantee access to CERN’s server that is already licensed in the per server mode, if it has already exceed its specified limit. The first diagram below shows how grid computing is done today. This technology is currently available. ElasticLM: A Novel Approach for Software Licensing in Distributed Computing Infrastructures, Wolfgang Ziegler, Fraunhofer Institute SCAI
Figure 4: ElasticLM

The second diagram shows how ElasticLM, a Software License Manager would operate on Grid computing and solving certain issues that were caused by the licenses.
4 Usage Records Format standards

The Open Grid Forum is a part of Grid community, which is responsible for setting standard formats and regulations. The group also provides Usage Records Formats and recommendations. It also defines an XML-format usage representation on a per job basis. The metrics listed in OGF-UR\(^{10}\) are divided into three main groups: base properties, differentiated properties, and extension framework (M. A. Pettipher, A. Khan, T.W. Robinson, X. Chan, 2007). Here are the different tables showing the Grid Accounting standard usage records.

4.1.1 Base properties

Base Properties define common usage metric requirements for a usage record like user and job identification (M. A. Pettipher, A. Khan, T.W. Robinson, X. Chan, 2007).

<table>
<thead>
<tr>
<th>Base Metric Name</th>
<th>Base Data Type</th>
<th>Semantic Clarification</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordIdentity</td>
<td>xsd:string</td>
<td>Uniquely defines a record in a set of all usage records among sharing communities</td>
</tr>
<tr>
<td>GlobalJobId</td>
<td>xsd:string</td>
<td>Global job identifier as assigned by a metascheduler or federation scheduler</td>
</tr>
<tr>
<td>LocalJobId</td>
<td>xsd:string</td>
<td>Local job identifier assigned by the batch queue</td>
</tr>
<tr>
<td>ProcessId</td>
<td>xsd:integer</td>
<td>The process id of the job</td>
</tr>
<tr>
<td>LocalUserId</td>
<td>xsd:string</td>
<td>The local user identity associated with the resource consumption</td>
</tr>
<tr>
<td>GlobalUsername</td>
<td>xsd:string</td>
<td>The global user identity associated with the resource consumption</td>
</tr>
<tr>
<td>JobName</td>
<td>xsd:string</td>
<td>The job name or application/executable name</td>
</tr>
<tr>
<td>Charge</td>
<td>xsd:float</td>
<td>Total charge of the job calculated by a (site-dependent) pricing system</td>
</tr>
<tr>
<td>Status</td>
<td>xsd:string</td>
<td>The completion status of the job</td>
</tr>
</tbody>
</table>

\(^{10}\) Open Grid Forum Usage Records:
WallDuration | xsd:duration | Wall clock time elapsed for the job
CpuDuration | xsd:duration | CPU duration summed over all processes of the job
EndTime | xsd:timestamp | The time at which the job completed
StartTime | xsd:timestamp | The time at which the job started
MachineName | xsd:string | A descriptive name of the machine on which the job ran
Host | xsd:string | The system host name on which the job run
SubmitHost | xsd:string | The system host name from which the job was submitted
Queue | xsd:string | The name of the queue from which the job was executed or submitted
ProjectName | xsd:string | The project associated with resource usage

### 4.1.2 Differentiated Properties

All the Resources are measured in this part of the work according to their Metric Name and data types (M. A. Pettipher, A. Khan, T.W. Robinson, X. Chan, 2007).

<table>
<thead>
<tr>
<th>Differentiated Metric Name</th>
<th>Base Data Type</th>
<th>Semantic Clarification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>xsd:positiveInteger</td>
<td>The amount of network resource used by the job.</td>
</tr>
<tr>
<td>Disk</td>
<td>xsd:positiveInteger</td>
<td>The storage used by the job</td>
</tr>
<tr>
<td>Memory</td>
<td>xsd:positiveInteger</td>
<td>The amount of memory used by all processes in the job</td>
</tr>
<tr>
<td>Swap</td>
<td>xsd:positiveInteger</td>
<td>The swap usage of the job</td>
</tr>
<tr>
<td>NodeCount</td>
<td>xsd:positiveInteger</td>
<td>Number of nodes used by the job. The definition of “node”</td>
</tr>
<tr>
<td>Processors</td>
<td>xsd:positiveInteger</td>
<td>The number of processors the job requested or used</td>
</tr>
<tr>
<td>TimeDuration</td>
<td>xsd:duration</td>
<td>Additional measure of time duration associated with the job</td>
</tr>
<tr>
<td>TimeInstant</td>
<td>xsd:dateTime</td>
<td>Additional timestamp property associated with the job e.g., time when queued</td>
</tr>
<tr>
<td>ServiceLevel</td>
<td>xsd:string</td>
<td>Quality of service associated with the resource consumption</td>
</tr>
</tbody>
</table>

### 4.1.3 Extension Framework

The extension properties are meant to be used for custom usage metric representations that are outside the norm defined by the base and differentiated
properties. The extension properties introduced within the OGF-UR are mainly used for encoding custom resource properties (M. A. Pettipher, A. Khan, T.W. Robinson, X. Chan, 2007).

