

# Grid Computing

Amr Rekaby<sup>1</sup>

<sup>1</sup>Egyptian Research and Scientific Innovation Lab (ERSIL), Egypt  
[rekaby0@hotmail.com](mailto:rekaby0@hotmail.com)

## I. INTRODUCTION

Grid computing is a new generation of distributed computing. The target of grid paradigm is how to construct strong processing power and storage resources by many small and weak resources.

Grid computing is a mesh of interconnected resources worldwide which constructs massive powerful capabilities. The user of the grid has the ability to use any (or many) of these interconnected resources in the grid to solve his problems, which cannot be solved by locally owned resources capabilities.

Grid computing is composed of many resources with different platforms and specifications (heterogeneous resources) not like the regular “Distributed Processing” which was always depending on similar resources (homogeneous resources). The high level structure of the grid is shown in Figure 1. Resources are grouped in local networks, organizations’ networks, universities’ networks, personal networks, etc. These networks connected at the end via the internet to construct the complete grid model.

The algorithms and methodologies that are used to manage security, task scheduling, resource balancing in a local network are called intra-grid scope or intra-grid methodologies, but the algorithms which handle the grid overall, the relation between networks and the common issues between networks are called inter-grid algorithms.

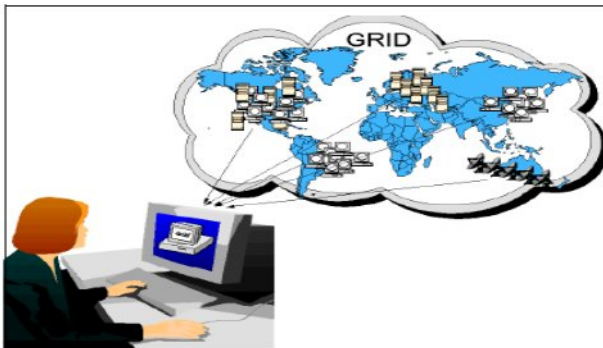


Figure 1 Grid World Wide View (source IBM) [5]

Grids are divided into two types:

- **Computing Grid:**  
Computing Grid is a collection of distributed computing resources. These resources are allocated in the user LAN or across different connected networks in the grid via an internet connection. The basic target in computing grid is constructing a powerful processing power. This could be done by putting the available resources in a pool and manage this pool to utilize all the resources, this utilization is depending on the running tasks in the grid and the status of the resources in the pool.
- **Data Grid:**  
It is a grid which concentrates on data storage distribution not a processing power. Scientific, medical and engineering applications require access to large storage areas –almost distributed- which can be extended to terabytes or petabytes.

Table 1 describes the brief comparison between grid computing and cloud computing from definition, types, components, and usage examples.

## II. ADVANTAGE OF GRID COMPUTING

In this section, a detailed view of the grid computing advantages will be described [5].



### A. Access To Inaccessible Resources

Under grid architecture and connectivity, the user can communicate with many resources, which are not accessible for him out of the grid umbrella. Before grid collaboration, huge organizations are used to try to capitalize their own resources to be able to get stronger/parallel processing power, larger storage space and so on. But in a grid environment, this organization can utilize the free or not fully utilized resources that are owned by other organizations. These resources may be not available to the client due to financial or political issues. The “resource” term in the grid doesn’t mean only hardware processing power, but also other hardware resources capabilities like devices, network connectivity, etc. This feature means that

there might be a user who has a task that needs specific network connectivity, so the grid is a perfect solution for this user if he finds what he looks for in it. The resources

also extend to be wider than just hardware. The resources may be installed software or even application license.

