XtremWeb : A Generic Global Computing System

Gilles Fedak, Cécile Germain, Vincent Néri and Franck Cappello
Laboratoire de Recherche en Informatique.
Université Paris Sud
http://www.XtremWeb.net

Abstract

Global Computing achieves high throughput computing by harvesting a very large number of unused computing resources connected to the Internet. This parallel computing model targets a parallel architecture defined by a very high number of nodes, poor communication performance and continuously varying resources. The unprecedented scale of the Global Computing architecture paradigm requires to revisit many basic issues related to parallel architecture : programming models, performance models, and class of applications or algorithms suitable to this architecture. XtremWeb is an experimental Global Computing platform dedicated to provide a tool for such studies.

This paper presents the design of XtremWeb. Two essential features of this design are multi-applications and high-performance. Accepting multiple applications allows institutions or enterprises to setup their own Global Computing applications or experiments. High-performance is ensured by scalability, fault tolerance, efficient scheduling and a large base of volunteer PCs. We also present an implementation of the first global application running on XtremWeb.

1 Introduction

The key idea of Global Computing is to harvest the idle time of Internet connected computers which may be widely distributed across the world, to run a very large and distributed application. All the computing power is provided by volunteer computers, which offer some of their idle time to execute a piece of the application. Thus Global Computing extends the cycle stealing model across the Internet.

Over the past years, globally distributed application like SETI@Home [5], Mersenne Prime Search [1] or distributed.net [2] have been more than a proof of concept for this computing model through their popular success and their huge aggregate computing power. Moreover, the recent interest of business companies and the spreading of computational economy [8] should engage more and more consumers and volunteers.

Indeed Internet is pervasive and the number of connected devices is constantly growing. Today there are 93 millions of connected hosts [3]. The computing power of handheld devices such as Personal Digital Assistant quickly increases. Typically, the current generation of handheld device is powered by a 200Mhz RISC processor, has 32MB of RAM, wireless communication capability and can be enhanced with an additional 1GB microdrive capacity storage. Considering also the revolutionary expansion of the mobile devices market such as cellular phones, the number of exploitable computing resources might be in the billion order in the next future. The challenge is to harness so many unused computing resources to build a *Very Large Parallel Computer*.

The XtremWeb project springs from a request from physicists of the Pierre Auger Observatory for meeting large computing requirements. Each year, they need to run the same simulation program on 6.10^5 different inputs. The equivalent computing power is 6.10^6 hours of a 300Mhz PC per year. Many other institutions rely on cycle stealing [6] or meta-computing [10, 4] techniques for class of applications that express trivial parallelism, the so-called *multi-parameter* or *embarrassingly parallel* applications. Following the SETI@home success story, Global Computing is getting more and more appealing for institutions. For instance [11] enumerates several SETI@home twin projects, each one developing its particular application and gathering its own set of volunteers. This raises two issues: first the lack of a general tool, toolbox or service for building a global computing application; second the spread of volunteers PCs among this different projects.

The XtremWeb project aims at building a platform for experimenting with global computing capabilities. Some issues to be addressed are: sizing the environment components (servers, network, workers) according to the applications features; high performance and secure execution (relies on sandboxing); modeling resource and workload management as inputs for scheduling algorithms; and the impact of the application characteristics, either compute- or data-intensive.

The next section presents the main issues of global computing systems. The XtremWeb general architecture is presented in section 3. The first implementation of the worker and the server in section 4. Section 5 exemplifies a typical application and discusses the particular problem of very large execution and the last section concludes.

2 Global Computing Issues

The aim of the project is to address a large range of issues related to a global computing system:
Scalability It must scale to hundreds of thousands nodes, with corresponding performance improvement.
Heterogeneity Across hardware, OS and basic software.
Availability The owner of a computing resource must be able to define a policy which limits the contribution of the resource, and the system must enforce this policy.
Fault tolerance The MTBF of an Internet-connected workstation has been evaluated to 13 hours which gives a failure rate of one workstation every 2 minutes in a 10K configuration. The architecture must accept very frequent faults while maintaining performance.
Security All participating computers should be protected against malicious or erroneous manipulations, and the result of the global computation should not be tampered with.
Dynamicity The system must be able to accommodate a continuously varying configuration as well as varying communication latency and throughput.
Usability The system should be easy to deploy and to use.

