

Users-Grid: A Unique and Transparent Grid-Operating System

Raihan Ur Rasool, Guo Qingping and Zhou Zhen

{raihan,qpguo}@mail.whut.edu.cn; zc_overwhelming@hotmail.com; shadi79h@hotmail.com.
Wuhan University of Technology, Wuhan, 430063, P.R.China.

ABSTRACT

Grid computing, which emerged as a result of rapid development in high speed networking is regarded as the prototype of next-generation cyber infrastructure for science. It enabled collective use of globally distributed resources and came up as a huge problem-solving environment. Besides the inherent challenges of distributed computing, it presents some new challenges including, how the research professionals, who are not computer expert and normal users who just want to get benefit from Grid without taking care of its complexities can be motivated to adopt new and more complex way of working. In this paper we have presented our idea and architecture of 'Users-Grid'. Utilizing Agent Technology and with its Service-Oriented architecture it will provide computing power transparently and easily. It gives the feature of true virtualization. It will be helpful to not only research communities and academia but also to business users.

Keyword: Grid Computing, Distributed Computing, Agents, Service Oriented Architecture, Resource Management System and Grid-OS.

1 INTRODUCTION

The Grid with all its fantastic promises presents another revolution, however like all revolution that has preceded it; we are also seeing our troubles multiply. While the Grid provides platform-neutral protocols for fundamental services, it lacks sufficient abstraction at the application level. A normal user spends too much time on procedures for accessing a resource and too little time on using the resource itself.

Scientists dream to make one great virtual computer with Grid technology by collecting resources of universities, supercomputers, national labs, etc. Let us step back and define virtualization. Actually 'virtualization is the process of presenting computing resources in ways that users and applications can easily get value out of them. In other words, it provides a logical rather than physical view of data, computing power, storage capacity, and other resources'. Until now we have been delivering this benefit of virtualization to users somehow, but not to applications. So we are missing an important point that existing applications need to be able to take advantage of Grid infrastructure transparently in order to get full virtualization. Abstraction of various Grid capabilities is needed so that they may be accessed easily from within an application, without requiring detailed knowledge about underlying fabric. We require new abstraction and concepts that allow applications to access and share resources and services across distributed, wide area network [1]

Grid technologies developed in the research community [2, 3] have addressed many important issues, however these efforts in most cases have been mainly focused on UNIX servers. There are many issues and in desktop environment which have not been addressed by such systems including dynamic naming, intermittent connection, untrusted users and so on. Further, such systems do not address a range of challenges unique to the Windows environment, whose major variants are the predominant desktop operating system [4].

We propose our unique idea namely a 'Users-Grid', an Open source Grid-Operating System. Based on Globus, using Service-Oriented architecture and Agent Technology, it gives the feature of true virtualization to users. Users-Grid enables automatic and seamless shift of extra load of computation from user's computer to the other nodes on Grid, which are underutilized and bill back the user according to the resource usage. In this way user will utilize Grid resources transparently without concerning how and where the job is submitted. This virtualized infrastructure allows applications to exploit heterogeneous resources across the enterprise. In Users-Grid all resources and computing capacity of Grid will be available whatever applications need it, whenever they need it, based on the applications' service needs. It will be able to seamlessly link disparate heterogeneous resources across platforms into a grid computing infrastructure.

The remaining sections of this paper present background and related work, ‘Users-Grid Architecture and working’ and conclusions.

2 BACKGROUND AND RELATED WORK

A brief listing of concerning software systems is provided here.

- Globus is an open-source initiative to produce a standard Grid architecture for distributed resources. Globus Toolkit [5] serves as the ‘de facto standard’ lower-level substrate for the Grid. Globus provides important Grid features such as authentication, authorization, secure communication and directory services. The Globus Toolkit is a collection of modules that provides standardized lower-level features for implementing a distributed system on the Grid.

However Globus Toolkit alone is insufficient, because there are some limitations of Globus Toolkit, such as

- ◊ It is crucial to understand that Globus has never been intended to directly address issues such as scheduling or load-balancing; these are the domain of higher and lower-level packages. While Globus provides APIs and libraries to ease implementation of these resources, it never has, and likely never will, provide an end-to-end solution for the Grid [6]. So it doesn’t have Job-schedulers (but it let’s third part schedulers to be used), hence no load balancing mechanism and no fault tolerance mechanism, it doesn’t provides broker function (it does, however, provide the grid information services function through the Monitoring and Discovery Service (MDS)).
- ◊ Jobs submitted on a Globus resource must have complete specifications: that is, they must explicitly state which Globus host they are to execute on, etc. No automated resource selection occurs without help from a package designed to provide such a service [7]. So it let’s you submit job at command prompt, by finding out free resources by RSL commands and to get results back by command prompt with RSL. This is great problem for normal user who is not willing to be a computer expert.
- ◊ In a web service model, the users are only responsible for accessing running services. The Globus Toolkit provides a service for remote execution of a job, but it does not

attempt to provide a standard hosting environment that will guarantee that the job has been executed correctly [8].