**Table 1** Grid computing VS. Cloud computing

Criteria	Cloud Computing	Grid Computing
<b>Definition</b>	Distributed computing methodology provides a cloud of services to be used by (thin or normal) clients.	Integration of computing resources, connected together to work as one massive computer power targeting a common goal.
<b>Types</b>	<ol style="list-style-type: none"> <li>1. Infrastructure as a Service (IaaS): Provide infrastructure storage resources as a service.</li> <li>2. Platform as a Service (PaaS): Provide a complete platform and its applications as a service.</li> <li>3. Software as a Service (SaaS): Provide specific software as a service.</li> </ol>	<ol style="list-style-type: none"> <li>1. Computing Grid: Integrates the processing power to construct huge processing capabilities.</li> <li>2. Storage Grid: Integrates the storage capacities to construct huge storage capabilities.</li> </ol>
<b>Components</b>	<ol style="list-style-type: none"> <li>1. Client (Thin or normal).</li> <li>2. Cloud Servers Infrastructure. Utility Computing (payment system).</li> </ol>	<ol style="list-style-type: none"> <li>1. Clients (Submitter/Host). Grid Managers (according to the grid model).</li> </ol>
<b>Components description</b>	<ol style="list-style-type: none"> <li>1. The client submits his work to the cloud servers.</li> <li>2. Cloud servers manage this work internally; client doesn't know where his work is done in the cloud.</li> <li>3. The cloud responses to the client after finish his work.</li> <li>4. Cloud servers internally could be established as distributed servers, parallel servers or even grid computing.</li> <li>5. Utility computing handle the provided service fees payment.</li> </ol>	<ol style="list-style-type: none"> <li>1. The client submit his work to the grid manager.</li> <li>2. The grid manager utilize all the resources which joined the grid to perform the submitted work.</li> <li>3. The client doesn't know where his work was executed.</li> <li>4. The grid manager replies to the client after finishing the work.</li> <li>5. Grid manager handle the usage of the resources fees (not commonly used step, depends on the grid usage).</li> </ol>
<b>Visualization</b>		
<b>Usage example</b>	Huge company like Google, IBM, and HP provide their services on a cloud to be public for users.	Universities and research labs integrate their available resources to get a strong processing/storage power.

<b>Applications</b>	<ol style="list-style-type: none"> <li>1. ERP systems.</li> <li>2. Public sector services.</li> </ol>	<ol style="list-style-type: none"> <li>1. Bioinformatics data storage.</li> <li>2. Chemical and nuclear calculations processing.</li> <li>3. Remote education services.</li> </ol>
---------------------	---	--

**B. Resource Utilization And Balancing**

While the resources in the grid systems are distributed on a lot of individuals or organizations sites, it is mandatory to have a central controlling system which controls the flow of tasks and their assignment in the grid. The running tasks on the grid environment have to be grid-enabled architecture. This "grid-enabled architecture" makes the task capable of being migrated from a resource to another due to the status of resources and the overall grid status [3].

The resources in the grid can be divided in any run time sample snapshot into three types:

- **Well utilized (balanced) resources:** The resources that have tasks approximately compatible with their capabilities.
- **Overloaded resources:** These are the resources which have tasks more than their capabilities. These resources should be handled by the central management through removing some of their assigned work. By this handling, these resources return into balanced state.
- **Under-loaded resource:** The resources that have tasks less than their capabilities. These resources construct a pool of free or unutilized resources. They will be the backup of any failed or overloaded resources, to migrate these tasks from the overloaded resources into one or more of these pool resources.

Figure 2 shows that if one of the resources becomes overloaded and contain tasks more than its capabilities, so some of overloading tasks should be migrated to other resources in another location.

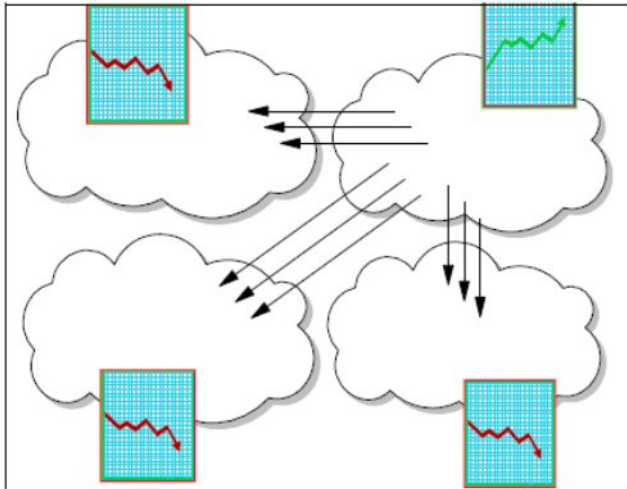


Figure 2 Load balancing and Task Migration (source IBM) [5]

Tasks migration operation has many constraints. The resources of the grid are distributed in different sites, and communication through inter-grid is depending on the

internet, so the internet communication bandwidth and network latency should be considered when thinking about tasks migration. But in intra-grid, the communication between resources depends on the communication way in this site (organization for example) and its bandwidth and hardware infrastructure.

The load balancing may be the magic solution for "time slot under-utilized" resources which are not used in defined time slot like universities and schools. All the universities resources are not used during the night, so these resources – which equivalents to money - are wasted in almost half of the day. If the grid system contacts the users (who have tasks) with providers (like universities and organizations which have unused resources), so we will finish users' tasks by utilizing the resources of these organizations, and the organization will get financial benefits income to themselves.

