High Throughput Computing: Stealing Unused Cycles

Luís Filipe Ferreira

Departamento de Informática, Universidade do Minho
4710 – 057 Braga, Portugal
mail@LuFiL.net

Abstract. High Throughput Computing systems (HTC) enable otherwise idles cycles to be available to computations that involve many independent tasks. In a Distributed Computing environment, distributed ownership of computing resources is the major obstacle HTC has to overcome to take advantage of under-utilized systems in the network. This paper addresses HTC and how it fits in current parallel and distributed computing architectures such as Cluster Computing, Internet Computing and Metacomputing. Some of the available HTC packages are presented with a focus on the Condor system. It concludes on the importance of HTC to decrease the gap from personal computing to true parallel departmental computing on existing companies. Finally it addresses the importance of this technology to the future of global computing across multiple organizations (GRID environments).

1 Introduction

Corporate networks are typically composed of several hundreds or thousands of personal computers interconnected over standard LANs. Many of these machines sit idle for long periods of time while their users are absent or busy. Some of them are equipped with top market processors that remain vastly under-utilized performing simple duties like word processing, opening mail or printing documents. These tasks rarely consume more than 10% of the available computing power on a machine.

The fact is that there is a huge amount of wasted computing power which has been paid for, and that aggregately could be the equivalent to a supercomputer. However, if a *distributed computing* structure is about to be set on this environment, some considerations like geographical distance, system heterogeneity, and distributed ownership of computing resources will have to be dealt with. The high latency and low bandwidth that typically characterize these networks will impose some limitations on the granularity and coupling of computational jobs. In that sense, one could say that coarse-grained computations from "embarrassingly parallel" large-scale applications are the best suited for this environment.

High Throughput Computing (HTC) systems fit nicely into these requirements and allow the harnessing of all this wasted computing power. These systems are able to deliver large amounts of computational power over a long period of time. The emphasis is not on providing high performance on the execution of a single job (High Performance Computing - HPC) but rather a high throughput for a lot of jobs with a common goal (hence the name). HPC environments are often measured in terms of Floating Operations Per Second (FLOPS) while HTC environments are more concerned in how many jobs can complete over a month or a year.

This paper presents HTC systems as the supporting technology for distributed parallel computing over corporate or departmental networks. These could range from a typical LAN within a small company to a big set of interconnected networks within a large organization. Some of these systems have a clear orientation towards Internet Computing and allow the remote execution of computational jobs outside the corporate network.

Section 2 presents the main problems this technology has to overcome, and how this re-

flects on the design of its components and on the features they have to support. A distinction is made between native HTC systems (NHTC) like Condor, and Internet Computing oriented ones (IHTC) like those derived from the SETI@home project (ex: Entropia).

Section 3 relates this technology to other distributed computing technologies like Grid Computing and places HTC systems within parallel computing architectures.

Existing HTC systems are presented in section 4 together with other related systems that also have to support this computation paradigm. Special emphasis is put on the Condor system as a true natively oriented system to HTC computing.

Finally some conclusions are drawn on the last chapter regarding the future of parallel computing over corporate networks and the importance of HTC to global computing across multiple organizations (Grid computing).

2 Components and Features

2.1 Components

HTC systems are composed of three essential software components:

- Job Providers, responsible for submitting jobs for remote execution.
  In native HTC systems (NHTC), they can be found within every user's machine. Job binaries will remain on the submitting machine until they are transferred to the executing machine.
  Internet oriented HTC systems (IHTC) tend to place this control on special privileged users and within the scope of a larger application on whose behalf the jobs will be running. Applications are then placed within a server from where job binaries are to be transferred.
- Job Executants, sometimes called Agents,
  These must be placed on every machine where jobs are to be remotely run. These jobs are typically run in a dedicated manner until they are completed, cancelled, or migrated into another machine. Heterogeneity of these machines is an important issue, especially in IHTC systems.
- Job Managers, responsible for job scheduling and monitoring and related functions.
  An important task is that of matching computational jobs to executing machines. Job Managers are run within one or more management servers that control the HTC environment.