- ◊ Globus toolkit requires users to construct and maintain their own distributing computing environment. So it is ‘just a toolkit’, and we need to build the supporting environment around it.
- Condor [9] is a software package for executing batch jobs on a variety of platforms, in particular, those that would otherwise be idle. The major features of condor are automatic resource location and job allocation, check pointing, and migration of processes. Condor provides a job management mechanism, scheduling policy, priority scheme, resource monitoring and resource management. It has the facility of ‘ClassAds’, ‘Job checkpoint and migration’ and ‘remote system calls’. Condor-G [10] represents the marriage of technologies from the Globus and the Condor projects. Condor-G can be used as the reliable submission and job management service for one or more sites, the condor HTC system can be used as the fabric management service for one or more sites, and finally Globus Toolkit services can be used as the bridge between them. Condor-G can be used as the reliable submission and job management service for one or more sites. The Condor software consists of two parts, a resource-management part and a job-management part. Condor-G provides the job management part of Condor. It uses the Globus Toolkit to start the jobs on the remote machine instead of the Condor protocols. Although Condor is a mature project and has been used widely in academia and research laboratories, but
 - ◊ It has not feature of automatic shifting of extra load of computation from user’s computer to the other nodes on Grid and billing back to user seamlessly.
 - ◊ It is a Resource Management System, not a Grid-Operating System. It can be used as local resource management (like LSF, PBS, Load Leveler, SunGrid Engine, etc.).
 - ◊ It has no facility to enable existing applications to run on Grid without change in the code hence no plug-in support to applications (e.g. Matlab, Excel etc...) is provided.
 - ◊ Users have to learn, how to write Condor command to submit job on Grid in special format (`condor_submit job_description.sub`), so it lacks sufficient GUIs to hide the complex view of Grid from normal user.

- ◊ It is relatively difficult to install and configure. Although it has a script to install but even then it requires many manual changes in files (under Linux). Condor-G requires Globus to be installed by the user. And it is also the most difficult task for a normal user to first install Globus and its pre-requisites and then to install Condor-G. Offcourse for non-computer expert users or researchers who just want to run their simulations on Grid to get fast results, it is impossible to learn computer mechanics first and then to start their work.
 - ◊ Using Condor gives user ‘cluster computing’, one can use Condor as a local scheduler/RMS, but for Grid Computing a complete system is required that can be built using above two described projects (Condor and Globus).
 - The Sun Grid Engine (SGE) [11] is based on the software developed by Genias known as codine/GRM. A user submits a job to the SGE, and declares a requirement profile for the job. When the queue is ready for the new job, the SGE determines suitable jobs for that queue and then dispatches the job with the highest priority or longest waiting time; it will try to start new jobs on the most suitable or least loaded queue. It has open source as well as commercial versions. It is also a RMS.
 - ◊ The software has been ported to many operating systems, including Solaris, Linux, IRIX, Tru64, AIX, HP/ux, but it is not supported (open source SGE) on Windows [12].
 - ◊ It is batch processing only, you don’t get rich-API support which makes easy to integrate it into your existing application environment.
 - ◊ It also has no facility of automatic load shifting and Plug-in to make existing applications to run on Grid. Hence it is not a complete solution.
- PBS and LSF also fall in this category, these are commercial products. One version of PBS (Open PBS) is claimed to be open source in many documents but it is no more downloadable [13]. Maui and CSF are also in the list of these successful and mature schedulers, however these batch processors and schedulers share the common limitations with SGE as described above.
- Vega PG [14], its main idea is to construct the Grid as a virtual computer in wide area. One important feature of the Vega PG is the user-centered philosophy, that is, Grid users can utilize Grid resources as their own will. On the contrary, the Vega PG adopts a lightweight architecture, which uses simplified structures and protocols. Vega maps between the Grid and traditional computer systems and use the virtual computer concept to build the Vega PG on legacy systems. It borrows many mature technologies from the traditional computer architecture design. However ,
 - ◊ It has also no facility to enable existing applications to run on Grid without code change, hence no plug-in support to applications is provided.
 - ◊ It has no feature of automatic shifting of extra load of computation from user’s computer to the other nodes on Grid and billing back to user seamlessly.
 - Legion/Avaki is an object-based system composed of independent objects, disjoint in logical address spaces that communicate with one another by method invocation. Legion/Avaki is comprehensive Grid software that enables efficient, effective and secure sharing of data and computing power. Its aim is to provide single system view to the users.
 - ◊ Legion/Avaki is a commercial product. Besides other few limitations, it has no feature of automatic shifting of extra load of computation from user’s computer to the other nodes on Grid and billing back to user seamlessly. Moreover it lacks Service Oriented Architecture.
 - Nimrod-G is a Grid broker that performs resource management and scheduling of parameter sweep and task-farming applications [15, 16]. Nimrod-G supports user-defined deadline and budget constraints for scheduling optimizations and manages the supply and demand of resources in the Grid using a set of resource trading service called GRACE [17]. Nimrod/G is implemented on top of Globus
 - ◊ It is a successful broker; but it is not a complete solution because it has no above described facilities.
 - UNICORE [18] include a uniform and easy to use GUI, an open architecture based on the concept of an abstract job and consistent security architecture. UNICORE provides an interface for job preparation and secure submission to distributed supercomputer resources. The UNIORE client enables the user to create, submit and control jobs from any workstations or PC on the internet. It is platform independent, Open Source software, but still is also not a complete solution because,
 - ◊ The current implementation does not support single sign-on [19]. It creates complexities for normal user.

- ◊ It also has no facility of automatic load shifting and Plug-in to make existing applications to run on Grid
- Netsolve is a client-server system that enables user to solve complex scientific problems remotely. The system allows users to access both hardware and software computational resources distributed across a network. A load-balancing policy is used by the Netsolve system to ensure good performance. Ninf and Netsolve are implementations of the GridRPC programming model.
- Ninf is an ongoing global networking computing infrastructure project that allows users to access computational resources including hardware, software and scientific data distributed across a wide area network with an easy-to-use interface
 - ◊ Ninf and Netsolve themselves are mutually interoperable, but they do not interoperate well [20] with other Grid tools on Globus such as GridFTP.