3 Real World Global Computing

XtremWeb is designed to provide a Global Computing framework for resolving different applications, on projects lead by institutions, commercial firms or open source communities. The following subsections presents the overall architecture of XtremWeb from this particular point of view.

There are two ways of using Xtremweb:
As a *Volunteer*. A Volunteer first registers to the XtremWeb administration server, then downloads the worker software and installs it on a single machine or on a network of computers. While idle, the Volunteer machine or machines will collaborate to a Global Computation.
As a **Collaborator**. A Collaborator also registers to XtremWeb administration server. Collaborators may download the whole XtremWeb software, that allows them to setup their own global distributed application. The XtremWeb administration server asks the Collaborators for an agreement. That agreement allows XtremWeb to exploit the leftover part of the resources which are gathered by a Collaborator. Thus a Collaborator community of PCs will work for the main XtremWeb only when 1) they are idle and 2) the Collaborator XtremWeb server has no work to send to his community of PCs. The next step toward full peer-to-peer computing is to allow the Collaborator applications to be spread over the set of XtremWeb servers.

Figure 1 gives an overview of how a Global Computing system could be organized. Volunteer PCs collaborating to the XtremWeb server (0), range from PCs farms owned by institutions with a high bandwidth Internet connection (1) to family home PC with a sporadic connection (2). XtremWeb allows collaborators to rule their own Global Computing Experiments (5,6) to distribute their applications. The system is flexible enough to allow some servers to be reconfigured to achieve special tasks (4) like the collection of results. In a more speculative way may also think of collaborating (6) between servers to aggregate resources or trading (7) between firms for profit. Finally an institution may also setup its *Intranet* XtremWeb (3) without outside links.

3.1 Security and Native Code Execution

XtremWeb targets high performance. Thus, although the workers protection suggests execution in a virtual environment, typically sandboxed Java bytecode, performance dictates that the end-user code should remain native.

Like many other Global Computing systems, XtremWeb uses native code execution. However, XtremWeb allows any worker to execute different and downloadable applications.

Currently, to become downloadable, the applications follow an ad-hoc verification process. First only trusted institution can propose codes to integrate in XtremWeb. Second, the code is tested on dedicated workers. Third the code is encrypted before downloading to workers. Fourth, the code download procedure uses a private-public key to secure the transaction. Fifth, the worker sends back a checksum of the code to the server and the server verifies its correctness. This verification process cannot prevent from any fault case because testing (second phase) may not execute all code section with all possible parameters sets. A more flexible way to allow downloadable high performance native code execution is the technique known as Software Fault Isolation. This kind of approach is necessary to allow the execution of any application without deeply checking the application before the execution. We plan to evaluate this approach in a future version of XtremWeb.

3.2 Communication Protocol between Worker and Server

All the communications are initiated by the worker. This allows an easier deployment because firewalls may block incoming requests from the server located outside of the local network. Note that this prevents from using slow protocol like http. The protocol between workers and server is independent of the communication layer. This layer may be generalist like TCP-UDP/IP sockets or a higher level layer such as CORBA or Java RMI. This implementation