<table>
<thead>
<tr>
<th>Extension Name</th>
<th>Metric</th>
<th>Base Data Type</th>
<th>Semantic Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>xsd:string</td>
<td></td>
<td>Extension property regarding the resource associated with the usage</td>
</tr>
<tr>
<td>ConsumableResource</td>
<td>xsd:string</td>
<td></td>
<td>Extension property regarding a measured resource associated with the usage</td>
</tr>
<tr>
<td>PhaseResource</td>
<td>xsd:string</td>
<td></td>
<td>Extension property regarding certain measured resource usage with phase unit attribute</td>
</tr>
<tr>
<td>VolumeResource</td>
<td>xsd:string</td>
<td></td>
<td>Extension property regarding certain measured resource usage with a storage unit attribute</td>
</tr>
</tbody>
</table>
5 Solutions and ideas for an accounting of software licenses

5.1 Solutions for accounting of software Licenses

The issues encountered with software licenses in Grid computing are mostly due to the lack of accounting mechanisms, which are not developed nowadays for such a huge and important task. In the following the thesis presents solutions of problems as described in section 3.1.2

5.1.1 Solutions to the Scenarios without License Managers

For example for the Scenario#1: Individual Licenses. The issue here is the License sharing and how to solve it when a Grid has no License Manager to monitor and verify the licenses of corporates and to see who is illegible and fulfilled the requirements to use the application. In order to grant access to the resources, IP addresses and institutions have to match. The vendors can then track the Licensees and know which companies can be granted the permission to communicate with the file server, utilize the lab’s resources, store and share their results. The DGAS mechanism is used for both cases. In this mechanism the Distributive resources are controlled and managed to satisfy the needs of licensees. It also targets at providing job-level resource usage metering in a client/server infrastructure.

Solution to the floating License issue: A limitation of Licenses (only 3 copies) has been already setup by the admin on how many people can have access to the resources. The new institute will need to purchase its own group License and fall under another contract requirement. This is important because by applying this method the software vendors were able to just make to count on how many copies of software for certain amount of companies they sold. For example our Grid with 20 PCs; Each CPU would have to obtain an identical copy of the License so that the software can run properly on the machine.

Solution to Token License issues: Here the help of a Token, which is activated once the installation is made, then recognizes the Institutions. The system administrator can then go back and check manually the encryption on the used token and determine which company is using or used the resources. In the second case, the card is used as a token and talks to the License database on the
lab’s supercomputer and is allowed to use of the resources. If U1 goes in W2 and uses their system with his card, L1 will still be activated. The reason being that the Grid administrator has registered U1 from W1 with L1. Here the institution identification does not play any role in this case. Tracking of licenses is quiet easy in this case.

5.1.2 Solution to scenarios with License Managers

Solution to scenario #1 per server License Mode: Here the License Manager then charges the company only once and enables the application to function. By using DGAS mechanism the License Manager in the CERN grid is able to recognize and distribute the needed resources according the virtual organization’s License. But the fact that there are 3 various licenses does not show how many processors have been used and where exactly the licenses are utilized. With this method CERN can then regulate the amount of resources they are sharing per servers and per needs. This scenario also showed that DGAS is based on a Grid Security Infrastructure (GSI). But to see and evaluate the amount of resources, data and CPUs used according to the distributed and granted licenses and the used software, Nimrod/G will be the appropriate accounting mechanism to utilize here. The reason being the choice on specific resources and time, which demonstrate the presence of a scheduling system.

Solution to scenario #2 Group License issues: For this scenario the License Manager will just focus on following characteristics: the single descriptive name for the group assigned mostly to companies or school with a large amount of users, specified number of per seat licenses assigned to the group and a specific list of users who are members of the group. The number of licenses L1 and L2 assigned should correspond to the number of computers in the licensing group. The system administrator of the company W1 will then give more users access than less. Just in case the number of users expected to connect is not exact. The License Manager will only focus on the devices that have been utilized for research and not the single user.

