

# XtremWeb & Condor : sharing resources between Internet connected Condor pools.

O. Lodygensky, G. Fedak, F. Cappello, V. Neri, M. Livny, D. Thain

LAL, Laboratoire de l'Accelérateur Lineaire, France

LRI, Laboratoire de Recherche en Informatique, France

Department of Computer Science, University of Wisconsin-Madison, Wisconsin, USA

## Abstract

*Grid computing presents two major challenges for deploying large scale applications across wide area networks gathering volunteers PC and clusters/parallel computers as computational resources: security and fault tolerance.*

*This paper presents a lightweight Grid solution for the deployment of multi-parameters applications on a set of clusters protected by firewalls. The system uses a hierarchical design based on Condor for managing each cluster locally and XtremWeb for enabling resource sharing among the clusters.*

*We discuss the security and fault tolerance mechanisms used for this design and demonstrate the usefulness of the approach measuring the performances of a multi-parameters bio-chemistry application deployed on two sites: University of Wisconsin/Madison and Paris South University.*

*This experiment shows that we can efficiently and safely harness the computational power of about 200 PC distributed on two geographic sites.*

## 1 Introduction

Multi-parameters applications represent very interesting candidates for Grid deployments. Many high performance computing users use such kind of application and their executions typically require the computation of many tasks controlled by a central manager. As a matter of fact, many of these users are not willing to spread their applications among volunteer participants connected to Internet like for SETI@home or Folding@home. They simply look for a Grid solution helping them to harness efficiently, safely and at a low installation cost, a flexible set of loosely coupled clusters (secretary and engineer workstation connected to switched Ethernet) and closely coupled clusters (Beowulf

cluster). The Grid solution must fulfill at least two enabling features: security and fault tolerance.

These applications are also interesting from the computer scientist point of view because they can fit the two known approaches of Large Scale Distributed Computing : peer to peer computing (P2P) and Grid computing [4].

“Grid” main goal is to achieve flexible and secure large scale computing with high performances between so called *virtual organizations*, entities that accept a resource exchange policy, based on high control about who shares what, what is shared and what are the sharing conditions. The main toolkit, Globus [14], is already used by several projects. One of the main contributions of Globus is its strong security mechanisms (GSI) which is the basis of many current Grid deployments. However, Globus which has been designed to allow many users to share resources owned by institutions may not be appropriate in the specific cases of multi-parameters applications ran from a centralized dispatcher. Another concern about Globus is its lack of mechanism for transparent fault tolerance, which leads to use a fault tolerance environment for executing multi-parametric applications or a fault tolerant implementation of multi-parametric application. The Heartbeat Monitor [13] is a proposed a way to detect faults, but it is not transparent, since applications have to register.

“P2P” has quite the same goals (to achieve flexible and secure large scale computing), but in a more decentralized way. Two of the main characteristics of P2P system are the resource volatility and the lack of security mechanisms for the participant resource, the application and its results. For example there is almost no identification procedure for users of P2P systems. However, current deployments such as SETI@home for P2P Computing and Napster, Kazaa for P2P file sharing, have demonstrated exceptional performance (several tens of Teraflops for SETI@home and about 1 Terabit/s of service bandwidth for Kazaa) and fault toler-

ance (the time between two connections or disconnections is lower than a minute).

The specific case of multi-parametric applications ran for a single trusted institution/community using a central dispatcher and a set of clusters protected by different administration domains allows to consider a P2P approach for solving the security and fault tolerance issues.

We present such a solution in this paper gathering clusters managed by Condor and protected by firewall by the XtremWeb P2P platform. The two first sections detail Condor and XtremWeb technologies. The next section presents how XtremWeb is used to safely harness firewall protected Condor pools and how it provides fault tolerance. The last section demonstrates the usefulness of the approach describing a real life multi-parameters application ran on two clusters: one at University of Wisconsin/Madison in USA and the other at University of Paris South in France.

## 2 Condor

Condor, a specialized batch system for managing compute-intensive jobs, is a project developed at University of Wisconsin-Madison [11]. Like any batch system, it manages submitted jobs, queuing and scheduling them accordingly to their priority.