2.2 Features

Low network performance is a big issue in HTC. This performance is also affected by the overhead on communication protocols (ex: TCP/IP). In order to maximize the efficiency of such an environment, computational jobs must have a sufficiently coarse-grained nature. The time they take to migrate over the network into a remote computer and to return their results to the submitting machine must be reduced compared to the time they take to execute on the remote machine. Message passing between jobs should also be reduced to a minimum. The ratio between computation and communication time must be maximized. These job requirements are typical of applications sometimes referred to as "embarrassingly parallel".

Another important issue is the fact that the machines in the network are "owned" by their personal users. These distributed owners are only willing to include their resources in

a HTC environment when they are convinced that their needs will be addressed and their rights protected. They want to control the way their resources are "offered" to external usage. They want to be able to say that only certain external jobs can be scheduled to their machines, only during certain periods of time, and if there's no keyboard or mouse activity for more than n seconds and processor activity is low. In the end it all goes down to compromising enterprise and user needs with user rights.

There are some important features that HTC systems should be able to support in order to overcome these and other problems:

- Adaptive load distribution.
  HTC systems have to deal with a dynamic pool of resources that is constantly changing as machines are added/subtracted or becoming available/unavailable. There must be a way for the system to adapt its job scheduling mechanism to these constant changes.
- Matchmaking.
  Matchmaking allows for intelligent scheduling policies where a match is made between computational jobs (resource consumers) and executing machines (resource owners). HTC systems should be able to deal with a complex set of definable requirements and preferences on the jobs and on the machines.
- Fault-tolerance.
  As the pool of resources increases the probability of failure in a particular machine gets higher. Even job management servers may fail. Long duration computations may be lost owing to these failures. There must be a way to recover from such events and resume the execution of jobs (see Checkpoints).
- Checkpoints and Migration.
  Checkpoints involves saving all the work a job has done up until a given point. If a particular machine crashes or is rebooted, the work can always be resumed from the last checkpoint. Periodical or asynchronous checkpoints may be imposed on computational jobs to prevent general hazards. Checkpoints also occur whenever a job is moved from one machine to another, which is known as "process migration".
- Security.
  When someone submits a set of jobs for remote execution in an HTC environment, they will be scattered across machines belonging to several other users. Intrusion may happen on the network when jobs are transferred or results are returned, or on the remote machines themselves. Security measures must be taken to prevent this from happening and reinforce system trust.
- Monitoring.
  Monitoring is an important task. Gathering statistical information about jobs that have been run in the past and on the machines will enable a more effective management of available computing resources.

3 Parallel Computing Architectures and HTC

Parallel computing in general has progressively moved from tightly coupled custom made systems, like Supercomputers, to loosely coupled systems built out of commodity components, like dedicated Clusters [7]. This move has been dictated by the ratio between total computational power and cost. Moore's Law and technology improvements on local network performance have assured it. Clusters have proved cost-effective in computations

where high performance computing on a single job is required.

In highly parallel large-scale applications composed of coarse-grained computations (HTC), scalability is an important issue. These applications enable an almost linear increase of performance as the number of processing nodes increases. When coarse-grained computations are the case, network performance is no longer such a big issue like in HPC systems. These high throughput applications impose the need for a high increase on the number of computing nodes with the smallest cost possible. These requirements have made possible the transition from High Performance Computing to Distributed Computing. High throughput applications no longer need a dedicated Supercomputer or Cluster within a closed environment, to cater for their computing needs. They may rely on already existing interconnected machines, within typical LANs, or scattered around the world in a global wide scale.

Distributed Computing is a very big "word" that embraces lots of different technologies. HTC oriented systems are one of these technologies and allow the harnessing of unused processor cycles across a corporate network. Two other computing technologies that fit within the scope of Distributed Computing will now be presented.

Internet Computing. Internet distributed computing has become popular with the SETI@home project. In this project, a web user may voluntarily download a screen-saver that, when activated, performs computations on behalf of the SETI program, in a search for extraterrestrial intelligence. This has inspired other similar projects to appear, including the fore mentioned IHTC systems. These systems allow different applications to be submitted by some research or commercial project as a kind of Application@home project, to be downloaded for execution by remote "well-intended" users.

Internet computing relies on the donation of PC time by thousands or even millions of individuals, which requires a huge public-relations effort that may involve philanthropic issues [6]. Some of these projects have a largely installed base of machines, which sometimes may be leased by other organizations.