The method chosen here to fix this issue was to use a system administrator who will control, limit and monitor the number of access to huge resources for registered IP-addresses only. The DGAS mechanism is used for both cases. In this mechanism the Distributive resources are controlled and managed to satisfy the needs of licensees. It also targeted at providing job-level resource usage metering in a client/server infrastructure.

Solution to scenario # 4: This mode requires a Client Access License for each computer that will access a particular product on any server. Once a computer is
licensed for a particular product, it can be used to access that product at any computer (e.g. Windows NT Server). Multiple users like C1, 2 and C1, 3 with multiple licenses L2 and L3 can also log on to that single computer. The Grid will just recognized the computer or IP address; not the 2 users Licenses.

Solution to # 6 with administrator is presented below
The system administrator can be able to control the distributed licenses by opening the firewall, matching their Licensing with the one on the grid and allowing the access to those who fulfilled the requirements. With this solution the Admin is doing most of the License managing and distributing work to make the grid functions properly.

![Figure 9: Solution to Scenario #6 with administrator](image)

Solutions to # 6 with License Manager is presented
In this solution the 3 companies have 3 different types of Licenses. The firewall is accessible when the licenses are matching the License in the Grid. But L1 and L2 are not compatible with the one in the Grid and L3 is. So W1 and W2 have the possibility to use the resources of CERN and finish their task. W1 and W2 will connect to W3 and use W3’s resources to access CERN data. The License Manager will then be notified for only the company W3, match their License L3 with its database and charge them according to the agreement they have.
Another solution in the same path will be to create another License L’, which will be used only when a third party W1 and W2 using a bridge W3 to get to the information. With this method the License Manager can still track the number of users or machines using the resources and still controlling the flow.

![Diagram](image)

Figure 10: Solution to Scenario #6 with License Manager

Solutions to # 6 with Administrator and License Manager

This solution here has 2 steps:

1. Administrator role: The Admin role here is minimal. He can be responsible only for granting access to the resources by opening the firewall to the Licenses Owners, giving them rights to get to the Grid.
2. License Manager role: The License Manager role will be the most important because it will control, monitor and match the users Licenses with its database before allowing them to get to the Grid resources so they can use them for their applications or software.

Solution of the scenario with the two License modes used at the same time: CERN’s License Manager in that case allows connections with the licensed computers only if the licenses are per server licenses and not per seat. That means L1, L2 and L3 has to be per server Licenses. L from CERN will have to
be the one with per seat License in order for it to work. Therefore, you can connect to the sites only if there are per server licenses available. The License Manager here will have to facilitate this function. Otherwise it will not work. (See 3.4.6 for example)

Figure 11: Solution to scenario # 6 with Administrator and License manager

5.1.2.1 Usage Records Format

The usage Records Format adopted for this thesis is the XML format because it ensures that applications communicate using the same language, the same grammar and the same structuring of data. It helps also understand the solutions of the accounting problems by using encoding documents in machine-readable form.

Solution without License Manager

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ScenariowithoutLicenseManager id="1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi: noNamespaceSchemaLocation="Scenario.xsd";
file:/Users/Maurice/Documents/Research/ Software Licenses Management in a Grid computing environment.docx">
<End-User>Institutions</End-User>
<Provider> Biocomputing Central Lab </Provider>
<Admin >System Administrator</Admin>
<NodeCount>40</NodeCount>
<NodeCount>3</NodeCount>
```
<Token>Licenses</Token>
<TrackingMethod>IPaddresses recognition</TrackingMethod>
<TrackingMethod>Copy</TrackingMethod>
<TrackingMethod>Group License</TrackingMethod>
<TrackingMethod>Identification</TrackingMethod>
<RecordIdentity>NULL</RecordIdentity>
<JobIdentity>NULL</JobIdentity>
<UserIdentity>W1</UserIdentity>
<UserIdentity>W2</UserIdentity>
<UserIdentity>W3</UserIdentity>
<ClientIdentity>W</ClientIdentity>
<JobName>Individual Licenses</JobName>
<virtualMachineId>NULL</virtualMachineId>
<Charge>NULL</Charge>
<Status>NULL</Status>
<Disk>NULL</Disk>
<Memory>NULL</Memory>
<Swap>NULL</Swap>
<Network>NULL</Network>
<TimeDuration>NULL</TimeDuration>
<TimeInstant>NULL</TimeInstant>
<ServiceLevel>NULL</ServiceLevel>
<WallDuration>NULL</WallDuration>
<CpuDuration>NULL</CpuDuration>
</sequence>
<Processors>20</Processors>
<EndTime>NULL</EndTime>
<StartTime>NULL</StartTime>
<MachineName>NULL</MachineName>
<SubmitHost>NULL</SubmitHost>
<Queue>NULL</Queue>
<ProjectName>Scenario without License Manager</ProjectName>
</ScenariowithoutLicenseManager>

**XML Schema**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified" targetNamespace="http://schema.ogf.org/urf/2003/09/urf">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">Software Licenses Management in a Grid computing environment.</xsd:documentation>
  </xsd:annotation>
</xsd:schema>
```
Solution with License Manager