Condor not only manages jobs; it more generally manages different kind of resources included in so-called *Condor pools*. A pool is an administrated domain of hosts, not specifically dedicated to Condor, which Condor uses as any user of the domain. Condor does not administrate these hosts by itself; administration still relies on domain administrator, especially (but not only) for security purposes which are fully dependent of the domain administration. For instance, jobs are launched with user/group rights as defined in the domain.

As Condor only manages resources, hosts of the pool are resources. Resources management follows rules defined within *ClassAds* to enable matchmaking on resource sharing. Condor uses *ClassAds* to resolve association between requests and proposals of resources. Users could request a host running an expected operating system, with an expected amount of memory to run jobs. The system would then look if it has such a resource available and resolve the request. On the other hand pool hosts may not be dedicated to Condor, as we previously said; users could propose their personal desktop, or laptop, publishing its *ClassAds* specifying utilization conditions such as available scheduling time (e.g. *from 9PM to 6 AM*), or owner CPU usage condition (e.g. *CPU available for Condor job if CPU usage under 10%*; *Condor job don't use more than 50% of CPU*).

Condor manages tasks following *master-worker* paradigm. This is a common parallel programming paradigm where one host (the *master*) control the others

(the *workers*). The master centralizes jobs, and sends all needed objects (binary, parameters files...) for each job to a worker which runs the provided application with specified environment (arguments, inputs...). As soon as a worker has finished, it sends output files back to the master. As the master keeps tracks of every resources, it is able to detect any worker failure which may simply occurs on the case host resource is not available any more accordingly to its *ClassAds*; on such cases, the master reschedules that lost job to another worker.

Rescheduling lost tasks is a waste of CPU time since new elected workers have to restart computation from the beginning. To avoid such loss of resources, Condor provides a *check point* service allowing the worker to send checkpoint images to a storage server.

We mentioned earlier in this section that Condor manages pools deployed in a single administrative domain. Resource sharing is not limited to one pool only and different ones (different domains) have a chance to communicate and exchange resources and tasks. To do so, Condor implements the "*flocking*". This allows sharing between pools, with due respect to policy of each. Unfortunately, the current Condor version (6.4.3) does not provide any mechanism to deal with firewall for inter administration domains resource sharing. The Condor team currently investigates this problem, considering the use of gateways [12].

Another Condor approach to break domain limitations is to use Globus[1] to connect pools to grid machines to submit jobs there. This Condor service, called *Condor-G*, is not included in the standard Condor distribution and has to be installed explicitly.

All this, of course, is possible if, and only if, one has access to Globus resources; to do so it is necessary to get a Globus account.

We will see on following sections how XtremWeb can resolve this inter domain limitations and complexities.

## 3 XtremWeb

XtremWeb is a P2P project developed at University of Paris-Sud, France[6]. It was originally designed to study execution models in the general framework of Global Computing and is now a fully production platform too; it has been released for Linux, Windows and MacOS-X.

As other GC projects like SETI@home or Folding@Home, XtremWeb is intended to distribute applications over a set of hosts using a cycle stealing scheme and particularly focuses on multi-parameters applications which have to be computed several times with different inputs and/or parameters, each computation being fully independent from each others. Some other projects propose

equivalent features, such as Nimrod [5]. But Nimrod uses a static set of resources only and security relies on standard Unix level security. We see in following paragraphs that XtremWeb uses dynamic resources, according to their availability and implements its own policy security.

XtremWeb manages tasks following the *coordinator-worker* paradigm. One host (the *coordinator*) manages a bag of tasks and coordinates their scheduling among a set of hosts (the *workers*) which are volunteers provided by institutional or private users and, as such, are not under the control of the coordinator and are very volatile in essence. Following these concepts, each action, all connections, are initiated by workers only. This behavior is commonly known as *pull* model and clearly implies independence of all components.

Tasks are scheduled to workers on their specific demand only since they may appear (connect to coordinator) and disappear (disconnect from coordinator) with no predictable pattern. Every worker connection is registered by the coordinator providing configuration (XtremWeb worker version, operating system, CPU type, memory size...). Software version helps the coordinator to check whether the worker should be updated or not; other informations are used to match scheduling accordingly to task needs.

XtremWeb uses connectionless protocols; workers are considered active as long as they contact the coordinator periodically after registration.

