

# THE GRID TAKEN LITERALLY

Brian Vinter, Department of Computer Science, University of Copenhagen

**This paper motivates and describes the Minimum intrusion Grid, MiG[2], a Grid model that seeks to take the word 'grid' literally. By literally we mean that the analogy to the electrical power grid[1] is taken as far as we see possible. Components that differentiate MiG from Globus and similar models include a zero-size code base that is required to be installed on client or resource computers, mandatory payment and pricing of Grid services, end-to-end anonymity between consumers and producers and approximating the perception of a PC by means of Grid computing. In addition MiG provides a new, simpler approach to Virtual Organizations and a seamless integration of the Public Resource Computing model into Grid.**

## I. INTRODUCTION

In the mid 1990's when Grid first was introduced, the idea of getting computing resources from a socket in the wall helped create a giant hype around Grid-computing. Since then one could argue that, as money has been pored into Grid research, the ambition level of the same research has dropped proportionally; to the point where Grid today appears to be merely web-services in another wrapping! Grid research and Grid systems seem to be severely limited by initial choices made towards location, naming and security aspects in Grid. Most, or even all, of these were not obviously wrong, or were not at all wrong at the time they were made, but in the end we have ended up with a Grid model that is a long way from providing computing resources from a socket in the wall, particularly for commercial enterprises and most especially for private users.

## II. DECOMPOSING THE PC

If we are to return to the path towards computing resources from a socket in the wall, we ought to look at computing resources as they appear today and thus what we ought to mimic in Grid. The de-facto perception of a computer-resource today is the PC, only few people on a global scale use supercomputers, clusters or even mainframes today. Naturally the fraction of people who do use these kinds of resources is disproportional in the Grid research community and our perception of computing resource thus often becomes one of supercomputers.

If however we are to address the needs of corporate and private users we need to address the PC as the presentation of computing resources. In the following we try to decompose the PC and make considerations towards how each component may be provided through a Grid model.

### *A. Processor*

Representing the CPU in Grid is the least of our problems; in fact one may argue that processing power is the only thing we have successfully modelled with the wall socket paradigm from the original Grid approach. In the Minimum intrusion Grid we do this no differently from standard Grid models. One small addition in MiG is that we do support real-time access to CPUs and allow multiple jobs to share the same CPU for low-intensity jobs. Thus if a user simply need to run an application, this task is most often not suitable for batch-processing, but on the other hand this type of task need not necessarily be computationally heavy and thus require the entire CPU.

### *B. Storage*

It is our belief that the storage-element in the standard Grid model is counter-intuitive to the PC-type user. The ordinary user will expect to be able to address his own files relative to a home-directory and not have to specify; the protocol by which the data may be accessed, the name of the machine where the data is kept, and the absolute path to the data on that machine. In MiG users per definition have a home-directory, and files that are specified in a job-description have paths relative to that home-directory. Users may easily access these files, either through a web-based gui, or by accessing them through their local computer, be it by simple copy semantics or by directly integrating their Grid folders in their local file-system, both of which are currently supported

### *C. I/O*

The perhaps most counter-intuitive part of the standard Grid model compared to the PC model of computing, is the idea that jobs should fit a batch computing paradigm. Not only are users accustomed to working interactively with their applications, most end-user applications are not suited for batch-processing and require the interactive operation support. To address this perspective the MiG model does support a Grid hosted frame-buffer to which interactive applications run on Grid may render their output. Keyboard and mouse input is supported in a similar manner. The end result is that users run applications interactively, using the files that are in their own directories.

The fact that the user does not know where an application is running or that his files are not co-located with the host that executes the application, does not affect the user's model of the underlying computer. Though the perception of a virtual PC is not perfect, it is close enough that working on Grid is not alien to the ordinary user.

### III. THINKING OF THE GRID

Once the use of Grid has been put in context the next step is rethinking the Grid resource models. Again we find it convenient to stick to the well known understanding of grid, in the meaning of the electric power-grid. The first limitation one must accept, though, is that electrical power is a fairly one-dimensional concept whereas computational resources are much more diverse and thus a lot harder to define. Even though, we seek to follow the power-grid analogy as far as it will take us. In this analogy we try to combine resources from dedicated computing sites with ad-hoc resources as found in public resource computing systems such as the BOINC[3] model.

#### *A. Delivery*

The standard reasoning why Grid is better than hosting your own resources lends itself to the analogy; why do we have electric power-grids and not individual generators? One of the arguments against hosting individual electricity generators is the complexity and cost of running your own generator. Compared to maintaining a generator, simply having electricity delivered to your house from the electric power-grid is much easier. One may then ask if the common Grid solutions deliver the same ease of service delivery or if the analogy breaks? Having electricity delivered to your house is trivial and requires no maintenance while the client-side software in most available Grid solutions is by no means trivial. Most users are hidden from this fact by application portals that host the client-side software, but to access Grid directly from your local computer requires a lot of software on the local machine.