Grid architecture also introduces a new concept for the grid user, the user in the grid doesn't know where the real resources, that practically running his tasks are.

**C. Reliability**

When we are talking about single processor power working on tasks, so the reliability here is very low because if this processing fails for any reason (task related or non task related) such as electricity shortage, so the already finished work will be lost.

If these tasks are working on distributed environment in an organization, so the reliability here is higher, because the overall failure probability is less than a single processor and there is an option to migrate the failed tasks to another resource in the organization. But also this is not the best reliability ever because may be some conditions related to the organization or even city or country will affect this reliability. In a grid system, resources are distributed worldwide, so the reliability here is the best due to different geographical locations where resources exist in. The reliability of the grid is one of the responsibilities of grid manager, to be able to handle it in case of failure.

From the other side, the network traffic, communication and messaging add a new factor of failure that was not existing in the normal single processor model.

**D. Parallel Computing And Scalability**

One of the most powerful features in a grid environment is parallel CPU computing. The applications that are able to work in parallel mode have to be written in parallel programming logic, to be constructed by independent parts at the end. These parts can work simultaneously in more than one CPU to reduce the time needed to finish the task. These parts called "sub-jobs". The application scalability increases as much as the dependencies between the sub-jobs decrease [1].

If we suppose that the task will take ten seconds to be finished, and this job is divided into ten equal sub-jobs, we suppose that each sub-job works in separate CPU, so the task theoretically will be finished in one second. But if that happened, the scalability is 100% success while this percentage is not visible in real situations, so this task will take more than one second due to communication time and other factors such as utilization of these ten CPUs.

The application logic and algorithm are key factors in application transition to parallel computing. If this algorithm is limited to few numbers of sub-jobs, so the scalability here is limited to this number of sub-jobs, also the dependency between these sub jobs (Task Dependency Tree). Definitely the infrastructure factors such as network bandwidth and communication protocols have an effect on the scalability.

### III. WHAT CAN BE DONE BY GRID COMPUTING AND RELEVANT CONSIDERATIONS

#### A. Grid Computing Application Examples

In this section, we mention examples of applications which could use grid in their solutions [8]. While the main grid goals as we discussed before are how to provide processing power and data capacity storage which are not easily owned by the user, so the applications that need massive processing power or extremely large data storage space are the suitable applications for it.

The following points list sample of these applications:

- **Medical Imaging:** This is the application that stores a remarkably large size of medical images and most probably makes heavy calculations on them.
- **Scientific Applications And Simulators:** These are applications, that simulate scientific problems and solutions belongs to physics, chemistry and other scientific fields as Astrology, geology, etc. Most of these applications need extraordinarily strong processing power due to exceptionally complex equations and functions in them [6].
- **E-Learning Applications:** Nowadays many of educational and research activities are done by E-learning. The information sharing will be more easily in grid infrastructure than old fashion processing, while all the resources in the grid are accessible by all the users in the grid [2].
- **Drug Discovery:** These are applications that targeting drug discovery. Due to this large experimental data and strong computations, so the grid infrastructure is an excellent solution in this case and it is widely used in this field.
- **Microprocessor Design:** Microprocessor development life cycle is not straightforward life cycle, so the life cycle will be improved if some processing can be done to simulate the results without creating the product itself, this hardware simulation requires a lot of processing power, so the

grid architecture is particularly suitable for the grid computing.

#### B. General Application Consideration

Although a grid-based environment may offer many advantages, not all application get the benefit from a grid. For example, some personal productivity applications are tightly coupled with a user's interface and do not consume a large number of computing resources. Running them on a grid would not provide significant benefits. Vice versa the application may be affected negatively in a grid environment.

The grid as an environment provides access to vast amounts of computing power, one of the simplest concepts of grid utilization is the ability of running an application somewhere else when your own machine is too busy or the user's machine does not have the required capabilities. Almost all kinds of application could be executed in a grid environment accordingly. You may not see spectacular performance gains unless the hosting machine is much powerful than the machine you usually use.

Applications that can be run in a batch mode are the easiest to execute on other resources within the grid. Applications that need interaction through a graphical user interfaces are more difficult to run on a grid.

#### C. CPU Consideration

Probably the most critical step in application grid enabling is determining whether the calculations can be done in parallel or not. While High Performance Computing (HPC) clusters are sometimes used to handle the execution of applications that can use parallel processing, grids provide the ability to run these applications across a heterogeneous, geographically dispersed set of clusters. Rather than running the application on a single homogenous cluster, the application can take advantage of the larger set of resources in the grid.