A worker requests task to compute accordingly to its own local policy dedicated to perturbation prevention, which is customizable (*available scheduling time, CPU usage conditions...*); it downloads task software and all expected objects (input file, arguments...) and starts computing the provided tasks. Computation goes on locally until it ends or dies for any reason, including due to host utilization policy rules. When a task is completed, the worker sends results back to coordinator and may ask for another one.

Scheduling is in *FIFO (first in, first out)* mode, which is a very trivial one. Since XtremWeb can schedule native and *Java* applications, the only match done is about CPU type, OS version and whether Java is enabled in workers. Java applications are distributed as *class* or *jar* files, whereas native ones are binary.

#### 4 Light weight Grid of Condor pools using XtremWeb

We wish to gain profit from both technologies, Condor and XtremWeb, to share resources between different Condor pools as shown in figure 1. The big picture consists in running XtremWeb workers as Condor tasks fetching their jobs from the central coordinator possibly belonging to different administration domains. The first issue (4.1) is to transform XtremWeb worker code as a stand alone applica-

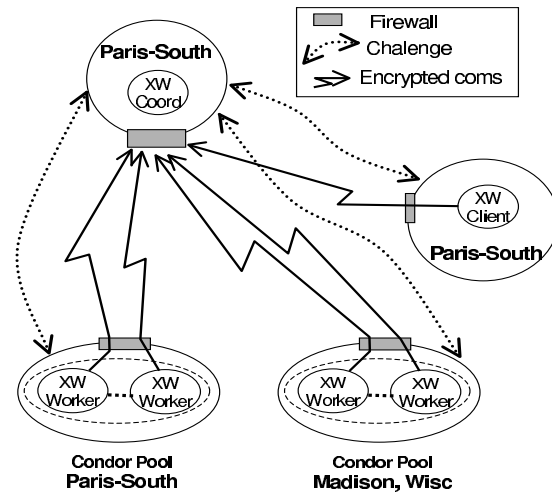


Figure 1. Connecting Condor pools using XtremWeb

tion runnable in Condor pools. The second issue (4.2) concerns the coordination of Condor tasks (XtremWeb workers) by a central coordinator. Third issue (4.3) consists in providing security mechanism to ensure the Condor nodes, results and applications protection. The final issue (4.4) is about fault tolerance: we must ensure that failed tasks don't affect the system and are restarted transparently from the user.

#### 4.1 XtremWeb workers as Condor service

Deploying XtremWeb workers as Condor service is the first point to address for our goal so that tasks could be scheduled on any Condor pool and coordinated by XtremWeb.

To deploy XtremWeb workers on Condor pools, we have to be able to dispatch them as Condor tasks. We must first make sure XtremWeb workers will not only be runnable, but will not perturb Condor scheduling.

XtremWeb workers use some "standard" Java libraries as those needed to implement *SSL* connections, or logging features. As no assumption should be made on Condor pool to deploy tasks, it is then necessary to implement XtremWeb worker as *stand-alone* application regarding its system needs. Unfortunately, current XtremWeb worker implementation can't bypass two of its needs: disk access first, as all downloaded objects are written down to ensure tasks will be fully computed even if worker host reboots and, second, Internet outgoing connections must be allowed to be able to connect the coordinator (see next subsection).

An access to administrative domains is necessary to deploy XtremWeb stand-alone worker as a Condor task. We have installed a 40 hosts pool at LRI which we manage ourselves and have an access to University of Wisconsin to schedule our XtremWeb workers to the Condor pool there, with up to nine hundred Condor workers installed.

These two pools are not dedicated to our experimentation and are strongly loaded by users. Our workers are then scheduled accordingly to pools policy and dedicated resource match makings, which are mainly proposed by volunteer users.

There is no specific XtremWeb policy defined regarding host utilization since Condor, which fully manages the scheduling, schedules workers, using its ClassAds. On the other hand, Condor master has no result because XtremWeb workers, by themselves, make no output, neither any result. Generated results are not visible by Condor system since XtremWeb already manages them; these results are sent back to the coordinator and removed from local disk. In fact, Condor only sees its own tasks (here, our XtremWeb worker) and has no way to check what is done if it does not rely on result files.

## 4.2 XtremWeb coordinator for Condor pools interconnection

Solving domain administrations and firewalls problems by connecting different Condor pools can be done through XtremWeb coordinator. Nimrod, already cited in this paper, proposes Nimrod/G [9], an implementation of Nimrod with Globus to distribute tasks among grid resources. We demonstrate in this section that XtremWeb solution is very easy to implement and the least constraining one we know so far since it does not need Globus a very good but heavy solution to connect widely distributed hosts.