Grid Computing. Grid computing [4] is a metacomputing technology that allows general-purpose large-scale resource sharing across multiple organizations. A Grid could be a virtual organization composed of the resources shared between those organizations, or the result of interconnecting several virtual organizations into a wider Grid system. For that purpose this technology relies on a basic set of Protocols, Services and APIs on which higher-level metacomputing services can be built (such as parallel programming tools and schedulers). It is still a relatively immature technology but with a growing impact on the scientific community. The future of massive parallel computing will most surely rely on this technology to take advantage of the millions of computing resources scattered worldwide, and at the same time being able to comply with the intrinsic details of each one of those resources (Supercomputers, Clusters, HTC pools, disperse Workstations, etc).

4 HTC Systems and Condor

4.1 HTC Systems

It was suggested that HTC systems should be divided into native systems (NHTC) and Internet computing oriented ones (IHTC).

In a NHTC environment like Condor (http://www.cs.wisc.edu/condor), any user is able to submit a job that can be put by the system into any other user's machine. IHTC systems, like Entropia (http://www.entropia.com), United Devices (MetaProcessor – http://

www.ud.com), and Parabon (Frontier SDK – http://www.parabon.com) inherit their basics from their Internet Computing background. They focus their attention on the applications themselves and tend to neglect user needs in favour of corporate needs delegating network machines to the role of mere executants (although they all must supply an application submission component for special users). This communication puts the emphasis on NHTC systems rather than IHTC systems, as the best mean of extending the network's computing resources to the users of these machines.

Other related systems that are able to support HTC are the Globus system [5] (http;//www.globus.org), a standard de facto in Grid computing, and Legion http://legion.virginia.edu), which is an object based metacomputing system. Legion differs from Globus mainly in its architecture and design principles. Globus can be characterized as a "sum of services" architecture, while Legion is an integrated architecture. Globus may interact with Condor in many ways, as described in the next section. It is also planned to interact with Legion and Entropia.

4.2 The Condor System

The Condor system [1,2] was built from the start as a "native" HTC system. It allows users to take advantage of idle machines in the network by providing uniform access to distributed owned computing resources. In a way, this system can be viewed as a cluster of heterogeneous workstations with non-dedicated nodes.

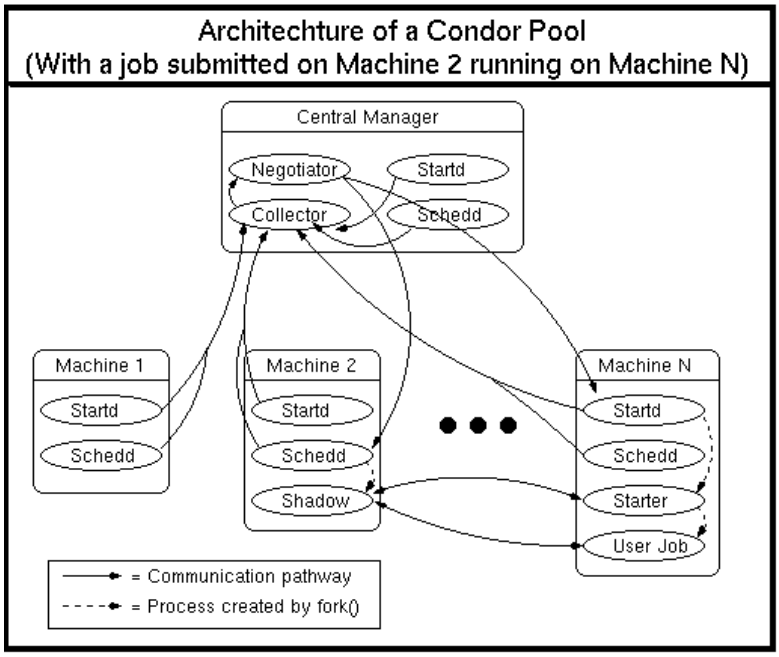
Heterogeneous machines in the network, on which resources are to be shared, are grouped into a management unit called a "Condor Pool". One machine, the "Central Manager" (CM), keeps track of all the resources and jobs in the pool.