Not all problems can be converted into parallel model. Figure 3 shows the effect that could be achieved by converting a serial application into parallel mode. The figure shows a dividing of serial logic into seven parallel tasks, which could be done simultaneously without any dependencies. The dependencies between the sub-tasks are the main factor of the decision, either the parallelism could be executed or not.

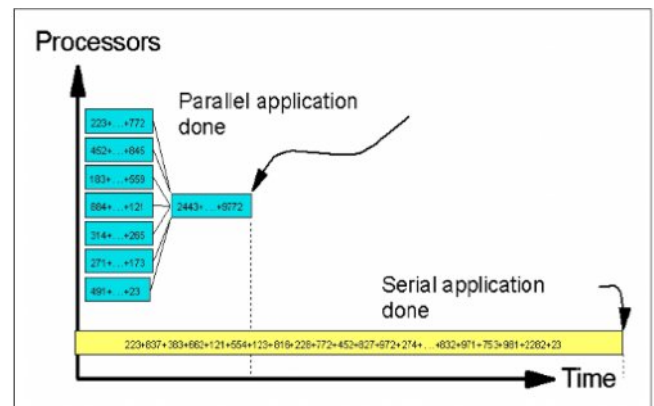


Figure 3 converting tasks execution into parallel way (source IBM) [4]



#### D. Data Consideration

Another factor of application consideration is data consideration. The task's data is the input and output information whatever they are stored in files or memory locations. The size of these data will be subject of transfer over the network in distributed computing model. Absolutely the most suitable form is minimizing the size of information that should be transferred either input or output of the task.

Duplicating information at many locations in grid nodes also is a challenging solution, which may reduce the traffic load if it is used efficiently, but it also opens other issues like updating this duplicate information, and how far the involved nodes will be notified.

Using a shared location for many grid resources also is a solution with pros and cons points.

The shared resources between grid's nodes face a challenging of resource locking. The locking feature could consider a timeout option, to release the resources which are locked for a long time.

#### IV. WHAT CAN NOT BE DONE BY GRID

After describing CPU and data consideration above, the applications that contain a lot of constraints such as sequential program flow (not match parallel mode), huge data to be transferred and user interaction centric applications are not suitable for grid solution.

#### V. SCOPE OF GRID COMPUTING SOLUTIONS

The scopes of the grid solutions are always extra layer above the operating system. There are many toolkits such as IBM Globus and Avaki which play this layer role above the operating systems to be able to manage the tasks scheduling, tasks transfer and other grid management activities. So the grid managers (whatever the role/responsibilities of this manager) are located as a horizontal layer over the operating system to manage the resource capabilities [7].

#### VI. REFERENCES

- [1] E. Ilavarasan / P. Thambidurai / R. Mahilmanan "Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System" Proceedings of the 4th International Symposium on Parallel and Distributed Computing (ISPDC'05) IEEE – 2005 – 0-7695-2434-6/05.
- [2] International Technical Support Organization "Grid Computing in Research and Education" – IBM Red Book – April 2005. [www.redbooks.ibm.com/redbooks/pdfs/sg246649.pdf](http://www.redbooks.ibm.com/redbooks/pdfs/sg246649.pdf)
- [3] International Technical Support Organization "Grid Services Programming and Application Enablement" – IBM Red Book – May 2004. <http://www.redbooks.ibm.com/redbooks/pdfs/sg246100.pdf>
- [4] International Technical Support Organization "Introduction to Grid Computing with Globus" – IBM Red Book – September 2003. <http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf>
- [5] International Technical Support Organization "Introduction to Grid Computing" – IBM Red Book – December 2005. <https://www.redbooks.ibm.com/redbooks/pdfs/sg246778.pdf>
- [6] Simon J. E. Taylor / Navonil Mustafee / Shane Kite / Stephen J. Turner / Steffen Straßburger "IMPROVING SIMULATION THROUGH ADVANCED COMPUTING TECHNIQUES: GRID COMPUTING AND SIMULATION INTEROPERABILITY"

Proceedings of the 2010 Winter Simulation Conference - 978-1-4244-9864-2/10 - IEEE - 2010.

- [7] The globus project materials "Introduction to Grid Computing". <http://www.globus.org/>
- [8] Viktors Berstis "Fundamentals of Grid Computing" – IBM Red Paper – 2002. <http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf>