Lessons learnt from cluster computing: How they can be applied to grid environments

Alberto Sánchez [a], Jesús Montes [a], María S. Pérez [a], Toni Cortes [b], Pilar Herrero [a]

[a] *Universidad Politécnica de Madrid, Madrid, Spain, {ascampos, jmontes, mperez, pherrero}@fi.upm.es*

[b] *Barcelona Supercomputing Center and the Universidad Politécnica de Barcelona, Barcelona, Spain. toni.cortes@bsc.es*

Abstract

Grid computing seems to be one of the most promising initiatives in the computing field. This discipline can be seen as an evolution of cluster computing, since it also keeps a good cost/performance relation. Besides, both disciplines, cluster and grid computing, are oriented to similar application fields, i.e., the resolution of complex problems. However, a large number of differences exists betweeen both paradigms. The main distiction between a grid and a cluster is the fact that a grid is composed of different administrative domains, whose resources are managed in a non centralized way. This involves a deep analysis with the aim of identifying those solutions, which have been succesfully applied in clusters and could help to solve challenging problems in grids. This paper outlines these solutions, emphasizing their different application in cluster and grid environments.

Key words: Cluster, Grid, Operating system, Load balancing, Programming models, Data managers, Transparent access, Security.

1 Introduction

Clusters entered the fray in the 1990s, mainly due to the development of the known Beowulf project [5]. Although other works related to clusters had been previously performed in universities and research centers, this project showed up that commodity clusters can be a very attractive alternative to expensive and powerful supercomputers. It is widely recognised that the key of this success is twofold:

- *Low cost:* The combination of inexpensive PC's can compete with costly supercomputers for providing high performance solutions.
- Availability of *powerful tools* and *open source software* for clusters. The most significant examples are the Linux operating system and programming libraries such as MPI and PVM.

Ad-hoc clusters have been built by a large number of academic and research institutions. The advantages offered by clusters have allowed "humble" research groups to run parallel and costly algorithms in a flexible and economic fashion. However, some problems have arisen in the *cluster world*. One of these problems is the maintainability and/or expandability of clusters. The economic advantage obtained by the use of commodity components only can be kept if clusters are extensible by means of addition of new hardware. However, a large number of tools and libraries are optimized for its use in homogeneous clusters. Furthermore, a cluster should provide a *single system image*. This is not possible in the case of multiple organizations cooperating for the resolution of hard problems. Many times, since the enormous computational power required by grand scientific challenges, resources of several organizations are needed to solve them. In these scenarios, the use of grids is more suitable.

A grid is not just a cluster of workstations used for aggregating power. The main distiction between a grid and a cluster is the fact that a grid is composed of different administrative domains, whose resources are managed in a non centralized way. The single system image concept contrasts with grids, where resources take a relevant role.

Nevertheless, both disciplines, cluster and grid computing, are oriented to similar application fields, i.e., the resolution of complex problems. Despite the scope of both of them are different (cluster computing is focused on High Performance Computing and grid is more suitable for High Throughput Computing), we believe that most of the advances performed in cluster computing can be applied to grids. Different appproaches have emerged in this direction. This paper tries to work out the sinergies between cluster and grid computing, with the aim of taking advantage of the huge work performed by scientists in the area of cluster computing.

The organization of this paper is the following: Sections 2 and 3 describe both computing paradigms and the problems they try to deal with. Then, Section 5 shows the grid challenges that could be solved by means of cluster-oriented solutions. Finally, Section 7 outlines the main conclusions of this study.

2 Cluster computing

From the first Cray to the very last IBM Blue Gene, supercomputers have been designed to solve highly calculation-intensive problems, in the cutting edge of science. Military applications, physical simulation, molecular modeling, weather forecasting, and financial analysis are a few examples in supercomputer's area of competence. On the other hand, mainframes were built for critical applications: financial transactions, bulk data processing, or statistical analysis. Contrary to supercomputers, mainframes focus more on reliability, availability and serviceability than pure CPU power. Both supercomputers and mainframes involve custom hardware and specific design to reach high end performances, thus making them expensive equipments.