Now we introduce projects which make existing applications to run on Grid.

- Active Sheets, it is a mechanism for parallelising traditional spreadsheets, it is component based interface for spreadsheet. It implements a two stage evaluation process for custom functions, however
 - ◊ It works with Nimrod and Netsolve grid middleware systems and provides parallelism at the spreadsheet function level.
- Inner Grid Nitya, It is a commercial product from GridSystems company, it grid-enables Excel, by distributing spreadsheet calculations among corporate PCs.
- Platform Symphony introduced the Adapter for Microsoft Excel which aims to accelerates online and batch operations for business critical financial services.
 - ◊ Platform Symphony is specifically designed for Enterprises, to pre-allocate number of clusters.

However,

- Three above described systems are not open-source; they do not provide rich API support for other existing applications, because these are neither a RMS nor complete Grid-Operating Systems.
- These projects are mainly focused on enterprise level instead of taking in mind the global distribution of load across the Grid.
- These commercial softwares mainly only focus on MS Excel to enable it to run on

Grid, without concern to other many important existing applications.

Non of the above described system has the feature of automatic shifting of extra load of computation from user's computer to the other nodes on Grid, which are underutilized and bill back the user according to the resource usage. Moreover none of the above described systems allows you to take your existing business critical applications (not only MS Excel) and run them on the grid without minor changing the code. On the other hand Users-Grid is an easy to use and totally transparent complete system which is not just a toolkit or RMS. It contains a complete accounting and billing system based on resource usage. Traditional solutions involve a lot of development and support, these approaches don't completely address resilience, scalability, parallel computing, reporting and administration, Users-Grid is single proposed solution which captures all of these requirements in one robust solutions.

3 Users-Grid VISION AND ARCHITECTURE

We need a complete solution not just a toolkit. We need such an infrastructure which should provide a real-time operating environment, should optimize applications' performance (speed), reliability and resiliency, scalability and costs. The software should act as a common, virtual layer to fulfill the processing needs of multiple applications. Given many applications are competing for resources; it should meet the application's execution needs by balancing processing requirements and business policies/priorities with resource coordination in a virtualized environment. For understanding purpose two scenarios are presented below,

(without Users-Grid):- Suppose person 'A' is business-user, working on his computer, he has opened many softwares including MS.Excel and Matlab, he is in lack of resources on his computer, so he is feeling much difficulty in work because of slow working speed of his computer. He is connected with a Grid but he doesn't know in which format and how to submit the jobs (after finding free resources) on the Grid to take advantage of. So Grid is useless for him even having Globus or Condor pre-installed on his computer.

(with Users-Grid):- If this user has Users-Grid installed on his computer and he is already connected to Grid then whenever he would be in lack of computational or data resources, Users-Grid would seamlessly transfer extra burden of computation on other computers on Grid. Users-Grid would provide

plug-in support for even MS.Excel users to transfer bulky spreadsheets computation on other computer and having results back. And depending on administrator's policies the user will be charged bill according to his resource usage. The user can also use Users-Grid GUI facilities to submit the jobs on the Grid and see the progress. The Users-Grid resource management system would automatically take care of which resource is suitable to run the job. It will automatically make scheduling policies.

In this way we can get benefit of Grid, no matter what type of application/job flow is it (parallel, serial or networked), by either shifting to whole job or sub jobs/tasks onto any other computer or to set of computer to run in parallel fashion (depending on the nature of job). Users-Grid uses Agent Technology to know about extra computational burden, or free resources on any computer onto Grid. Agents cooperate with each other to balance workload in the global grid environment using service advertisement and discovery mechanisms. Both local schedulers and agents can contribute to overall grid load balancing, which can significantly improve grid. The proposed architecture of Users-Grid is presented in figure 1.

3.1 Users-Grid Base Layer

Users-Grid actually uses three already developed and mature technologies from the grid community. It uses Globus Toolkit and Condor at its Users-Grid Base Layer. In addition to these two already developed systems, it uses an accounting system that tracks resource usage. We chose these three systems on the basis of maturity, being open source and having implementation on many known operating systems. Users-Grid Base layer executes applications in a scalable manner. At this layer we are using three already developed technologies together which are,

- Globus Toolkit
- Condor
- Gold, Accounting and Allocation manager

So by using this architecture we used existing technologies rather to re-invent the wheel. We have to integrate and make them work together.

3.1.1. Globus:

Globus is the important part of Users-Grid Base layer; here we give brief description of the most relevant Globus services to Users-Grid,

- GSI: Grid Security Infrastructure serves as the common authentication facility underlying all the features of Globus. It enables single sign-on using certificates delegation based on PKI.