Condor team proposes their solution too, Condor-G, based on Globus standard mechanisms: *GSI*, for security, allows usage of grid resources with a single authentication; *GRAM*, for tasks executions, helps to submit and manage queues of jobs to grid hosts; and *GASS*, for file transfers between grid resources. Condor-G uses that last to transfer executables and all other needed files to remotely compute a task.

XtremWeb protocols, as defined, resolve firewall problems by using single side communications. Firewalls are usually configured asymmetrically allowing outgoing connections and blocking incoming ones. Workers and clients behind a firewall or even a gateway implementing *NAT* can then contact the coordinator and receive answers through the same opened canal. The coordinator, on his side, can receive connections since it uses the standard *Web* port (80)

and firewalls are usually configured to let incoming connections to this port. If the firewall in front of the coordinator stops these types of connections, this will be the only one to be reconfigured so that the full system works.

Using XtremWeb coordinator doesn't imply neither any new Condor service installation nor firewall configuration as mentioned earlier; as soon as a coordinator is implemented, a global computing structure is ready and task dispatching occur as connections arise. Coordinator does not notice whether workers could run on a Condor pool, or inside a cluster or even as individual contribution.

For now, there is no real communication between XtremWeb and Condor; for instance, ClassAds as defined in Condor are not used by XtremWeb to grant match makings; neither are Condor queues: XtremWeb can't exchange any tasks with Condor. These are subject for future work.

## 4.3 Security

Our goal is to safely connect different administration domains in a single XtremWeb network. In this network, we assume that the users connect to the dispatcher administration domain for submitting tasks. XtremWeb has the responsibility to ensure user authentication, hosts (workers) integrity, application and results protection and user execution logging.

**User authentication and execution logging.** The coordinator site manages a list of authorized users as ACLs. It is the responsibility of the system administrator to register new users (and revoke non desired ones) on the coordinator. After registration, the coordinator provides a key to be used by the user for each subsequent connection. For each connection, a challenge is ran in order to ensure that the user is registered on the coordinator. All communications between the user XW client and the coordinator are encrypted using SSL. Then the coordinator works as a proxy for the user: all tasks are submitted to the workers through the coordinator credential. All executions on the workers are logged in the security perspective: all tasks contain a descriptor with the actual user credential so that workers and coordinator can take appropriate corrective action (user revocation), in case of security problem.

The design does not currently rely on certificates and presents a certain degree of risk for "Man is the Middle" (*MIM*) attacks but risks are very limited since 1) attacks should origin from within the Wisconsin or the LRI cluster only (due to TCP protocols), and 2) workers and clients software include coordinator public key, then if one is able to securely ensure worker and client binaries installation

to dedicated pools, the full system is not subject to *MIM* attacks since key exchanges will not be necessary any more.

A certificate system, like GSI in Globus, is under integration in XtremWeb. Subsequent experiments and future XtremWeb installations will implement one, based on Open-SSL, allowing extension of clients and workers authentication by the coordinator.

**Applications, parameters and results protection.** Condor pools belonging to different administration domains fetch applications and tasks, and store results on the central coordinator. The only security issue concerning applications, parameters and results transfers is then about the connections between the Condor pools and the coordinator. To ensure connection security between domains, 1) every connection from any client and worker to the coordinator is encrypted through *SSL* tunnels; 2) workers can only connect to the coordinator for which they have the public key. These two mechanisms prevent malicious participants to be able to intercept and read any connection, to connect to the coordinator and Condor pool workers to connect to a wrong XtremWeb coordinator.

**Node integrity.** If, for any reason, a malicious user succeeds on accessing the system and launching an aggressive application, XtremWeb workers still protect their host by implementing *sand-boxing*[2, 3, 7] for binary applications. This is a secure way to execute applications, providing rights to do some actions and denying some others. One should note that Java applications are always executed inside a virtual machine which includes security[10]; XtremWeb uses this functionality in two levels, one for the worker itself and a more restrictive one for the downloaded Java byte code. On the contrary, binary (or *native*) applications have access to the full hosting system by nature; workers are configured to run any task of that type inside a sand-box which is fully customizable, from memory usage to file system operations.