Over the last decade cluster computing has emerged as an alternative to both traditionally specific-design supercomputers and mainframes. A cluster can be seen as an integration of computers, networks and software that intends to provide a single system image [6]. Computers used in clusters, also called nodes, can be conventional tower cases, single rack unit SMP systems or even blade servers, which provide a greater CPU density. Nodes do not usually use specific design or custom hardware. Typically two classes of nodes can be distinguished: the computing nodes, that only perform computing tasks, and one or more master nodes, in charge of the file system, the data storage, the system monitoring, and the clustering middleware. The network is used to provide high speed and low latency communications, most popular includes gigabit or ten-gigabit Ethernet, Myrinet or Infiniband networks.

This architecture provides two crucial advantages against supercomputers or mainframes. First, scaling a cluster is as easy as adding or removing nodes. Secondly, the cost versus computing-power ratio makes clusters an affordable solution to solve problems that were previously cost-prohibitive for many institutions.

Depending on the software and the middleware, the cluster can be a high-availability cluster, a load balancing cluster, or a high-performance computing cluster, thus handling transaction intensive workloads, data I/O intensive workloads, compute intensive workloads, or mixed workloads. High-availability and load balancing clusters do not run cluster-aware applications. Each node performs an entire task. In high availability clusters, if a node is taken out or service fails, the load is transferred to another node. In load balancing clusters, requests are dispatched between nodes, but each node handles the same type of requests. Load balancing clusters are typically used for web-hosting solutions.

On the other hand, high-performance computing (HPC) clusters run cluster-aware applications. Each task is divided into multiple subsets, distributed in multiple nodes. HPC cluster fits CPU intensive tasks such as scientific analysis, or financial data analysis. Several approaches have been found out to manage this task division. Message-passing models such as MPI [26] or PVM [33] have been developed to allow fast message communication and synchronization between nodes. In another way, MOSIX [25] tends to provide a single system image, handling load balancing through preemptive process migration instead of request dispatching.

The availability of powerful cluster software and tools as open source and/or free software, such as MPI and PVM, OpenMOSIX [27], BSD and Linux, is a key factor in the popularity of cluster computing. It allowed the emergence of so-called Beowulf HPC clusters, made without any custom hardware component, running a free UNIX-like operating system, and capable of handling complex algorithms in a flexible fashion for a reduced cost.

3 Grid computing

Nowadays, the scientific community is facing some problems that even flexible, powerful systems as clusters are not capable to solve. This is the case of the Grand Challenge problems [17], which can be solved but determining their solutions may require computational resources that exceed those that can be provided by a single organization. Problems like ecosystem simulations, molecular biology, nuclear weapons simulations or fluid dynamics can be considered of this kind. It is necessary, in consequence, the development of new technologies that allow scientific investigation to deal with these situations.

Grid computing means an innovative step in trying to solve Grand Challenge problems, enabling applications to take advantage of widely separated computational resources belonging to several different organizations in a secure and reliable way. The term grid first appeared in [20], inspired by the ideas behind the Electrical Power Grid: Uniformly provide, from any point and at low costs, access to resources. Thus, a grid can be seen as a set of computer systems interconnected through the Internet or a corporative WAN, aiming to perform great computational tasks. It has no centralised control, and allows the aggregation of geographically dispersed resources into a theoretical virtual supercomputer. From the user point of view, the grid automatises the access to computational resources, assuring security restrictions and reliability.

In [12], Ian Foster defines the main characteristics of any grids:

- Non centralized control of resources, enabling different administration policies and local management systems
- Use of open protocols and standards.
- Providing quality of service concerning performance, availability and security.

One of the main advantages of a grid infrastructure relies on the fact that it is specifically designed to be scalable and adaptable. It can grow and include from a few resources to millions of them and it is prepared to support unexpected failures or loss of their resources. This is achieved in a larger order-of-magnitude than in cluster computing, where systems can only grow in a controlled way to a certain extent, after which the most profitable solution can be to change entirely the system. These features provide theoretical unlimited computational power to the grid, due to the expandable set of systems that can be attached and adapted. Furthermore, its decentralized administration allows independent administrative domains (such as corporative networks) to join and contribute to the system. These domains do not lose any administration control during the process. This is a great advantage over traditional distributed systems, that are always subject to a centralised control involving a grown limitation. As well, the grid enables the integration of heterogeneous systems. It can be achieved by the use of open protocols and standards interconnection and collaboration between diverse computational resources. This can not even be considered in cluster computing, due to the paradigm architectural limitations. Nevertheless, in spite of the differences between clusters and grids, both technologies are focused in a good cost/performance ratio.