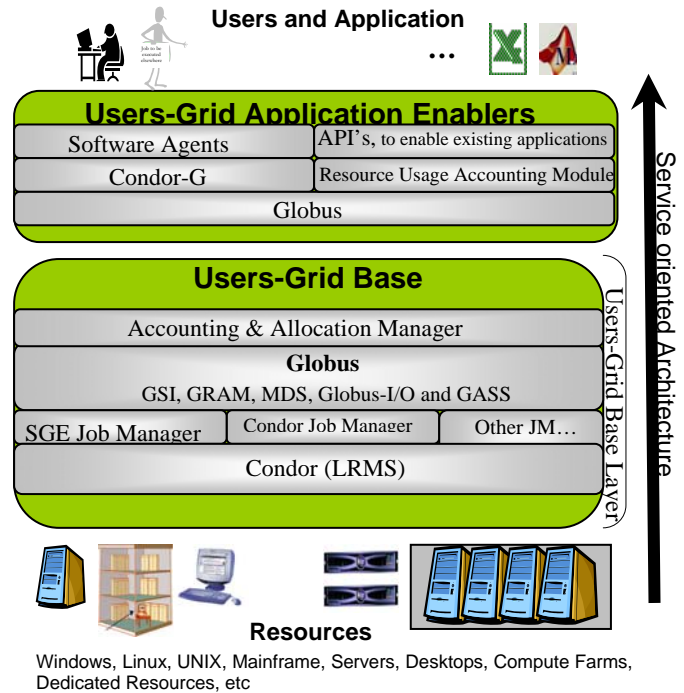


Figure 1 Users-Grid Layered Architecture

- GRAM: Globus Resource Allocation Manager is a 'secure inetd' that authenticates clients using GSI-based certificates, maps to the local user account, and invokes executable files.
- MDS: Monitoring and Discovery Service is a directory service that provides resource information within the Grid. MDS consists of two layers of LDAP servers, GIIS, which manages project-wide information, and GRIS which is responsible for site local information.
- Globus-I/O: Globus-I/O enables secure communication between Grid peers using GSI, providing standard read, write APIs, and non-blocking I/O that integrates with the Globus Threads library.
- GASS: Global Access to secondary storage provides easy-to-use tile transfer facility across Grid peers. GASS can be used in conjunction with GRAM to stage client side files

3.1.2. Condor:

Condor is a specialized job and resource management system (RMS) for compute intensive jobs. The combination of the Globus and Condor software packages presents an attractive solution for Grid applications. Globus includes the ability to use Condor as a back-end scheduler. Users may submit jobs to a single Globus resource, which is actually a Condor cluster. Then, Condor sends that job to whichever machine is available. Meanwhile, the user on the submitting end can remain blissfully unaware that Condor is being used [7].

Globus introduces the concept of jobmanagers, which allows clients to request that the Globus gatekeeper direct certain jobs to different targets. By default, only a ``fork'' jobmanager is included. Fork simply causes a job request to be immediately run (using, oddly enough, a fork() system call) on the Globus machine. Support for a Condor jobmanager is included with the Globus distribution but is not configured by default. We integrate Jobmanagers of Condor and SGE in our Users-Grid Base Layer.

Condor-G speaks Globus protocols natively; a Condor-G client will not treat Condor pools made available via Globus any differently than it would any other Globus resource. Submitting Globus jobs using Condor-G provides a much higher level of service than simply using 'globus-job-run'. If the job crashes while running remotely, Condor-G can submit it again without user intervention. Condor-G provides a level of service guarantee that is not available with 'globus-job-run'. Condor-G does hold several advantages over simply using Globus directly; however, the most important advantage is its support for DAGMan. The other significant advantage is that it provides a facility for submitting jobs to both Condor pools and to any resources running Globus, regardless of whether those Globus resources are running PBS, LSF, or have no batch scheduler at all.

Globus tools, wherever possible, behave in extremely simple, predictable ways. If a 'globus-job-run' command cannot proceed because the network is down, it will quit immediately with an error message. While this may be useful for custom scripting, it is not immediately clear how to submit jobs when reliable, once-and-only-once execution is desired. Scripting a solution is not easy because, in the event of failure, any number of components between the client and server may end up in unpredictable states from which it may or may not be possible to recover.

Sun Grid Engine or any other local resource management (LRMS) software can be used, however we choose Condor because of its wide range implementations in academia and industry. The remote host is not required to run any Condor software; rather, jobs are submitted to the remote host's Globus job manager in the usual way. In the following Figure, we show Condor as our LRMS.

Condor runs as a set of independent daemon programs, each of which performs specific tasks in either tracking machines or tasks, or in matching tasks with machines. So, the functionality of the various machines in a Condor pool may be broken into three categories: submit hosts, execute hosts, and master host. In following figure 2, we present how condor fits into Users-Grid architecture with Globus.

Condor; combined with the security, authentication, and encapsulation of Globus [7].

3.1.3. Gold Accounting and Allocation manager:

Gold is an open source accounting system that tracks resource usage on High Performance Computers. It acts much like a bank in which accounts are established in order to pre-allocate which users and projects can use resources on which machines and over which time frames. Gold supports familiar operations such as deposits, withdrawals, transfers and refunds. It provides balance and usage feedback to users, managers, and system administrators [21].

Gold can be used as a real-time debiting system in which jobs are charged at the moment of completion. When used in a grid environment, Gold facilitates trust by allowing lending organizations to manage what the costing rules are for usage of their resources and job submitters to determine how much their job is going to cost them before they start, ensuring all parties can agree to the transaction and giving each party a first-hand accounting record [21].