Delivering resources to Grid is even worse, the current Globus 4 distribution requires more than 150MB of compressed applications to be downloaded, installed and maintained to put a Linux based host on Grid as a resource[9].

Part of the ‘minimum intrusion’ in the name “Minimum intrusion Grid”, relates to just the topic of maintaining Grid software on clients and resources. The initial motivation was to require “as little as possible”, i.e. minimum, to be installed on either end of the Grid. However “minimum” software installed is a very hard metric to measure since testing for minima is impossible and we kept ending up in fruitless discussions whether a system was minimal or not. Thus we took the decision that “minimum” is none and by striving towards a zero-sized install base it is obvious that there is nothing more to remove and thus we have obtained minimum.

Achieving a zero-sized code-base at client and server end turned out to be fully possible, and today the MiG system may be operated from a user’s perspective using only a browser that is X.509 enabled. Using the browser the user may submit and manage Grid jobs, upload, edit and manage files in his Grid home-directory and control access to the resources he owns on Grid.

Managing resources on Grid is equally simple. The resource host simply creates a local user which is the identity that will execute Grid jobs. This user must be able to use SSH to access the resource and HTTPS to communicate to the outside world, and the private SSH-key of the user, as well as the public SSH-key of the host is registered with MiG at which time the resource can be used in Grid with mutual trust of identities between Grid and the resource. For resources that do not allow ingoing SSH or outgoing HTTPS, special solutions exist, however one or the other must be supported. Common to all the solutions is that there is no resource certificate involved, the owner registers the resource with her own certificate, but there is no Grid software or certificates installed on the resource side.

In reality any user may add a personal account to Grid as a resource without the local system-administrators knowledge; naturally we do not endorse such behaviour.

#### *B. Power plants*

The power-grid is primarily powered by large power-plants, fossil-fuel, nuclear or hydro-electric, with a constant, or at least dependable, production. Similarly the Grid is primarily driven by reliable, professionally run, compute-centres. These centres provide schedulable resources of very

high reliability. However the resources at these centres are not free, in fact hosting and running a PC in a compute-centre is usually more expensive than buying and running a privately owned PC. Here the standard Grid model fails as access to these resources is not governed by payments, as we use with electricity, but by a first-come-first-serve principle or at best by a simple allocation scheme. In MiG we insist on pricing the resources to induce a need based utilization of the expensive resources. Pricing of the resources happens as a continuous double auction, which has been proved to have honest pricing as the winning strategy[11]. Through this model we ensure that resources are utilized at all times while pressing needs can easily be served first since the user may simply raise the price he is willing to pay to get the job executed.

### *C. Wind turbines*

On the electric power-grid there are resources that may produce electricity at close to no cost once the initial investment has been covered, these resources include solar panels and most predominantly wind turbines. Similarly the computing Grid has an abundance of individual PCs that are virtually free to use but, like wind or solar power, the individual resource is low-powered and highly unreliable from an availability perspective. MiG provides two models for harvesting PC cycles, a ‘full’ model that makes the resource appear as a classic Grid resource[4], and a limited model that reduces joining Grid to entering a web-page[5].

The ‘full’ Grid resource model in MiG is based on a small screensaver which on activation starts a virtual machine, typically the free VMware player[6]. The virtual machine in turn boots a small Linux image which automatically starts servicing as a Grid resource. Once the screen-saver is deactivated, the virtual machine is forcefully killed to ensure that the user does not feel any side-effects from the contribution to Grid. As these resources are so volatile, their successful usage heavily depend on MiGs ability to successfully guess the length of time a screen-saver is likely to be active and to schedule tasks that will finish in that time. Users with sufficient memory in their computers may chose to run the virtual machine at a low priority at all times to provide even more processing power to MiG.

The One-Click model offers what we believe to be the simplest means of providing resources to the Grid. Users need only enter a special URL and their web-browser will load a Java-applet that serves as a Grid resource script. This script in turn contacts MiG and offers to execute any compatible jobs. For a job to be One-Click compatible it must inherit from a MiG provided class that takes care of file access and also provides means for semi-transparent check-pointing.

A third windmill model has just been released. This model allows owners of Playstation-3 game-consoles to download an ISO image for a live-CD and boot their Playstation-3 in a Grid resource mode. Contrary to the Folding@Home[7] model that comes preinstalled on the Playstation-3, the MiG model is a ‘true’ Grid approach – thus there are no binaries preinstalled since these come as part of the Grid job. As the Playstation-3 is immensely powerful we hope that this model will prove successful, not only from a technical point of view but also from an adaptation perspective.