For every computer in the Condor Pool, certain programs called the "Condor daemons" must run all the time. The "schedd daemon" keeps track of all jobs that have been submitted on a given machine and represents a user of the Condor system. The "startd daemon" monitors the status of a machine and is responsible for initiating the remote execution of a job on that machine. Both the "schedd" and "startd" daemons report to another daemon, in the CM, called the "collector". The collector maintains a global view, and can be queried for information about the status of the pool. Another daemon on the CM, the "negotiator", periodically takes information from the collector to find idle machines and to match them with pending jobs. Jobs are then transferred directly from the submitting to the executing machine.

Condor allows the submission of many jobs at once in a unit called a "cluster". Each job within a cluster is called a "process". A "Job ID" is composed of the cluster number and the process number.

Standard Condor jobs allow Checkpoints and the Migration of jobs. Jobs are checkpoints into a file on the submitting machine. A "Checkpoint Server" can be installed at a Condor pool, which is a single machine where all checkpoints are stored.

When a "standard" job is submitted into a Condor pool it waits in its job queue until the CM matches it with a given remote machine. The Condor daemons on each machine are then sent a message by the CM. The "schedd" on the submitting machine starts up another daemon called the "shadow" and the "startd" on the executing machine also creates another daemon, "starter". The "starter" actually starts the job, which involves transferring the binary from the submitting machine. It also monitors job execution, supports checkpoints, and vacates the job from that machine in the event the machine is reclaimed by its owner. In standard Condor jobs, distributed ownership issues are handled by redirecting system calls on the User Job in the executing machine back to the original submitting machine, into the job's "shadow" daemon (Remote System Calls).



Vanilla jobs do not support Checkpoints or Remote System Calls, these features are provided in Standard jobs because their binaries have to be previously re-linked with the Condor libraries.

Condor allows complex Matchmaking policies through the ClassAd mechanism. Job requirements and preferences, advertised by users in submitting machines, are matched against resource requirements and preferences advertised by owners in executing machines. These can be described in terms of powerful expressions, resulting in Condor's adaptation to nearly any desired policy.

Flocking is a feature of Condor that allows different pools to be hooked together. A user is then able to "flock" into other remote Condor pools. The list of remote pools is a property of the Schedd, not the Central Manager, so different users can "flock" to different pools and remote pools can allow specific users. A pool's local users can have priority over remote users "flocking" in.

In a way, Condor allows the unification of different computing technologies by supporting distinct Universes on which different types of jobs may be executed:

- Standard Universe, where "standard" jobs are run (not supported by Windows yet).
- Vanilla Universe, where "vanilla" jobs are run.
- Scheduler Universe.

These jobs run on the submitting machine and are supplied together with a Direct Acyclic Graph (DAG) structure. This structure represents dependencies between regular jobs (ex: do not run job B until job A has successfully completed) and serves as a meta-scheduler managing the submission of these jobs to Condor based on DAG dependencies.

MPI and PVM Universes. These Universes enable access to traditional High Performance Cluster Computing systems, providing Condor has been installed on all the dedicated Cluster nodes and directed to the CM. Parallel message passing jobs built on MPI and PVM standards will then be allowed for submission.

Globus Universe. Installing Condor-G activates this Universe. Condor-G [3] is the bridge that connects Condor to Globus, the standard in Grid computing technology. Globus jobs

can be submitted to this Universe into Grid machines all over the world and still take part on Condor's queue management features.

Another important feature supported by this Universe is the ability to send a "GlideIn" job into a Grid machine, which will temporarily install the Startd daemon onto that machine and add it to the Condor pool. Standard, Vanilla, PVM, or MPI Universe jobs will then be matched and run on Globus resources.

5 Conclusions

5.1 Parallel Computing in Corporate Networks

HTC systems are already available for commercial use and companies with higher computational needs can largely benefit from them. Several factors could assert this technology as a major step towards parallel computing in corporate networks:

- Cost: companies, especially smaller ones, aren't still very willing to consider a costly investment on a dedicated HPC structure like Cluster computing (although much cheaper than a Supercomputer). HTC systems make it possible to take advantage of the already available computing resources spread across under-utilized machines on the network. However some investment may still be put to consideration: HTC software (although free use may be considered for some situations like Academic research), probably one or two Servers for job management, and the cost associated with supporting the system.