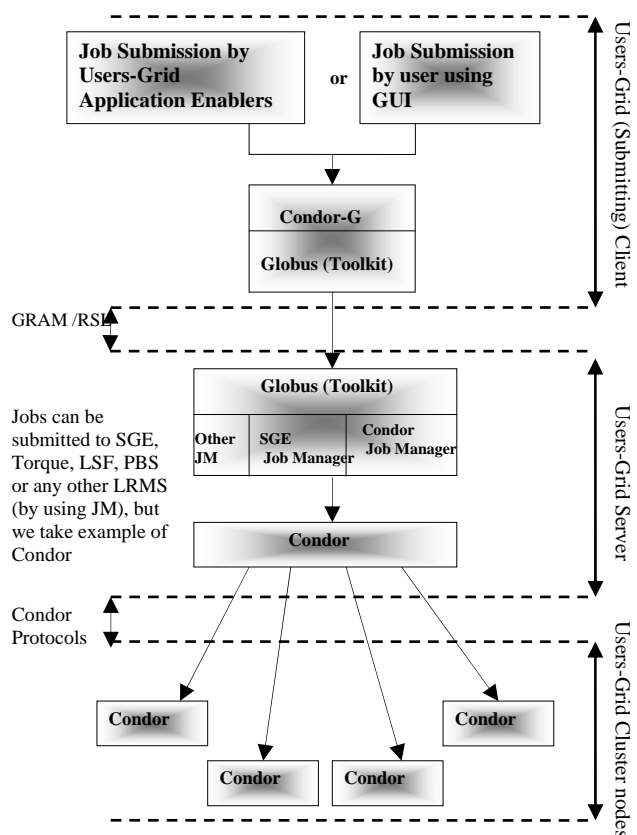


Figure 2 Users-Grid-Globus-Condor topology, how condor fits in this scenario

It is highly customizable and flexible. It provides scalability, security and maintainability. It can also provide a web-based interface to users. Considering its features and characteristics we found it best to be used with Users-Grid after few changes in the code.

3.2 Users-Grid Application Enablers Layer

An application is said to be grid-enabled when it can simply run in a grid. ‘Users-Grid Application Enablers’ layer enables existing applications to run on Grid. It provides seamless and transparent access of Grid resources to existing applications like MS Excel, Matlab, etc. It provides rich APIs support which is utilized by Agent to analyze the ‘application flow’, Job types and tasks within an application.

Some like to think of a grid as a distributed cluster. Although such parallel applications certainly can take advantage of a grid, nobody should dismiss the use of grids for other types of applications as well. Even a single threaded batch job could benefit from a grid environment by being able to run on any of a set of systems in the grid, taking advantage of unused cycles. A grid environment that can be used to execute any of a variety of jobs across multiple resources, transparently to the user, provides greater availability, reliability, and cost efficiencies than may exist with dedicated servers [22]. Similarly the need for large amounts of data storage can also benefit from a grid. Examples include thoroughly analyzing credit data for fraud detection, bankruptcy warning mechanisms, or usage patterns of credit cards. Operations on vast amounts of data by uniform calculations, such as the search for identifiable sequences in the human genome database, are also well suited for Grid environments. [22]

To take advantage of multiple resources concurrently in a grid, we have to consider whether the processing of the data can happen in parallel tasks or whether it must be serialized and the consequences of one job waiting for input data from another job. At this stage it is also very important to know about Application flow and Job flow. Application flow is the flow of work between the jobs that make up the grid application. The internal flow of work within a job itself is called job flow. So analyzing the type of flow within an application delivers the first determining factor of suitability for a grid. This does not mean that a complex networked application flow excludes implementation on a grid, nor does a simple flow type determine an easy deployment on a grid. Rather, besides the flow types, the sum of all qualifying factors allows for a good evaluation of how to enable an application for a grid. There are three basic types of application flow that can be identified: which are Parallel, Serial and Networked [22].

Although comprehensive details of application flows, Job types (for the suitability to Grid) can be found at [22], but it is necessary to discuss how we can do parallelization in serial application flow. In this case there is a single thread of job execution where each of the subsequent jobs has to wait for its predecessor to end and deliver output data as input to

the next job. In serial flow, the advantages of running in a grid environment are not based on access to multiple systems in parallel, but rather on the ability to use any of several appropriate and available resources. As each job does not necessarily have to run on the same resource, so if a particular job requires specialized resources that can be accommodated, while the other jobs may run on more standard and inexpensive resources. The ability for the jobs to run on any of a number of resources also increases the application’s availability and reliability. It is better to check whether the single jobs are really dependent of each other, or whether due to its nature they can be split into parallel executable units for submission on a grid [22].

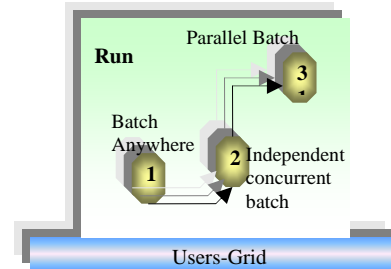


Figure 3 Enable an existing application to run on Grid (without change in code).

For example, when dealing with mathematical calculations the commutative and associative laws can be exploited. In iterative scenarios (for example, convergent approximation calculations) where the output of one job is required as input to the next job of the same kind, a serial job flow is required to reach the desired result. For best performance these kinds of processes might be executed on a single CPU or cluster. An application may consist of a large number of such calculations where the start parameters are taken from a discrete set of values. Each resulting serial application flow then could be launched in parallel on a grid in order to utilize more resources [22].

For a given problem or application it would be necessary to break it down into independent units. To take advantage of parallel execution in a grid, it is important to analyze tasks within an application to determine whether they can be broken down into individual and atomic units of work that can be run as individual jobs [22]. As grid standards such as OGSA mature, grid enablement will mean that the application can run as a Web service in a grid environment, while optionally taking advantage of the various services provided by the grid infrastructure. For example, Java 2 Platform, Enterprise Edition (J2EE) or C application run as a Web service on top of grid-enabled middleware, such as the upcoming version of ‘WebSphere Application Server’, while optionally taking advantage of the

services provided by the middleware and the grid infrastructure [23]. Service-oriented architectures and Web services are becoming the substrate on which strategic grid environments are based.