As a summary, the grid can be considered as a new environment where the computational possibilities are expanded and immensely diversified. These new conditions can help the scientific community to face Grand Challenge problems in a way that has not been possible so far. Nevertheless, grid computing is still an innovative technology, with many aspects left to explore and refine.

4 Similarities and differences between cluster and grid computing

5 Lessons learnt from cluster computing and their application to grids

In this section, we plan to present a set of functionality, optimization, and/or mechanisms that have been successfully applied in cluster environments and that should be applied (with possibly some minor modifications) to grid environments. Our impression is that both worlds, although sharing a lot of common problems, do not share many of the solutions, and thus synergies that would greatly benefit both worlds are missed.

5.1 Integrating grid solutions into the OS

The first, and probably most important solution is also the most difficult to achieve. Most cluster solutions [39],[4],[38] have some patches and/or modules in the kernel to make sure the operating system is aware of some distributed concepts such as parallel application, unique process ID, etc. On the other hand, grid solutions are fully implemented as middleware and the operating systems underneath have no idea about what is running on top [18],[11],[19].

This integration of grid concepts into the kernel would have many interesting implications such as the ones described here:

- Better resource enforcement. If the operating system is aware of grid applications, it can implement policies that really restrict the number of resources requested by the application. For instance, in current system, there is no way to limit the number of total forks in an application. Currently it has to be done on a per-node basis or detected after some time and the react to this "not-allowed" use of resources. In this way we could enforce global resource consumption of applications. Although efficient policies will need to be devised, the first step is to include the concept of job into the kernel.
- More accurate job monitoring/accounting. As the concept of job is not in the kernel, and the integration between the middleware and the OS is very loose, the information does not flow well enough and plenty of information is lost in the way. For instance, error codes are lost in many systems or the status of the different parts of a job is not easily accessed by the user. In the same line, if the kernel is involved, the accounting can be done in a much more accurate way than if it is not.
- Interaction with jobs in the traditional way. Again, if jobs are known by the kernel, we can implement traditional interaction mechanisms like signals,

that are very useful. These mechanisms are available for applications in clusters, but not in the grid.
- Transparent application check-pointing. Finally, given the high probability of node failure in grid environments, check-pointing is a key issue in these systems. As current environments do not have support from the OS, application checkpoint is either not transparent or only functional for a set of resources but not all (i.e. I/O, IDs, etc.).

We are aware of the complexity of integrating grid issues in the operating system, not for the technical problems, but because resources used by a grid application may come from different administrative domains and using a specific OS may not match the local policy. Nevertheless, some basic support could be proposed and then try to push it to different operating systems. In this way, XtreemOS is taking a first step trying to integrate some grid concepts into Linux [24].

On the other hand, it would be possible to build a complete grid operating system. In this way, the GridOS project [29] is working to provide OS support to the common grid middleware. Some other interesting aspects for turning grid middleware into a grid operating system are shown in [1].

5.2 Load balancing and better resource usage

Migration of processes is something common in clusters that is not used to its potential in a grid environment. It is true that it makes no sense to try to balance all nodes in a grid to a similar level (we may be talking of millions of nodes and traditional load balancing would not scale). Nevertheless, what can be done is twofold. On the one hand we can unload very loaded resources and on the other hand we can take advantage of better resources that become available.

- Unloading nodes. If a given node becomes too loaded, it would make sense to be able to migrate part of its jobs to another node that is much less loaded as it is done in cluster systems. It is true that current grid systems assume exclusive access of the jobs to the resource and thus this migration may just consist on moving not-started jobs from one node to another (which is currently supported by many systems). The problem is that in real grids, we will not be able to assume such exclusiveness [24], and interactive processes will also be executed in nodes running parts of grid job. In this case, this load balancing will be a must.
- Migrating to better resources. Although the appearance of better nodes in a

cluster is not a frequent task (it is normally a much more stable environment than a grid), there has been some work computing the benefits of migration to another node for load-balance issues [2]. All this work should be extended to the grid to evaluate if migrating a job to new and better resources makes sense or not (performance wise).