Java, as sand-boxes, have a performance cost[8]; one can then disable this functionality on highly secured systems, such as clusters under a fully closed firewall.

#### 4.4 Fault tolerance mechanisms

Several fault tolerance mechanisms are used in XtremWeb to handle clients, workers and coordinator failures. The main purpose of these mechanisms is to enable the system to restart properly after any failure (worker and coordinator). It is not currently intended to provide minimum service interruption using techniques like redundancy, but is planned as future work.

The client submits tasks to the coordinator as transactions. Before submitting any task, the client contacts the coordinator to fetch any previous submitted tasks. This ensures that when the client restarts from a fault, it does not resubmit previously submitted tasks. Results are managed according to the user needs. They can be discarded immediately after fetch or kept by the coordinator until the end of the session. So if a failed client restarts, it is the responsibility of the client programmer to fetch relevant results.

Worker periodically informs the coordinator sending an *alive* signal so that the coordinator doesn't reschedule its task to another worker. If this signal is not received after a time out, the coordinator considers the worker as lost and reschedules the same task on another available worker, if any. A worker can unpromptly be told to stop its current task if it has been disconnected for too long (i.e. if it has not signaled the coordinator in time) to avoid redundant task and, more, result overwriting. After any interruption (shut down or policy rule match), a worker restarts computing its task from the beginning (no checkpointing); this is useful if the task has not yet been rescheduled by the coordinator.

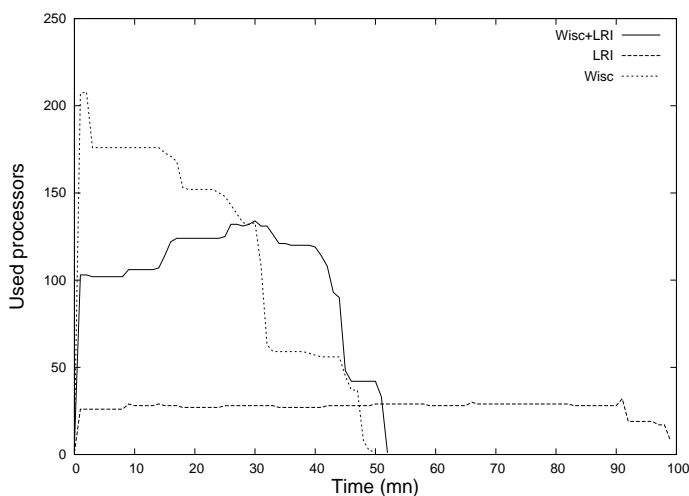
The coordinator stores on reliable media (disk) informations concerning tasks and workers. On start, the coordinator reads the information stored on reliable media to setup its proper state; it then retrieves tasks that have already been scheduled. The client submits tasks and the worker fetches tasks using transactions. This ensures a consistent state when the coordinator restarts from fault while the client and the worker have not failed.

All this is fully transparent to the distributed application and the end user is only aware about its results, computed or not (if no matching worker available). The only concern is the state of the coordinator; on failure the system manager has to re launch it.

## 5 Applications and analyses

To demonstrate the full system, we ran a bio molecular application for the IBBMC (Molecular and Cellular Biochemistry and Biophysics Institute) laboratory at Paris South University (France) which research interests include understanding protein dynamic structure parameters that affect stability and activity of proteins.

The general context is the understanding of stability and expression parameters of proteins activity from the analysis of their dynamic properties and their structure. The main goal is to evaluate the stability and perturbing factors of the protein folding by mutations. Molecular modeling is used to explore the conformational possibilities



**Figure 2. Processors utilization for different runs.**

of macromolecules at a time scale lower than 1 nanosec. The application consists in a multi-parameters computation requiring a large set of independent tasks. This is a 4 steps process. The first step generates  $n$  starting conformations along coordinate of interest. The second step performs  $m$  constrained molecular dynamics simulations for each starting conformation ( $n * m$  workers). The third step gathers statistics and the final step computes free energy profile.

The experiment consists in 320 tasks. The execution time of the run on a single machine (AMD 1.8Ghz) takes 43 hours and 44 min. This value is kept as reference in the following. We then deploy the same amount of tasks on three global computing configurations, all using Condor as deployment system on volunteer hosts and XtremWeb as dispatching system to enable inter domain hosts utilization.