We use few strategies for grid enablement. For instance, an existing application can be enabled in a way that its most resource-intensive computational pieces run as a Batch Anywhere job on any suitable computer in the grid. The 'Users-Grid Base' decides which machine to assign at run time. Then, by eliminating concurrence inhibitors, the same application can evolve so that its grid-enabled piece runs as an Independent Concurrent Batch instance of the same batch job. It supports multiple independent instances of the same application running concurrent. This gives the overall application far more throughput than having only one instance running at a time. It could eliminate more concurrence inhibitors and allow its grid-enabled piece to run as a Parallel Batch job. Notice that what might be changing is the definition of a work unit, not the application. A work unit in this case would be an array of the original work units used to implement the first two strategies for enablement [23]. Two components must be added: one subdivides the work and hands it to the grid. The other aggregates results. In Users-Grid Agent aggregates the result with the help of APIs. The following figure 3 describes these steps.

Another possibility is to leave the batch world behind. But this strategy requires change in the code of existing application. An agent uses Users-Grid's APIs to enable many existing applications to get benefit from Grid, it is necessary to describe what an agent is and how it works.

3.2.1. Agent Works in Users-Grid:

An agent can be defined as an autonomous, (preferably) intelligent, collaborative, adaptive computational entity. Here, intelligence is the ability to infer and execute needed actions, and seek and incorporate relevant information, given certain goals [24]. While agents can be as simple as subroutines, typically they are larger entities with some sort of persistent control (e.g., distinct control threads within a single address space, distinct processes on a single machine, or separate processes on different machines). The salient feature of the language used by agents is its expressiveness. It allows for the exchange of data and logical information, individual commands and scripts (i.e., programs) [25].

So in Users-Grid agent resides like a daemon in user's computer and monitors the resource usage continuously and whenever resource usage of user's computer is more than pre-set limit (for a specific period of time, according to policies by user), it does the following with the help of Users-Grid's APIs,

1. Check the types of applications currently running and analyze the type of flow within an application.
2. Analyze the task within an application.
3. Is there any one which can be enabled to run on Grid? If yes, then
4. Analyze Job criteria (Batch Job, Interactive Job, Standard Application, Parallel Application)
 - a. Either, analyze to determine how best to split the application into individual jobs, maximizing parallelism and then break the application's computation into independent, individual and atomic units of work that can be run as individual jobs, and submit to 'Job submit Module' of 'Users-Grid Application Enablers layer'.
 - b. Or, shift the whole job (instead of splitting it) to remote node, which has the facility to run this job, with the help of 'Job submit Module' of 'Users-Grid Application Enablers layer'.

4. HOW DOES Users-Grid WORK?

As Users-Grid architecture includes Condor and Globus, it will include separate Server and Client installations packages. 'Users-Grid client' can be configured as both 'submit' and 'execute' machine. 'Application Enabler Layer' is very important to be included; however 'Users- Grid Base layer' would not be included in Client installation package.

For server installation, many optional components of 'Application Enabler Layer' may not be installed. Users-Grid can submit the jobs to other nodes of the grid even if those nodes do not have Users-Grid installed. But these nodes should have Globus and local Resource Management Software installed already. In following figures 4, 5 and 6, we have presented the possible three different scenarios to use Users-Grid, the diagrams are self descriptive.

4.1 Running Existing applications on Grid

We want to show that how can we enable existing applications to run on Grid, so we take an example of MS Excel for simplicity. MS.Excel is a comprehensive tool for financial analysis and business modeling. Many organizations use this spreadsheet application for performing analysis, simulations and modeling. While Excel provides a powerful and flexible environment for building formula and macro-based applications, the performance of these applications are limited by the fact that Excel runs in a desktop personal computer environment. To accelerate performance of these applications, Users-Grid provides the ability to run Excel spreadsheets in parallel on a grid environment.

Users-Grid allows Excel applications to run on a grid with little or no modification. Users-Grid provides plug-in support to Excel. So user can use Grid's resources right from his desktop, whenever he needs.