#### *D. Accounting*

In the electric power-grid resources are not free and Grid resources are actually quite expensive even though they are often available for free through Grid. The need for accounting and payment is fairly accepted in the Grid community by now, and much work has been done in this, the best of which is done under the Gridbus project[10]. Rather than adding accounting and pricing to the Grid model the MiG approach is to base all scheduling on economics. As a resource becomes available it is sold to the job which is willing to pay the most for the available resource. If there are more resources than jobs, the situation is reversed, a new job is sold to the resource with the lowest asking price. The actual price is never the asking or bidding price, but based on a game-theoretic approach that is shown to have honest-pricing as a dominant strategy. The details of this system are described in [11].

#### IV. EXTRAS

There are obviously components in Grid computing that do not fit the power-grid analogy. While we have to abandon the power-grid analogy for these components we still seek to meet our internal

design criteria for ‘minimum intrusion’, which dictates that one should not have to install Grid specific software to join Grid as user or a resource provider.

#### *A. Virtual Organizations*

Virtual organizations in MiG have been implemented as VGrids[8], a concept that turns out to be both more flexible and powerful than the VOMS approach. Through VGrids users may in a simple manner share resources and data and the creation and management of virtual organizations is left to the users without any central management. In addition a VGrid automatically provides the user group with public- and private web-pages, a Wiki and a CVS repository.

Since resources may also be associated with a VGrid, and since a resource is simply a user account on the host, managing fair-share scheduled resources on MiG is trivial. A fair-share scheduled resource on Grid may simply add a user-account to each quota holder that wish to use the resource through Grid. The local fair-share management system will then handle the allocations and transparently register usage from both the local queue and Grid since the system simply see the Grid as yet-another-user.

#### *B. Non-computing resources*

The strict enforcement of payment as a means of scheduling jobs in Grid has opened for the introduction of non-computing resources as Grid resources. A resource thus may be any IP enabled component, such as a computer controlled telescope, but more interestingly it may also be human resources. Through MiG one may request a SIP-telephony based consultancy on any topic that helpdesks exists for, and the scheduling mechanism will automatically pass the phone-call to the cheapest provider that offers the features the user needs, i.e. a Windows OS helpdesk.

#### *C. Video Conferencing*

As an extension of the VGrid implementation of virtual organizations, video-conferencing is currently being tested as a service to virtual organizations. Access to the virtual organization is controlled through membership of the organisation and this does not require additional username and password setups. At the same time the conference is automatically recorded and stored in the area that belongs to the virtual organization.

#### D. Meta computing

A common misconception of Grid computing is that a Grid acts as a virtual supercomputer and thus allows a user to run a single application on hundreds of CPUs that are distributed across the Grid. Though this was never the intention of Grid computing the MiG project does provide support for meta-computing for those programmers that wish to use this approach. The features that are provided to help the meta-computing approach along are co-scheduling of tasks and Grid-based shared data-structures and databases that are persistent across the termination of individual jobs.

#### V. CONCLUSIONS

Minimum intrusion Grid seeks to meet the general perception of both Grid and computers to fulfil the idea of compute resources from a socket in the wall. MiG has design criteria which dictate that no Grid specific software must be required to be installed, that users and resources must remain mutually anonymous and that users must be able to run real interactive applications. MiG is fully functional, highly reliable robust, several research projects are operated through MiG today and we hope that many more will see the advantages of the simple approach to Grid that MiG provides.

#### REFERENCES

- [1] I. Foster, C. Kesselman. "Computational Grids.", *Chapter 2 of The Grid: Blueprint for a New Computing Infrastructure*, Morgan-Kaufman, 1999.
- [2] Brian Vinter, "The Architecture of the Minimum intrusion Grid: MiG", *in proc of Communicating Process Architectures 2005*.
- [3] D.P. Anderson , "BOINC: a system for public-resource computing and storage", *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04)*.
- [4] Rasmus Andersen and Brian Vinter, "Harvesting Idle Windows CPU Cycles for Grid Computing", *proceedings of GCA-2006*.
- [5] Martin Rehr and Brian Vinter, "The One-Click Grid-resource model", *submitted for publication*.
- [6] <http://www.vmware.com/products/player/>
- [7] <http://folding.stanford.edu/FAQ-PS3.html>
- [8] Henrik H Karlsen and Brian Vinter, "VGrids - Virtual Grids in Minimum intrusion Grid", *in proc. of ETNGRID 2006*
- [9] <http://globus.org/toolkit/downloads/4.0/>
- [10] Alexander Barmouta and Rajkumar Buyya, "GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration", *Workshop on Internet Computing and E-Commerce, Proceedings of the 17th Annual International Parallel and Distributed Processing Symposium (IPDPS 2003), IEEE Computer Society Press, USA, April 22-26, 2003, Nice, France*.
- [11] René Brask Sørensen, "Analysing Resource Allocation in Minimum Intrusion Grid (MiG) using Mechanism Design", *MS. Sc dissertation, department of Computer Science, University of Copenhagen*.