5.3 Use of programming models

The use of programming models enables developing appropriate programs in an efficient way. This is especially useful in complex environments, like a cluster and moreover a grid. Therefore, several cluster-based programming models han been conceived in the last years, standing out MPI, PVM, RPC and OpenMP [28]. Some of these approaches could be help to define grid programming models because of grid and cluster similarities. A complete taxonomy of grid programming models where most of them are based on clusters is described in [21].

MPI is one of the most known approaches that has been adapted to the grid, by means of the definition of MPICH-G2 [16]. RPC is other programming model that has been sucessfully applied to both, clusters and grids. GridRPC and OmniRPC are grid-based implementations of this model. Other programming models such as OpenMP could be used in the grid if some efforts are devoted in adapting the granularity they deal a step further than for clusters. For instance, there has been some work trying to adapt OpenMP granularity on run-time according to the needs of the system where it is being executed [10] or change the use of OpenMP to higher granularity [3].

5.4 File systems vs. data managers

Current grid systems manage data in a very naïve way. As there is no global file system, the grid relies on the local file systems and copies data to the nodes where the job is executed. In addition, some systems, try to reduce this data movement maintaining data replicated a different sites, but this is just a patch to the problem [8],[35],[34]. The real solution again goes in the same direction clusters went, implementing a global file system that can be used from all nodes. Implementing such a global file system instead of the current data managers would have the following advantages, among many others:

- Global view of the system. The first advantage of such a system would be a global view that would simplify the deployment of applications. Libraries could be in this shared file system, users could have their working directories

globally visible, among other benefits that would make grid usage much simpler than today.

- Allow parallel access to files. Another advantage of a global file system is that optimization such as parallel access to files in the way of stripping [22],[30],[23],[31],[37] or using in parallel potential replicas (hidden to the user) would become possible. In current system, we can only rely on the file system nodes have which is normally a sequential one.
- Allow optimizations such as cooperative caching. In the same line, we can also implement traditional cluster ideas such as cooperative cache where the memory of remote nodes is used to reduce access to disk [9]. Of course, this mechanism would need to be adapted from the small range of cluster to the large range of the grid and only allow remote caching within a small distance.

Several approaches to this grid file system have arisen. Most of them, such as GFS from OGF or XtreemFS [7], are still in a very initial state. However, the cooperation between MAPFS-Grid [32] and GAS broker [36] constitutes a complete solution to this problem, although there are open issues even to be solved. GAS provides a global view of the file system discovering files and monitoring resources, while MAPFS-Grid is in charge of efficiently accessing files in a parallel way.

5.5 Transparent access

Another typical characteristic of applications found in clusters is the contact point regardless of the location of the application. Once the application is created, other application can contact it using the same IP address even if it is migrated. This is another feature that grids have inherited and adapted from the cluster world.

In 2002 the grid community changed its orientation to a services model [13]. This new architecture, named Open Grid Services Architecture (OGSA) provides both the creation and the maintenance of the different services offered by several organizations. Specifically, grid services are a considerable extension of web services. In this sense, whereas web services are stateless, grid services are stateful and support notifications and life cycle management,

Since grid services are an evolution of web services, these are accessed by using a similar method named Web Services Addressing. WS-Addressing [40] provides a more versatile mechanism than URI (Uniform Resource Identifiers) to manage services and resources. Thus, grid services are transparently ad-

dressed in a similar way than web pages, using EndPoint References (EPR). An Endpoint Reference is the contact point of a service instance. It contains the required address and metainformation to interact with the service. Each grid job has associated to a unique identifier. This id enable recovering the EPR where the job is running, hiding the access complexity and providing reliability.

5.6 Security

Since there is not a centralized control in a grid, resources can be distributed in a wide area network crossing organizational boundaries. This makes possible that they could be accessed by several users of different organizations. Therefore, it will be necessary to control the accesses to the resources. In [14] the concept of virtual organization is defined, showing its boundaries. It must be made sure that only certain organizations or users can access certain resources, and especially that they are really who they claim that they are. This involves the security will be a key factor in grid computing.