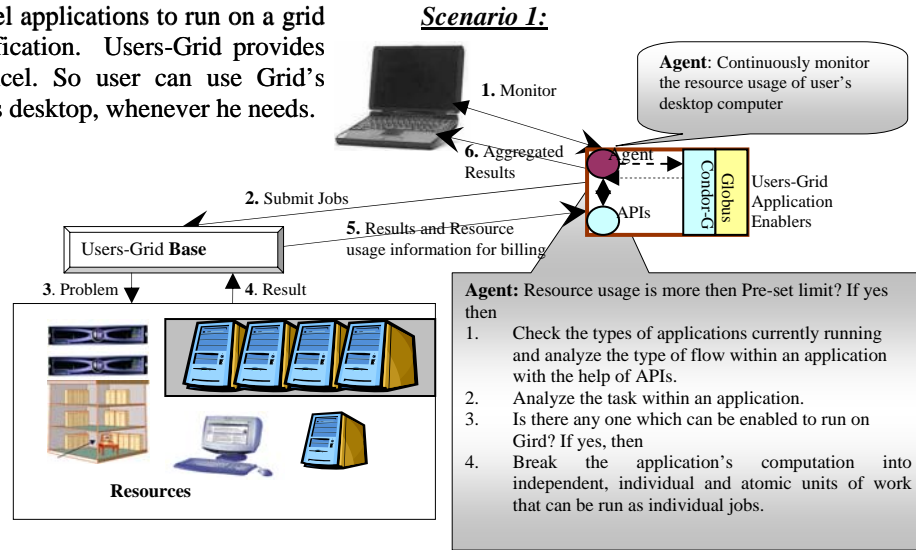


Figure 4 Automatic Shifting of extra computational load on other nodes of Grid

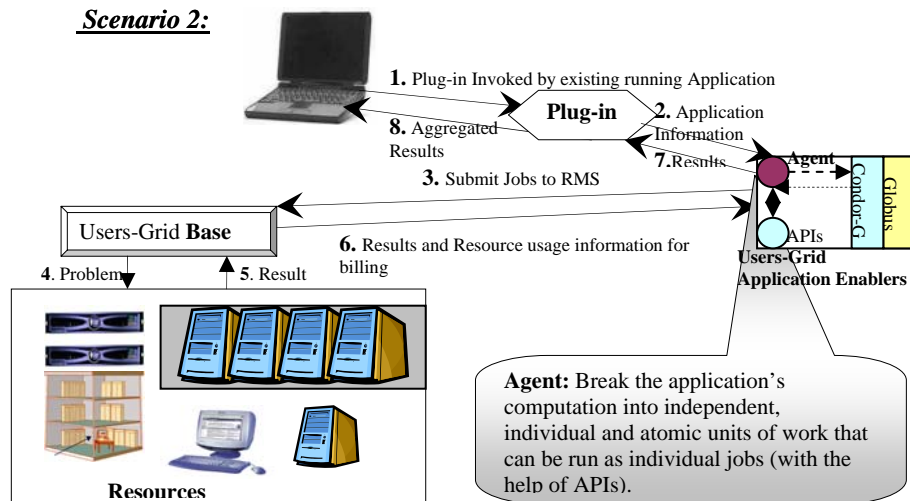


Figure 5: Running existing applications on Grid with Plug-in

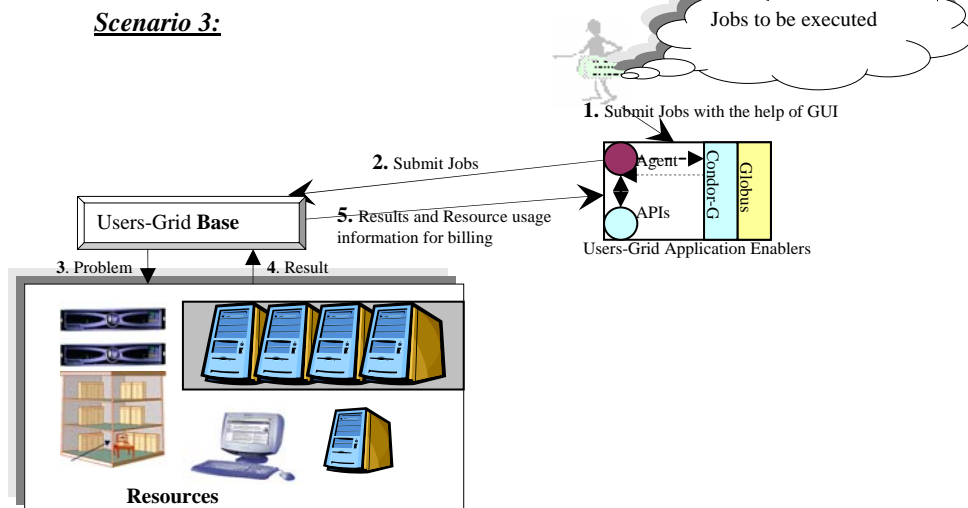


Figure 6: Simple Job submission to Users-Grid Base by GUI Facility

We want to develop a .Net Plug-in to make Excel Grid-enabled by enabling composition and execution of parametric sweep applications on remote grid nodes and using Users-Grid Base Layer, which contains Condor. This plug-in support will be provided on Windows platform only. With the help of parametric sweep model, multiple instances of a job can run in parallel on different grid nodes with different parameters [26, 27].

Potentially we had two choices to run make Excel Grid-enabled

1. To transfer the work to remote instances of Microsoft Excel
2. To transfer the work to other alternative faster implementations e.g. .Net, C++, Java etc.

We have chosen the second choice, but for the understanding purpose we describe both approaches here in following figures 7 and 8.

To transfer the work to remote instances of Microsoft Excel

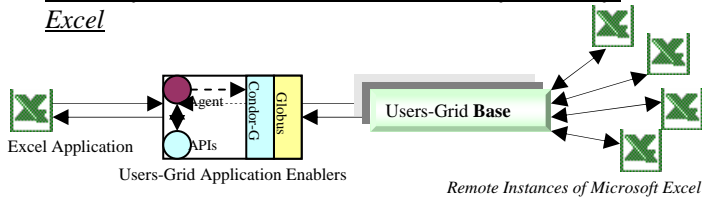


Figure 7

It is very simple approach indeed, but it needs Excel to be already installed on remote machines.

2. To transfer the work to other alternative faster implementations

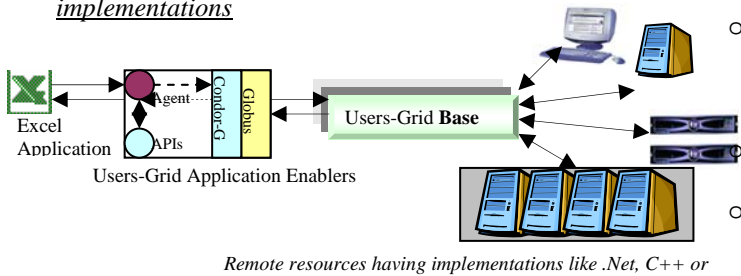


Figure 8

As Excel's interpreted execution is slow, if we transfer work to other faster implementations such as .NET, Java and C++ then we can get better result. In this way we will use existing software's functionality into Excel.