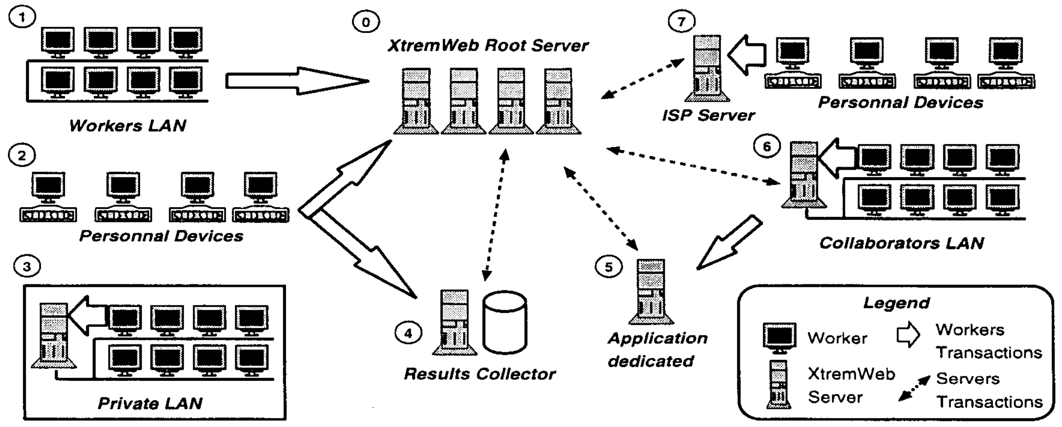

Figure 1. Overview of the XtremWeb Global Computing Framework.

also allows to use more specialized protocols such as SSL or RSVP.

Figure 2 shows the protocol between Workers and Servers. A worker is a machine identified by its name and its owner. The protocol between worker and server consists of four requests detailed below:

- The first request *hostRegister* goes to the last contacted server or to the root server. This first connection authenticates the server to the worker. The server sends back what is called a communication vector. The communication vector specifies the list of servers that may provide tasks to the worker and the communication layer (protocol and port) on which they may be contacted. In the simplest case, the server may return its own address.
- Next, the worker asks for a job from the server, through the *workRequest* request. The worker provides a description of its runtime environment (e.g. operating system, architecture, etc.) and the list of the binaries previously downloaded and stored in a local cache directory. According to this information, the server selects a task, and sends back to the worker a description of the task, the task inputs, the binary of the application corresponding to the runtime of the worker if necessary, and the address of a server that is able to store the results.
- During the computation the worker periodically invokes *workAlive* to signal its activity to the server. The server continuously monitors these calls, to implement a timeout protocol. When a worker has not called for a sufficient long time, the worker is considered down and its task may be rescheduled to another worker.
- At the end of the computation the worker sends back results to the specified address, through the *workResult* call. This call is echoed back to the server which has provided the work, so as to signal the completion of this piece of work.

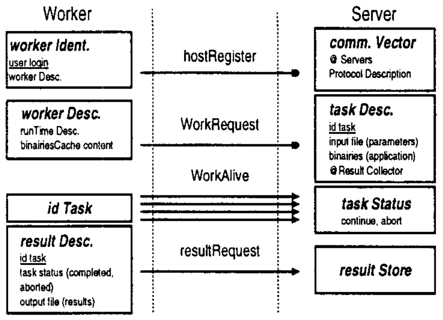

Figure 2. The Protocol between the Worker and the Server

4 Architecture

4.1 Worker Architecture

The worker has two main functionalities: providing the resources of the Volunteer machine to the execution of the XtremWeb computation, and executing the application provided by the server. For a wide acceptance of a Global Computing project, the design of the worker software should put emphasis on the respect of the Volunteer. Thus, the worker architecture stresses user-configuration capability, non-intrusive environment and security.

4.1.1 Virtualizing the Worker Resources

The Volunteer decides when XtremWeb may run a computation and what resources the computation may use. The availability of a given machine depends on the user presence (detected through the keyboard or mouse activity), the presence of non-interactive tasks (detected through the CPU, memory and I/O usage) and other conditions like night and day for instance. Resource utilization is continuously monitored by the worker. An interface to the resources is provided by Operating System features, e.g. the /proc directory for the Unix OSes. A user defines an availability policy simply by indicating for each resource a threshold above which the computer is usable for a Global Computation and a threshold that provokes the interruption of the computation.

4.1.2 Implementation

The implementation of the XtremWeb worker is mainly written in Java, the calls to specific OS functionalities are written in C, integrated through the Java Native Interface. The choice of the Java language allows the core of the XtremWeb to be easily ported over different architectures. Currently, the worker is available for download for Linux Intel86 and Windows Platforms. We plan to port the software to Linux on PowerPC, to Solaris on Sparc and Windows Pocket for handheld devices. Note that this part of the system is not related to performance, but to portability and security.