<?xml version="1.0" encoding="UTF-8"?>
<ScenariowithLicenseManager id="1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Scenario.xsd";
file:/Users/Maurice/Documents/Research/Software Licenses Management in a Grid computing environment.docx">
  <End-User>Institutions</End-User>
  <Provider>Biocomputing Central Lab</Provider>
  <Admin>System Administrator</Admin>
  <NodeCount>40</NodeCount>
  <NodeCount>3</NodeCount>
</ScenariowithLicenseManager>
<TokenTracker>License Manager</TokenTracker>

<TrackingMethod>IP addresses recognition</TrackingMethod>
<TrackingMethod>Copy</TrackingMethod>
<TrackingMethod>Group License</TrackingMethod>
<TrackingMethod>Identification</TrackingMethod>
<TrackingMethod>License specification</TrackingMethod>
<TrackingMethod>Database order</TrackingMethod>

<RecordIdentity>NULL</RecordIdentity>

<JobIdentity>NULL</JobIdentity>

<CPUs>N1</CPUs>
<CPUs>N2</CPUs>
<CPUs>N3</CPUs>
<CPUs>N4</CPUs>

<UserIdentity>W1</UserIdentity>
<UserIdentity>W2</UserIdentity>
<UserIdentity>W3</UserIdentity>

<ClientIdentity>W</ClientIdentity>

<JobName>Individual Licenses</JobName>

<virtualMachineId>NULL</virtualMachineId>

<Charge>NULL</Charge>

<Status>NULL</Status>

<Disk>NULL</Disk>

<Memory>NULL</Memory>

<Swap>NULL</Swap>

<Network>NULL</Network>

<TimeDuration>NULL NULL</TimeDuration>

<TimeInstant>NULL</TimeInstant>

<ServiceLevel>NULL</ServiceLevel>

<WallDuration>NULL</WallDuration>

<CpuDuration>NULL</CpuDuration>

</sequence>NULL

<Processors>20</Processors>

<EndTime>NULL</EndTime>

<StartTime>NULL</StartTime>

<MachineName>NULL</MachineName>

<SubmitHost>NULL</SubmitHost>

<Queue>NULL</Queue>

<ProjectName>Scenario with License Manager</ProjectName>

</Scenario with License Manager>

XML Schema

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://schema.ogf.org/urf/2003/09/urf"
/>
Software Licenses Management in a Grid computing environment.
<xsd:choice maxOccurs="unbounded" minOccurs="0">
<xsd:annotation>
<xsd:documentation>
<xsd:complexType name="Token">
<xsd:attribute name="value" type="xsd:string" use="required"/>
<xsd:attribute name="assignDefault" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="TokenTracker">
<xsd:attribute name="value" type="xsd:string" use="required"/>
<xsd:attribute name="assignDefault" type="xsd:boolean"/>
</xsd:complexType>
</xsd:documentation>
</xsd:annotation>
<xsd:element ref="urf:Disk"/>
<xsd:element ref="urf:Memory"/>
<xsd:element ref="urf:Swap"/>
<xsd:element ref="urf:Network"/>
<xsd:element ref="urf:WallDuration"/>
<xsd:element ref="urf:CpuDuration"/>
<xsd:element ref="urf:TimeDuration"/>
<xsd:element ref="urf:TimeInstant"/>
<xsd:element ref="urf:ServiceLevel"/>
</xsd:choice>
<xsd:choice maxOccurs="unbounded" minOccurs="0">
<xsd:sequence maxOccurs="1" minOccurs="0">
</xsd:sequence>
<xsd:element maxOccurs="unbounded" minOccurs="0" ref="urf:NodeCount"/>
<xsd:element maxOccurs="1" minOccurs="0" ref="urf:Processors"/>
<xsd:element ref="urf:EndTime"/>
<xsd:element ref="urf:StartTime"/>
<xsd:element ref="urf:MachineName"/>
<xsd:element ref="urf:SubmitHost"/>
<xsd:element ref="urf:Queue"/>
<xsd:sequence maxOccurs="1" minOccurs="0">
<xsd:element maxOccurs="unbounded" minOccurs="0" ref="urf:ProjectName"/>
</xsd:sequence>
<xsd:element name=" ScenariowithLicenseManager id" substitutionGroup="urf:Scenario" type="xsd:positiveInteger" use="required"/>
</xsd:complexType>
</xsd:element>
</schema>
5.2 Characteristics of a futuristic software License Manager