6 REFERENCES

- [1] The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. I. Foster, C. Kesselman, J. Nick, S. Tuecke, *Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.*
- [2] Foster, I. and Keselman, C.(1998) The Globus project: A status report. Presented at *IPPS/SPDP '98 Heterogeneous Computing Workshop, 1998.*
- [3] Grimshaw, A. and Wulf, W. (1997) The legion vision of a worldwide virtual computer. *Communications of the ACM, 40(1):39-45.*

5 CONCLUSION

Grid revolutionized the computing world by enabling collective use of globally distributed resources, but the technologies, softwares or toolkits available today to realize this idea, make it more difficult to use for normal user, who is not willing to be a computer specialist, just to use the Grid. Technologies need to be matured and we need to build supporting environment around these toolkits to get a better level of abstraction, in order to simplify its use. Grid benefits should not be limited to only science and academia or research laboratories, a desktop user should be given access to Grid resources seamlessly without putting user into complexities. It is to realize that we cannot just start application development from the scratch to get benefit from Grid. We have to enable existing business and science applications to get benefits from this new technology.

We proposed Users-Grid and described its complete architecture and working, it is single proposed solution which captures all of these requirements in one robust solution. In brief we present its unique features

- o Automatic shifting of extra load of computation from user's computer to the other nodes on Grid, which are underutilized and bill back the user according to the resource usage.
- o Enabling existing applications to run on Grid, with no change in the code.
- o Plug-in support with many existing applications to make them grid-enabled.
- o Maximum GUI support, and save user from command prompt as much as possible. Thus user needn't to learn a special language (RSL) just to submit a job onto Grid
- o Web based portal to submit, monitor, manage and see resource usage and accounting statistics.
- o Implements administrators and users policies for resource availability, security and accounting issues.

- [4] Fran Berman and Geoffrey Fox, “*Grid Computing Making the Global Infrastructure a Reality*”, John Wiley & Sons, pp. 340-341, May 2003.
- [5] I. Foster, I. and Kesselman, C. (1997) Globus: a metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 115–128.
- [6] Globus: <http://grid.phys.uvic.ca/docs/uberdoc/node6.html>
- [7] Globus-Only topology:
<http://grid.phys.uvic.ca/docs/uberdoc/node11.html> (node11.html to node14.html)
- [8] Fran Berman and Geoffrey Fox, “*Grid Computing Making the Global Infrastructure a Reality*”, John Wiley & Sons, pp. 252-253, May 2003.
- [9] Condor, <http://www.cs.wisc.edu/condor/>.
- [10] Frey, J., Tannenbaum, T., Foster, I., Livny, M. and Tuecke, S. (2001) Condor-G: a computation management agent for multi-institutional grids. *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC)*, San Francisco, CA, August, 7–9, 2001.
- [11] Sun Grid Engine: , <http://www.sun.com/software/Gridware/>.
- [12] SuSe Linux to Distribute Sun’s Grid Engine Software.
<http://siliconvalley.internet.com/news/article.php/1013251>
- [13] Open PBS (*it has link to download but yet un-downloadable*)
<http://www.openpbs.org/download.html>
- [14] The Vega Personal Grid: A Lightweight Grid Architecture , Wei Li, Zhiwei Xu, Bingchen Li, Yili Gong, Fourteenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2002)
- [15] Buyya, R., Abramson, D. and Giddy, J. (2000) Nimrod/G: An architecture for a resource management and scheduling system in a global computational Grid. *The 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000)*, Beijing, China, 2000.
- [16] Abramson, D., Giddy, J. and Kotler, L. (2000) High performance parametric modeling with nimrod/G: killer application for the global Grid? *International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society Press.
- [17] Buyya, R., Giddy, J. and Abramson, D. (2000) An evaluation of economy-based resource trading and scheduling on computational power Grids for parameter sweep applications, *The Second Workshop on Active Middleware Services (AMS 2000)*, In Conjunction with HPDC 2001. Pittsburgh, USA: Kluwer Academic Press.
- [18] Almond, J. and Snelling, D. (1999) UNICORE: Uniform access to supercomputing as an element of electronic commerce. *Future Generation Computer Systems*, **15**, 539–548.
- [19] Fran Berman and Geoffrey Fox, “*Grid Computing Making the Global Infrastructure a Reality*”, John Wiley & Sons, pp. 711-712, May 2003.
- [20] Fran Berman and Geoffrey Fox, “*Grid Computing Making the Global Infrastructure a Reality*”, John Wiley & Sons, pp. 626-627, May 2003.
- [21] Gold Accounting and Allocation Manager: <http://www.emsl.pnl.gov/docs/mscf/gold/>
- [22] Bart Jacob and Luis Ferreira, “*IBM-Enabling Applications for Grid Computing With Globus*”, IBM, pp. 45-53, June 2003.
- [23] Excel for Business Intelligence, September 9, 2003.
<http://www.microsoft.com/office/previous/xp/business/intelligence/excel.asp>
- [24] Agents.<http://www-2.cs.cmu.edu/~softagents/intro.htm>
- [25] Agents.<http://ils.unc.edu/gants/agentbib.html>
- [26] Cheuk, G., ActiveSheets: Grid Computing with Spreadsheets, The First Australian Grid Forum Workshop (OzGrid-1) Dec. 9-10, 2002, Melbourne, Victoria, Australia.
- [27] Existing applications for Grid:
<http://www-128.ibm.com/developerworks/grid/library/gr-xist/index.html#IDAHE2UB>