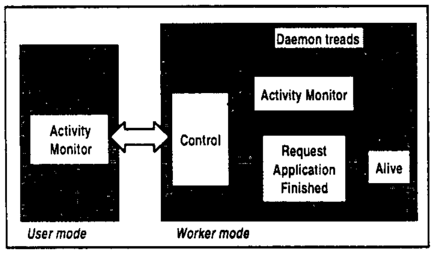

Figure 3. Architecture of the worker

Figure 3 shows the worker architecture. In user mode, a background thread running at low priority monitors the computer activity, following the availability policy. When the computer becomes available, a new thread is launched. This thread starts with a control thread, that creates a monitoring thread, a compute thread and an alive thread, and waits forever for the monitoring thread to terminate. The compute thread runs the binary of the global application. When the monitoring thread detects an increase in the machine load (except its own activity) or the presence of the User, it terminates immediately, causing the other threads to die. The compute and alive thread run as Java daemon threads. This implementation ensures that whatever synchronization is implied in the invocation of remote method on the dispatcher, threads cannot become deadlocked.

We provide a tool called xwLan to simplify the installation of the worker software. This tool has two purposes: configuring the pool of machines, selecting the machines to use and ensuring that the worker software runs on the selected machines. This software is written in the Perl scripting language as a set of object-oriented modules. The system administrator may configure his LAN with a graphical user interface front-end, but may also develop its own custom scripts to maintain the installation.

4.2 Server Architecture

XtremWeb servers are responsible for hosting applications (in form of pre-compiled binaries for different architectures) and tasks submitted by clients and for providing these tasks and applications to the workers willing to compute.

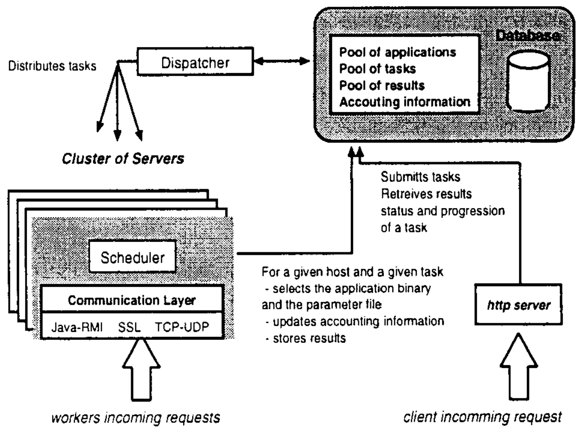

Figure 4. Architecture of the Server

Figure 4 is an overview of the server architecture. The server is composed of several modules that we describe below:

Pool of Applications XtremWeb focuses on sequential multi-parameters applications. Distributing an application consists of providing pre-compiled binaries for different platforms. It also needs a description of the application providing its name and defining how the application handles its parameters and what are the results of the applications. It is also possible to provide different versions of the application (upgrading on the worker is automatic).

Pool of Jobs A client submits tasks to XtremWeb by providing the reference to the application and a set of parameters that defines a set of tasks. Parameters may be several files, a file read for the standard input of the application, or arguments on the command line. For certain types of multi-parameters application, an external tool (like Nimrod/G [4]) should be used to translate the variation of each parameter into a set of files, each one describing one task. When the task is stored in the database, it receives a unique *task identifier*. This *task identifier* refers the task for all the subsequent operations (scheduling, accounting, and result retrieving).

Accounting modules Accounting modules store informations about the tasks. These informations are the reference to the host which have performed the jobs (or a list of hosts if the tasks have been interrupted), the client who submitted the tasks, and various timestamps concerning the tasks.

4.2.1 Scheduling and Servers Specialization

The scheduler in XtremWeb is decoupled in two parts:

Dispatcher The dispatcher selects a bunch of tasks from the task pool and distributes that bunch to the scheduler. The task selection currently follows a very simple scheme. First the dispatcher determines the foremost application to provide the task. Priority is configured by setting minimum ratios of tasks running per application. Second, the dispatcher selects the last recently submitted task for that application.