A good License Manager has to be:

1. **Flexible:** This means it has to be able to change or be changed to accommodate the needs of the provider, the vendor and the end-user. With this feature a License Manager can be inserted in any type of software.

2. **Standardized:** This means it has to be specific to a type of software or type of functionalities or operations, fit to all systems specifications according to the types. It will have to be an universal system. This feature will allow every country that possesses this technology to share, use and distribute the resources without having to create a conflict of interest to a third party. Everybody will be able to profit from this knowledge. So that it is no more differences between resources from researches made in private or public organizations and those made in schools for example.

3. **Smart:** It has to be easy to remodel itself every time a problem occurs or a new issue comes up. The License Manager has to be able to generate new licenses after researching all options available and feasible in all Grids. Then it can standardize itself to fit an universal requirement and needs. It will also be able to easily control the utilization of resources, manage equally the distribution of Licenses and track the rights and types of licenses movements.

5.3 Process of acquisition of licenses

With this new technology acquiring licenses will no longer be a big issue because of the License managers that will accommodate the licenses by usage. If a research company and a university which is based on researches want to use resources from a major center like CERN or NASA for example, the License Managers will be smart enough to categorize the License as being for researches purposes on a specific subject no more by institutions types.

5.4 Pros and cons of software License Manager

In this section the pros and the cons of License Manager will be discussed in details.
5.4.1 Pros of software License Manager

This new concept of License Manager will facilitate and ameliorate their role in term of maintenance. The most beneficiary of all will be the software vendors whose job will be made easier just by making the License Managers smart. Because of this option also the End-users and providers will have exactly what they required. It will ease vendors and providers roles by tracking and monitoring exactly the software licenses movements and utilizations. The expansion of the Grid environment will continue to grow and attract more users all around the world. Also by facilitating On-Demand Licenses and Pay-per-Use Licenses the vendors will Manager easily and better their Licenses and provide a variety of services to their End-users.

5.4.2 Cons of software License Manager

Despite the fact that the different actors’ roles will be facilitated, there will be disadvantages with this system. Many virtual organizations and countries, which will be favorable to this advanced technology, will have to form a lot of new qualified people according to their needs, to maintain the new features. Because of these new features added the technology we have nowadays cannot afford to track the amount of licenses that will be let to users. To use the Pay-per-Use License for example the vendors have to create a system that can allow the End-users to benefit only the amount of time they are using the software.
6 Summary and Future work

The grid computing world is expanding in an exponential velocity. This thesis has demonstrated and another aspect of software License managers in a grid environment. The Thesis also helped to understand and define the concept of software license. A differentiation has been made between this concept and other types of licenses. It has been also noticed that due to the lack of accounting mechanisms License managers cannot be used to track and monitor the distribution and services of licenses. In this thesis different scenarios have been exposed and resolved using the Distributed Grid Accounting System (DGAS). As usage records format XML format was used to show the amelioration and the solutions of the chosen scenarios. As future work on the grid computing environment a focus has to be made on the resources sharing between countries within or outside a Grid because the separation of resources on the will cause its destruction. If the grid resources are properly and equally shared in the whole world, it will motivated and add many countries into it and will grow. This technology will then permit the researches and the services to be made more efficiently and faster. The future work has to be made on creating an international grid that will be able to accommodate all nations and can be accessed from everywhere in the planet earth. A standard License Manager that can recognize all software licenses in the world with the specifications of each country. Another aspect that which needs to be developed is the job scheduling issue. In a grid computing system huge amount of data- and resources management are distributed and utilized every time. By solving the job scheduling issue the high performance in grid computing system it will be also solved (Grouping-Based Job Scheduling Model In, 2006).
7 Glossary

**ISVs:** *(Independent Software vendors)* People or companies who control and own the access, the uses and legal innovations of software.

**BEinGRID:** *(Business Experiments in Grid)* Research project funded by European Commission to create effective routes to develop and expand the idea of *Grid computing* across the EU and in the business models with researches using Grid Technologies.