Grid Security Infrastructure (GSI) [15] transparently interacts with common remote access tools, such as remote login, remote file access, and programming libraries like MPI, to provide security. GSI is based on public-key cryptography, and therefore can be configured to guarantee privacy, integrity, and authentication.

The concept of *single sign-on* meant a great improvement in cluster systems, since it makes easy the access to them. Although in the case of grids, this probably could imply an interaction with the operating system, the need to support single sign-on for grid users have been recovered by GSI and there is a complete implementation. The single sign-on provided by GSI includes delegation of credentials for both programs or others users that run on behalf of a user and require a subset of the user's rights to access resources. Nevertheless, there are some security problems, mainly related to the power of the resource provider, that are not clearly solved yet.

6 MAPFS-Grid, a case of study

7 Conclusions and Future trends

This article has shown the similarities between cluster and grid computing. From these sinergies, the article has identified those *hot topics* where some ideas extracted from cluster computing can help to improve current grid solutions. The topics presented in this article are by no means complete, these are the core issues we consider they could be the starting point for this sharing of solutions. In addition, some of them have started to be addressed in some projects, but have not made their way through yet.

From our point of view, the most important aspects which should be addressed in grids with the help of cluster solutions are: (i) the integration of grids and operating systems, (ii) load balancing approaches, (iii) programming models, (iv) grid file systems, (v) transparent access and (vi) security. The first three aspects are those topics in which the existing initiatives are still in an early stage. Deeper research advances has been performed in the rest of the fields. As the grid research progresses, the success of the ideas analysed in this study could be demonstrated not only with these incipient projects but also actual approaches.

Acknowledgment

We thank the XtreemOS consortium and especially to Julita Corbalan, Erich Foch, Felix Hupfeld, Christine Morin, and Gregor Pipan for the interesting discussion we have had in the last year that have been the seed for some of the ideas discussed in this paper. This work has been partially supported by the Spanish Ministry of Science and Technology under the TIN2004-07739-C02-01 grant and two UPM pre-doctoral grants.

References

[1] Arshad Ali, Richard McClatchey, Ashiq Anjum, Irfan Habib, Kamran Soomro, Mohammed Asif, Ali Adil, and Athar Mohsin. From grid middleware to a grid operating system. In *GCC '06: Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06)*, pages 9–16, Washington, DC, USA, 2006. IEEE Computer Society.

[2] Gabriel Antoniu, Luc Bouge, and Raymond Namyst. An Efficient and Transparent Thread Migration Scheme in the PM2 Runtime System. In *Proceedings of the 11 IPPS/SPDP'99 Workshops Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, pages 496–510, London, UK, 1999. Springer-Verlag.

[3] Jairo Balart, Alejandro Duran, Marc Gonzlez, Xavier Martorell, Eduard Ayguad, and Jess Labarta. Experiences parallelizing a Web Server with OpenMP. In *First International Workshop on OpenMP (IWOMP 2005)*, 2005.

[4] Amnon Barak and Oren La'adan. The MOSIX multicomputer operating system for high performance cluster computing. *Future Generation Computer Systems*, 13(4–5):361–372, 1998.

[5] Beowulf.org: The Beowulf Cluster Site.

[6] Rajkumar Buyya, Toni Cortes, and Hai Jin. Single system image. *The International Journal of High Performance Computing Applications*, 15(2):124–135, 2001.

[7] Eugenio Cesario, Toni Cortes, Erich Focht, Matthias Hess, Felix Hupfeld, Bjrn Kolbeck, Jess Malo, and Jan Stender. Jonathan Mart. XtreemFS - an object-based file system for large-scale federated IT infrastructures. In *Linux Tag 2007*, Berlin, Germany, May 2007.

[8] Ann Chervenak, Ewa Deelman, Ian Foster, Leanne Guy, Wolfgang Hoschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripeanu, Bob Schwartzkopf, Heinz Stockinger, Kurt Stockinger, and Brian Tierney. Giggle: a framework for constructing scalable replica location services. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–17, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[9] T. Cortes and J. Labarta. PAFS: A new generation in Cooperative Caching. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98)*, Las Vegas (Nevada), July 1998.