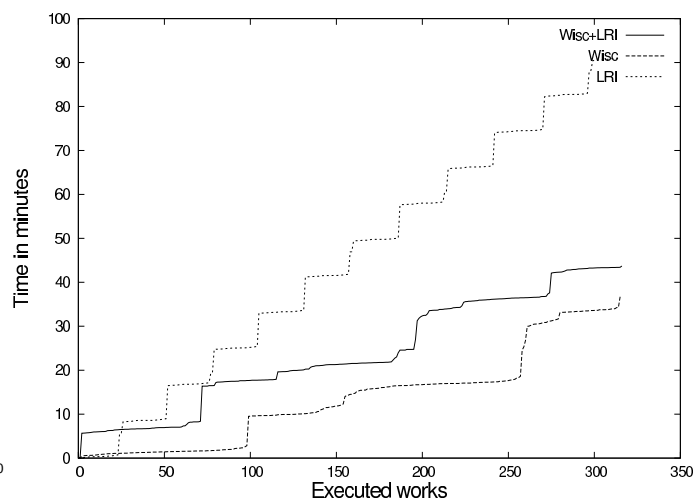
Table 1 summarizes hosts types for each experiments.

Exp.	Wisc		LRI	
	600Mhz	900Mhz	1800Mhz	Total
LRI			30	30
Wisc	61	104		165
W+LRI	50	73	9	132

**Table 1. CPU provided by different domains**

Figure 2 presents the number of CPUs used for each minute of the execution and for all experiments.

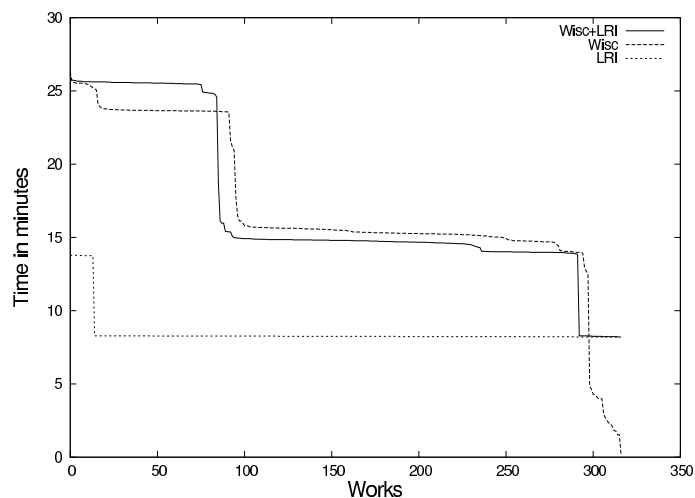
For each experience the left slop is the "warmup" phase where tasks are submitted but not yet distributed among



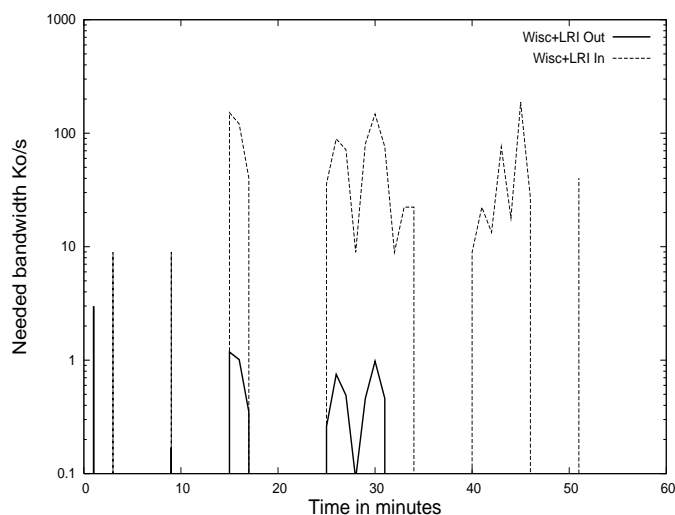
**Figure 3. Results relative arrival time for various platforms.**

all CPUs. The right slop is the "cool down" phase where are not enough jobs to use all CPUs. Since the scheduling algorithm is naive (there is no task redistribution to faster CPUs), the execution finishes waiting for the slowest CPUs. Figure 3 shows needed delays to get the results, for these different configurations, delays being calculated from the first arrived result.