Scheduler The scheduler has the responsibility of getting a bag of tasks done. The tasks are currently scheduled in a FIFO scheme. When a worker requests a work the scheduler first selects the task that may be run by this particular worker according to its run time environment and the existence of a pre-build binary of the application. The scheduler also detects aborted tasks by maintaining timeout on the *workAlive* signals (see sub-section 3.2) and reschedules it when necessary.

The dispatcher and scheduler policies are dynamically configurable. New policies can be defined (e.g. LIFO, or much more complex according to specific application/resource criteria), and hot-plugged into the server, through the Java Reflection API.

The server throughput is increased by using cluster of tasks servers. Servers run the scheduler module and one of the servers runs the dispatcher and delivers tasks to the servers. Load balancing among the servers may be achieved through an external tool, able to sustain high throughput, typically via mechanisms supported by OS kernel (such as the Linux Virtual Server) or DNS aliases. Specialized servers may be dedicated to collecting the results so as to reduce the traffic to the root servers.

4.2.2 Implementation

The Server implementation mostly uses free software components that are available on most operating systems, and Java, which is inherently portable. The database software is MySQL, but most of the languages we use (Java, PHP and Perl) provide vendor neutral access to database (like Java DataBase Connectivity or Perl DataBase Interface) so changing to another database from a different vendor would not require many modifications of the code. Volunteers and administrators of XtremWeb can interact with the system through a Web interface : to submit tasks, follow the progression of the tasks, retrieve the results and for Volunteers to have the ranking of their team.

5 Application

5.1 Background

The Pierre Auger Observatory [9] project is an international effort to study the highest energy cosmic rays, above 10^{19} eV. The origin of such very high energy cosmic rays is completely unknown. In fact, until the fortuitous detection of two events above 10^{20} eV, the theory did not allow them to happen. Such events are extremely rare and cosmic rays cannot be directly observed at the earth level. However, when cosmic ray particles (primaries) strike the earth's atmosphere, collisions with air molecules initiate cascades of secondary particles, called *air showers*. Two giant detector arrays, each covering 3000 km^2, will be constructed in the Northern and Southern Hemispheres. The objective of the arrays is to measure the arrival direction, energy, and mass composition of cosmic ray air showers above 10^{19} eV during many years.

Air showers must also be numerically simulated, in particular by the AIRES [12] (*Air Showers Extended Simulation*) program. The simulated results will be compared against the actual observations to infer the primaries characteristics described above, during all the experiment. The number of independent simulations to be executed is very large: the simulation is based on a Monte-Carlo scheme, requiring many runs with the same input parameters to compute averages. models. 300MHz PC per year. At this step of our work and of the Auger experiment, the XtremWeb project is a tentative to deliver complementary resources, to the production of the classical high-performance computing facilities.

5.2 Implementation of AIRES on Top of XtremWeb

A sequential execution of Aires lasts between 5 and 10 hours on a 300Mhz PC, but the execution time directly depends on the thinning parameter. Larger execution (more precise) may be launched using a lower calling value. Execution time can be predicted with reasonable accuracy from the input parameters and the worker machine performance. Thus it is unlikely that a remote host would stay available during this time.

There are two approaches to circumvent this problem:

The first approach is to use the built-in checkpointing mechanism of Aires. This is enabled by indicating within the parameter file the number of checkpointing operation per run. Aires saves the whole set of particles composing the air showers, states of variables and the already computed results in a file format called *Internal Dump File* (IDF). The simulation can continue from the saved state on the same machine during the next period of availability or the IDF file may be send back to server as a result and a new task is created which have the IDF file as parameter, allowing a kind of "process migration".

However, this solution has several drawbacks: 1) The IDF file requires a large disk space (up to several dozens of MegaBytes), thus it may only be activated with User's permission. 2) if the task needs to be rescheduled, this would required an incremental checkpointing solution that may stress the network traffic to the server.

The second approach is to downsize the granularity of an Aires execution.

Basically, AIRES manages a stack of particles composing the Air Shower. A simulation starts by stacking the primary particle. Then, the simulation proceeds step by step following an iteration process until the stack is empty. For each particle, a "faith" is probabilistically computed: either annihilation, new particles creation or storing in a result buffer. As a consequence, the number of particles inside the stack evolves during the execution, decreasing or increasing at each iteration.