[10] J. J. Costa, T. Cortes, X. Martorell, E. Ayguade, and J. Labarta. Running openmp applications efficiently on an everything-shared sdsm. *J. Parallel Distrib. Comput.*, 66(5):647–658, 2006.

[11] Dietmar W. Erwin and David F. Snelling. UNICORE: A Grid computing environment. *Lecture Notes in Computer Science*, 2150, 2001.

[12] Ian Foster. What is the Grid? A Three Point Checklist. *Grid Today*, 1(6), Jul 2002.

[13] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, January 2002.

[14] Ian T. Foster. The anatomy of the grid: Enabling scalable virtual organizations. In *Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*, pages 1–4, London, UK, 2001. Springer-Verlag.

[15] Ian T. Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. In *ACM Conference on Computer and Communications Security*, pages 83–92, 1998.

[16] MPICH-G2.

[17] Grand Challenge Applications.

[18] The Globus Alliance.

[19] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, and Paul F. Reynolds Jr. Legion: The next logical step toward a nationwide virtual computer. Technical Report CS-94-21, Department of Computer Science, University of Virginia, August 1994.

[20] Carl Kesselman and Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, November 1998.

[21] Craig Lee and Domenico Talia. Grid programming models: Current tools, issues and directions. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*, pages 555–578. Wiley, 2003.

[22] Lustre: scalable, secure, robust, highly-available cluster file system.

[23] N. Miller, R. Latham, R. Ross, and P. Carns. Improving Cluster Performance with PVFS2. *ClusterWorld Magazine*, 2(4), April 2004.

[24] Christine Morin. XtreemOS: A Grid Operating System Making your Computer Ready for Participating in Virtual Organizations. *ISORC'07. 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, 00:393–402, 2007.

[25] MOSIX.

[26] MPI - the Message Passing Interface standard.

[27] openMosix, an Open Source Linux Cluster Project.

[28] OpenMP: Simple, Portable, Scalable SMP Programming.

[29] P. Padala and J. N. Wilson. GridOS: Operating System Services for Grid Architectures. In *International Conference on High Performance Computing*, 2002.

[30] Panasas.

[31] María S. Pérez, Jesús Carretero, José M. Peña Félix García, and Víctor Robles. MAPFS: A flexible multiagent parallel file system for clusters. *Future Generation Computer Systems*, 22(5):620–632, 2006.

[32] M.S. Pérez, A. Sánchez, P. Herrero, and V. Robles. *A New Approach for overcoming the I/O crisis in grid environments*, chapter 19, pages 311–321. American Scientific Publisher, January 2006.

[33] PVM: Parallel Virtual Machine.

[34] A. Rajasekar, R. Moore, B. Ludascher, and I. Zaslavsky. The GRID adventures: SDSC'S storage resource broker and Web services in digital library applications. In *Proceedings of the 4th All-Russian Scientific Conference (RCDL'02) Digital Libraries: Advanced Methods and Technologies, Digital Collections*, Dubna, Russia, 2002. Joint Institute for Nuclear Research.

[35] A. Samar and H. Stockinger. Grid Data Management Pilot (GDMP): A Tool for Wide Area Replication in High-Energy Physics. In *Proc. of IASTED International Conference on Applied Informatics (AI 2001)*, Innsbruck, Austria, 2001.

[36] A. Sánchez, M. S. Pérez, J. Montes, P. Gueant, and T. Cortes. A Prediction-based Autonomic Storage Architecture for Grids. *Special Issue of Journal of Autonomic and Trusted Computing*, (accepted, to appear), 2007.

[37] Frank Schmuck and Roger Haskin. Gpfs: A shared-disk file system for large computing clusters. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, page 19, Berkeley, CA, USA, 2002. USENIX Association.

[38] Thomas L. Sterling, John Salmon, Donald J. Becker, and Daniel F. Savarese. *How to build a Beowulf: a guide to the implementation and application of PC clusters*. MIT Press, Cambridge, MA, USA, 1999.

[39] Geoffroy Vallée, Renaud Lottiaux, Louis Rilling, Jean-Yves Berthou, Ivan Dutka Malhen, and Christine Morin. A case for single system image cluster operating systems: The kerrighed approach. *Parallel Processing Letters*, 13(2):95–122, 2003.

[40] WS-Addressing specification.