The speed up between the different configurations is clearly visible here. Figure 4 presents an histogram of the task duration (all task durations are plotted sorted in a decreasing order) for the all the experiments.



**Figure 4. Execution time per work.**



**Figure 5. Used network bandwidth.**

In this graph we can easily distinguish the number of CPU types (speed) use in each experiences, given by the number of "plateau". *LRI* uses only one kinds of CPUs (AMD 1.8Ghz) while *Wisc+LRI* uses 3 kinds of CPUs. Figure 5 shows needed network bandwidth for the "full" configuration, where *Wisc* and *LRI* pools were used.

Finally, figure 6 shows tasks distribution among hosts for all experiments. They are detailed below.

First configuration uses 30 hosts available at *LRI*; all AMD 1.8Ghz powered, ten are bi-processors, the other twenty being mono-processors. This platform is used in production and our experiment shares CPU cycles with others users. Figure 3 shows that all tasks are done in 1 hour, 30 min and 54 seconds (speedup is about 29, due to other users hosts utilization) with a very regular execution time due to a single host configuration. The curve shows plateau, because all hosts get their works and put results simultaneously as they all need the same time to compute, as we can see in figure 4; whereas figure 6 shows a quite regular distribution among hosts: the load balancing is satisfying.

Second configuration has 165 hosts available at University of Wisconsin; 61 are PIII 600Mhz powered, the other 104 are PIII 900Mhz, all bi-processors. Figure 3 shows the execution is completed in 36 min and 41 seconds (a speedup about 88); some plateau are still present here, but less clearly visible since this configuration includes two host types which react differently as shown in figure 4. That last configuration is 4.5 times more powerful and the gain is multiplied by three only, due to a lower availability of nodes. In this case, tasks are regularly distributed as shown in figure 6.

The third configuration used both domains. Only nine hosts were available at *LRI* (5 mono-processors and 4 bi-processors) and 123 at University of Wisconsin (still all bi-processors, 50 at 500Mhz, 73 at 900Mhz); the throughput gain has decreased of 20% compared to our second configuration, exactly as available power (figures 2 and 3). Figures 3 and 4 show that the least regular curves are for this experiment, because of three different host types.

Figure 5, only available for this experiment, shows the needed network bandwidth. The bold line, labeled *Wisc+LRI Out* shows the outgoing traffic from the coordinator to workers (binary code and tasks parameters); the first vertical line correspond to binary download and paramters for initial tasks. The thin line, labeled *Wisc+LRI In* shows the in-coming traffic from workers to the coordinator (tasks results); the right most part of the figure shows only thin lines, because there is no more task (and then no more parameter to download) after minute 32 of the experiment. A significant point of this figure is the fact that lines are parallels between minutes 10 and 32; this is due to the fact that workers download new parameters immediatly after results upload, as long as there are tasks to compute.

Figure 6 shows that the load balance is preserved (i.e. tasks distribution among hosts in different domains corresponds to the different hosts type).

## 6 Conclusions

We have presented a lightweight approach for harnessing the computational power of clusters belonging to different administration domains for the purpose of running multi-parameters applications from a centralized coordinator.

We believe that this Grid deployment corresponds to the expectations of many users who don't need to have interactive and direct access to several firewall protected clusters.

We have presented a hierarchical architecture based on Condor for managing cluster nodes and XtremWeb for coordinating the task execution among the connected clusters.

This architecture fulfills the requirements of Grid deployments ensuring strong security (client/worker/dispatcher authentication + communication encryption) and fault tolerance using resilient components fetching their context before restarting.

We have demonstrated the usefulness of this approach using a biochemistry application and running hundreds of tasks on up to 200 nodes spread among two different sites in USA and France.