There is no collision between particles belonging to deferent sub-showers. So, the global shower can be viewed as a hierarchy of independent showers.

Downsizing Aires consists in making custom sub-tasks by following three steps:

- The server starts an execution of Aires with the original parameter file until the iteration count has reached a threshold.
- Sub-tasks are defined by special parameters for each sub shower. These parameters describe the stack part that will be used for the sub-shower simulation. The stack is partitioned according to an iso-energy scheme (i.e. the partitions of the stack have the same total energy).
- The server generates sub-tasks by producing a special IDF file that contains only the part of the stack belonging to the sub-shower. Finally sub-tasks are inserted in the pool of tasks.

A parallelized version of Aires [7] has been implemented, to check the effectiveness of iso-energy partitioning the stack. Performance evaluation shows a nearly linear speed-up on a 64 nodes cluster of Pentium Pro. Without iso-energy partitioning, scalability is limited to four nodes. This experiment demonstrates that the iso-energy partitioning of the stack is a relevant method for creating custom sub-tasks with predictable execution duration.

6 Conclusion

In this paper, we have presented XtremWeb, a framework for experimenting the Global Computing model.

The XtremWeb architecture exhibits several properties. By supporting native execution, XtremWeb may be used by institutions for setting up their own Global Computing system based on their legacy applications. High Performance relies on efficient scheduling and servers specialization. Future work will develop extensive testing on the XtremWeb architecture to evaluate quantitatively its characteristics and limits.

Tools have been developed to enforce the usability of the platform. The first version of the worker software has been released for public download in November 2000, and the whole system should follow shortly. Focus is now on developing the user base of XtremWeb.

Finally we have described the implementation of Aires on top of XtremWeb. This example has shown that XtremWeb is a simple solution for a large scale distribution of a multi parameters application.

Our immediate goal is to make XtremWeb fully usable by Collaborators. In particular we plan to implement security protection with native code execution sand-boxing, safe inter-worker communications, a peer-to-peer model allowing any certified user to submit jobs to anyone. Future work is to find the relevant performance parameters that define and characterize a parallel architecture built through the Global Computing model.

References

[1] Great internet mersenne prime search. gimps discovers 36th known mersenne prime. press release. sept 1997. http://www.mersenne.org.

[2] Rsa labs' 64bit rc5 encryption challenge. http://www.distributed.net.

[3] Internet software consortiom and network wizards – internet domain survey, July 2000. http://www.isc.org/ds/.

[4] D. Abramson, R. Buyya, and J. Giddy. Nimrod/g: An architecture of a resource management and scheduling system in a global computational grid. In I. C. S. Press, editor, *International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia'2000)*, Beijing, China, 2000.

[5] D. Anderson, S. Bowyer, J. Cobb, D. Gedye, W. T. Sullivan, and D. Werthimer. A new major seti project based on project serendip data and 100,000 personal computers. In *Astronomical and Biochemical Origins and the Search for Life in the Universe, Proc. of the Fifth Intl. Conf. on Bioastronomy*, 1997.

[6] J. Basney and M. Levy. *Deploying a High Throughput Computing Cluster*, volume 1, chapter 5. Prentice Hall, 1999.

[7] G. Bosilca, G. Fedak, O. Richard, and F. Cappello. High performance computing with rpc programming style. In *Proceedings of the First Myrinet User Group Conference MUG2000*, September 2000.

[8] R. Buyya, D. Abramson, and J. Giddy. Economy driven resource management architecture for computational power grids. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2000)*, Las Vegas, USA, 2000.

[9] J. Cronin and A. Watson. The pierre auger observatory. http://www.auger.org.

[10] I. Foster and C. Kesselman. The globus project: a status report. *Futur Generation Computer System*, 40:35–48, 1999.

[11] K. Pearson. Internet distributed computing projects, November 2000. http://www.nyx.net/ kpearson/distrib.html.

[12] S. J. Sciutto. Aires : Air showers extended simulation. Department of Physics of the Universidad Nacional de La Plata, Argentina, 1995. http://www.fisica.unlp.edu.ar/auger/aires/.