**Grid computing:** A services-oriented architecture based on standard, open and general-purpose protocols and interfaces. It can be devised in 4 categories:

- Computational Grid with a Counting capacity
- Data Grid with only data
- Service Grid
- Resource grid

**Clusters:** Groups of loosely coupled computers configured together to work as a unit. *(Shared-nothing model, Shared-disk model, shared-memory architecture)*

**RPC:** *(Remote Procedure Call)* allows a program to call a procedure at a remote location across a network.

**SME:** *(small- to medium sized enterprises)* companies with limited budgets

**Resources:** Any physical or virtual entity of limited availability that needs to be consumed to obtain a benefit from it

**Web services:** Web based applications that use open, XML-based standards and transport protocols to exchange data with clients.

**OGSA:** *(Open Grid Services Architecture)* architecture for a service-oriented grid computing environment for business and scientific use, developed within the Global Grid Forum (GGF). OGSA is based on several other Web service technologies, notably WSDL and SOAP, but it aims to be largely agnostic in relation to the transport-level handling of
data. OGSA has been adopted as grid architecture by a number of grid projects including the Globus Alliance

**Software services:** are services that are provided to customers that make available licensed products and that display, run, access, or otherwise interact with these licensed products, whether or not the services provider receives a fee. Some examples include Web site or blog services, e-mail services, content delivery services, and customer relationship services. The customer receives software as a service instead of as a packaged product.

**APEL (Accounting Processor for Event Log)** A log processing application which is used to interpret gatekeeper and batch system logs to produce accounting records

**DGAS: (Distributed Grid Accounting System)** Distributed Grid Accounting System deployed in the Italian Grid Infrastructure

**GASA:** (Grid Accounting Services Architecture) Also commonly called GridBank, was created to allow the sharing of resources between multiples administrative domains and platforms and support unambiguous recording of user identities against resource usage

**SGAS: (SweGrid Accounting System)** SGAS is a computational Grid composed of six computer clusters with a total of six hundred CPUs.

**GRASP (Grid Based Application Service Provision)** A system infrastructure for business models for Application Service Provision (ASP), based on virtual organizations. Grid Services paradigm as a means of providing a timely and effective technological basis supporting the evolution of the ASP market towards a sustainable Utility Computing model

**GSAX (Grid Service Accounting Extensions)** an extensible OGSA accounting and logging framework. It is designed to provide a functionally modular accounting framework which can be expanded by adding or changing components, to allow use of accounting at many levels of application and user understanding, to provide information at different levels of granularity (from real-time information to data on a per-job basis), to integrate QoS and service-level agreements into the accounting framework, and at different levels, to be independent of any economic model, and to allow dynamic pricing stages.
**Nimrod/G:** A Grid and Cloud aware version of the Nimrod distributed computing middleware (part of the Nimrod toolkit). Nimrod/G can directly execute large-scale distributed parameter sweep and Monte-Carlo computational experiments.
# Table of Figures

FIGURE 1: RESOURCE SHARING IN GRID COMPUTING (CYRIL BRIQUET, PIERRE-ARNOUL DE MARNEFFE, 2006)................................................................. 23

FIGURE 2: LAYERED GRID COMPUTING ARCHITECTURE ............................................................................ 26

FIGURE 4: LICENSE MANAGER ARCHITECTURE ....................................................................................... 50

FIGURE 5: RELATIONSHIP BETWEEN USER, ISV AND RESOURCE WITH GENLM (NALPEIRON, 2009) ........................................................................... 54

FIGURE 6: SCENARIO WITH GROUP LICENSE ........................................................................................ 63

FIGURE 7: SCENARIO WITH LICENSE MANAGER AND CLIENTS ................................................................. 65

FIGURE 8: SCENARIO WITH GRID ADMIN AND LICENSE MANAGER ...................................................... 67

FIGURE 9: SOLUTION TO SCENARIO #6 WITH ADMINISTRATOR ............................................................ 75

FIGURE 10: SOLUTION TO SCENARIO #6 WITH LICENSE MANAGER ..................................................... 76

FIGURE 11: SOLUTION TO SCENARIO #6 WITH ADMINISTRATOR AND LICENSE MANAGER............ 77
9 Bibliography


Rob Byrom R.Byrom@rl.ac.uk, John Walk J.A.Walk.rl.ac.uk, David Kant D.Kant@rl.ac.uk. (2004, December 17). User guide for Apel - Accounting using PBS Event Logging. pp. 4-15.