-Flexibility: at the same time, users are not still willing to fully share their computing resources in a dedicated way. HTC systems allow these users to satisfy their computational needs while preserving their ownership rights. Policies are generally supported that allow a machine's owner to keep control on the way its computing resources are integrated into the system (i.e. which jobs are executed under what circumstances). Other policies can also enable the definition of requirements and preferences on the applications themselves. This should lead to an effective matchmaking policy between computational jobs and executing machines.

Security considerations also play a very important role on supporting these mechanisms.

- Expandability: these systems generally allow some form of Internet Computing or the ability to hook into other kinds of computing structures like Clusters or Grid environments (Condor). On that sense HTC systems can function as a starting point for the unification of several parallel computing technologies leaving the user with a wide range of computing alternatives for their applications.

However there are still some drawbacks on the use of this technology:

- Finer grain limitations
  HTC systems were built taking in consideration the high latency and low bandwidth of standard corporate LANs or even WANs. This may imply that only coarse-grained computations are fit to take advantage of these systems. However, the constant decreasing of network latency and increasing of available bandwidth could certainly shorten the distance from HTC to HPC systems, and enable finer-grained integration on the system (the name would probably have to change to HTPC).
- Resource sharing limitations
  Current HTC systems are built to take advantage of under-utilized *computing* resources on the network. This means taking advantage of available processor cycles and program memory on the machines. Other *general* resources like secondary memory and other I/O devices are generally not taken into account. Metacomputing systems are typically more suited for that purpose. However, general client-server technology should be able to fill

that gap in some way.

All these considerations validate HTC systems as an important technology in the parallel computing arena. Common scientific and compute intensive applications can gradually support HTC environments natively so that idle computing resources in the network can share the application's load in a transparent way. Similarly, companies can gradually consider a small investment in an HTC structure to take advantage of those applications, which in turn will encourage further developments in HTC technology. In that sense, this could just be the self-feeding spiral that would lead the way to a small revolution in parallel computing.

5.2 HTC Systems and Grid Computing

In many ways Grid computing has a lot in common with HTC systems and can largely benefit from its technology. They are both suited for large-scale parallel computing across geographically distributed computing resources where high network latency, low bandwidth and adaptive policies still impose limitations on computation granularity. We could say that Grid computing must support HTC. However, keep in mind that Grid computing is a much more general-purpose structure with a higher abstraction level on which even HPC can be supported as well.

One of the biggest evidences of the relation between Grid Computing and HTC is the way in which Globus (a Grid Computing Infrastructure Toolkit) and Condor (an HTC system) can cooperate (section 4.2). Condor allows for jobs to be executed on the Globus Universe and even supports a way of migrating its job scheduling mechanism into a Globus environment (GlideIn jobs). On the Globus side Condor can provide a uniform view of processor resources and implement a given Site's Job Scheduling policy.

In a more practical sense one could figure out a particular company that could use Condor as a way of managing its internal computing resources in a tightly controlled and unified manner. This company would then resort to Globus and Grid computing for accessing external computing or other general-purpose resources, or for sharing these resources with the outside world.

References

[1] Litzkow, M.J., Livny, M., Mutka, M.W.: Condor – A Hunter of Idle Workstations, Proceedings of the 8th International Conference of Distributed Computing Systems, IEEE (1998)

[2] Couvares, P., Tannenbaum, T.: Condor Tutorial, First EuroGlobus Workshop, Computer Sciences Department, University of Wisconsin-Madison (2001)

[3] Frey, J., Tannenbaum, T., Livny, M., Foster, I., Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids, Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC10), (2001)

[4] Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid, Enabling Scalable Virtual Organizations, International Journal on Supercomputer Applications, (2001)

[5] Foster, I., Kesselman, K., Globus: A Metacomputing Infrastructure Toolkit, Argonne National Laboratory, & Information Sciences Institute, USC (1996)

[6] The Economist Newspaper Group Ltd.: Computing power on tap, "http://www.economist.com/PrinterFriendly.cfm?Story_ID=662301", (2001)

[7] Buyya, R.: High Performance Cluster Computing: Architecture and Systems, ISBN 0-13-013784-7, Prentice-Hall PTR, NJ, USA, (1999).