## References

- [1] The physiology of the grid: An open grid services architecture for distributed systems integration. In *Open*

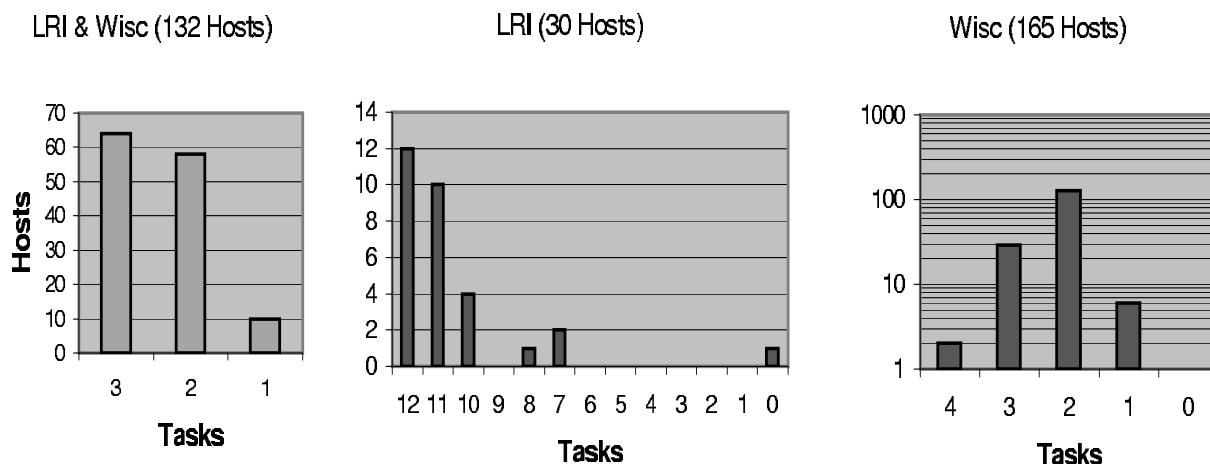


Figure 6. Tasks distribution among hosts when running the biochemistry application.

*Grid Service Infrastructure WG, Global Grid Forum, 2002.*

- [2] P. Kmiec A. Alexandrov and K. Schauer. Consh: a confined execution environment for internet computations. In *Proceedings of the Usenix annual technical conference*, <http://www.usenix.org/events/usenix99/>, 1999.
- [3] A. Acharya and M. Raju. Mapbox: using parameterized behavior classes to confine applications. In *Technical report TRCS99-25, University of California, Santa Barbara*, 1999.
- [4] Ian Foster and Carl Kesselman. The grid: Blueprint for a new computing infrastructure – isbn=1-55860-475-8. Morgan Kaufman Publisher, 1998.
- [5] D. Abramson J. Giddy R. Susic J. Giddy. Nimrod: A tool for performing parametrised simulations using distributed workstations. In *The 4th IEEE Symposium on High Performance Distributed Computing, IEEE Computer society*, 1995.
- [6] Vincent Neri Gilles Fedak, Cecile Germain and Franck Cappello. Xtremweb : a generic global computing platform – ccgrid’2001 special session global computing on personal devices. IEEE press, 2000.
- [7] R. Thomas I. Goldberg; D. Wagner and E. Brewer. A secure environment for untrusted help application – confining the wily hacker. In *Proceedings of the 6th Usenix Security Symposium*, 1996.
- [8] L. Pottage J.M. Bull, L.A. Smith and R. Freeman. Benchmarking java against c and fortran for scientific applications. In *ISCOPE Conference, LNCS volumes 1343, Springer*, pages 97–105, 2001.
- [9] D. Abramson J. Giddy L. Kotler. High performance parametric modeling with nimrod/g: killer application for the global grid? In *International Parallel and Distributed Processing Symposium (IPDPS)*, pages 520–528, 2000.
- [10] H. Prafullchandra L. Gong, M. Muller and R. Schemers. Going beyond the sandbox: an overview of the new security architecture in the java development kit 1.2. In *Usenix Symposium on Internet Technologies and Systems*, 1997.
- [11] Miron Livny Michael Litzkow and Matt Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems, IEEE Computer Society Press*, pages 104–111, Madison, Wisconsin, 1988.
- [12] Colby O’Donnell. Condor and firewalls. In <http://www.cs.wisc.edu/condor/>, University of Wisconsin - Madison, 2002.
- [13] P. Stelling, I. Foster, C. Kesselman, C. Lee, and Gregor von Laszewski. A Fault Detection Service for Wide Area Distributed Computations. In *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing*, pages 268–278, Chicago, IL, 28-31 July 1998.
- [14] I. Foster J. Geisler W. Nickless W. Smith S. Tuecke. Globus: A metacomputing infrastructure toolkit. In *Proc. 5th IEEE Symposium on High Performance Distributed Computing*, 1997.