3-2005

Guest Editor's Introduction: Cluster Computing

George K. Thiruvathukal
Loyola University Chicago, gkt@cs.luc.edu

Recommended Citation

George K. Thiruvathukal, "Guest Editors' Introduction: Cluster Computing," Computing in Science and Engineering, vol. 7, no. 2, pp. 11-13, Mar./Apr. 2005, doi:10.1109/MCSE.2005.33

CLUSTER COMPUTING

What is cluster computing? In a nutshell, it involves the use of a network of computing resources to provide a comparatively economical package with capabilities once reserved for supercomputers.

Beowulf.org, home of the Beowulf cluster-computing project, provides a more specific definition:

> *Cluster* is a widely used term meaning independent computers combined into a unified system through software and networking. At the most fundamental level, when two or more computers are used together to solve a problem, it is considered a cluster. Clusters are typically used for High Availability (HA) for greater reliability or High Performance Computing (HPC) to provide greater computational power than a single computer can provide.

As we might expect from perhaps the best-known clustering project, the Beowulf project provides the most specific definition for a Beowulf cluster:

> Beowulf clusters are scalable performance clusters based on commodity hardware, on a private system network, with open source software (Linux) infrastructure. The designer can improve performance proportionally with added machines. The commodity hardware can be any of a number of mass-market, stand-alone compute nodes as simple as two networked computers each running Linux and sharing a file system or

1521-9615/05/$20.00 © 2005 IEEE
Copublished by the IEEE CS and the AIP

GEORGE K. THIRUVATHUKAL
Loyola University Chicago

as complex as 1024 nodes with a high-speed, low-latency network.

The Beowulf project has done much to shape the dialog of cluster computing, and to a large extent, it defines cluster computing as we know it today. Most clusters have similar hardware characteristics: an Intel-based processor (Pentium, AMD Athlon, Itanium, or Opteron), standard storage (Enhanced Integrated Drive Electronics [EIDE], Serial Advanced Technology Attachment [SATA], or Small Computer Systems Interface [SCSI]), some form of networking (Ethernet, or sometimes Myrinet), rack-mounted or in a blade design (to enable more dense packing), and running the Linux operating system. For the most part, clusters use software similar to that used in traditional supercomputing, including Fortran, C, C++, the message-passing interface (MPI), and scientific programming libraries such as Linpack and Lapack.

The Revolution

In this issue, we look at certain applications of cluster computing to problem solving. As the Beowulf project and clustering revolution celebrate more than 10 years in existence, it's interesting to see what remains the same and what has changed. Let's look at a few aspects of the clustering revolution in more detail.

Commodity Hardware

The components we use to build a system are called *commodity hardware*. This definition often implies PC-compatible hardware, but it also includes Macintoshes and hardware that doesn't look much like a PC at all (such as rack-mounted servers and blades). The distinction between commodity and proprietary is often blurred, but ultimately, true commodity status refers to the ability to obtain components (or even complete systems) on the open market and leverage economies of scale. Suppose, for example, that you want to build a cluster of PCs. You have a choice of many vendors for motherboards, processors, system memory, and hard drives, but more than that, you can select an appropriate price point for your performance needs.

Networking

For the most part, we build supercomputers from specialized networks. The parallel computing literature is supersaturated with research on so-called interconnection networks, which are specialized to allow optimal (or near optimal) interprocessor communication in large-scale parallel systems.

For clusters, commodity networking technology such as Ethernet dominates the market. Ethernet emerged as the unlikely hero when many of the ideas of interconnection networks were pushed gradually into network switches, allowing nodes within the cluster to use the Ethernet controllers largely as access devices only.

However, clusters aren't limited to using Ethernet. Other solutions such as Myrinet offer reduced communication latency when every last drop of network bandwidth must be utilized.

Open Software Infrastructure

Although Linux has emerged as the operating system of choice for cluster computing, not all of the contributors to this special issue rely on it exclusively. We intentionally solicited articles from researchers using other operating systems, to demonstrate that clustering is becoming more ubiquitous and isn't limited to the Linux platform.

We didn't receive any contributions from researchers using Windows. However, I worked on a Windows-based cluster computing application (in computational finance) this past summer. Virtually any major operating system can be used and can attest to its potential for cluster computing, subject only to personal taste and software availability.

Top500 Supercomputers

To see the true impact of clustering, take a look at the Top500 list, which lists the 500 most powerful supercomputers. Although not all of these systems are clusters, a quick sampling reveals that most of them are:

- The Barcelona Supercomputer Center (ranked fourth) is built entirely with eServer blade systems from IBM.
- Lawrence Livermore ranks fifth with its Itanium2 cluster.
- The Virginia Tech cluster (ranked seventh) runs Apple XServe systems and Gbit Ethernet.

Ranking at the top is IBM's Blue Gene, which uses many of clustering's ideas, but is, in fact, a hybrid of a cluster and a pure supercomputer. As IBM describes it at www.research.ibm.com/bluegene/,

> Blue Gene is an IBM supercomputing project dedicated to building a new family of supercomputers optimized for bandwidth, scalability, and the ability to handle large amounts of data while consuming a fraction of the power and floor space required by today's fastest systems.

It goes without saying that getting on this list will not be cheap. Although most of these systems use commodity components, some cost more than others. Choosing the latest and greatest processor, for example, will always run up the tab. If you're building a system with 1,024 processors, a US$100 price difference on the CPU alone increases the overall cost by $10,240.

Ten Years Running

With the Beowulf project recently celebrating its 10th anniversary, it's appropriate to look at the state of cluster-computing research. Cluster computing is an expansive field, and it would take multiple special issues to cover it completely. Rather than attempting to do that here, we opted to look at clustering via representative research projects that are aimed at scientific application development.

About the Contributions

In this issue, I solicited articles with the intention of addressing two distinct goals:

1. how to get started with cluster computing, especially for those who have never done it before, and
2. how to get the most out of cluster computing and address serious application development that scales well.

In some cases, an article addresses both of these goals. The first two articles are focused on "getting started," whereas the last two are focused on "getting the most out of cluster computing." Without further ado, here's a summary of the contributions.

Matthias K. Gobbert takes us back to where it all started by describing how to build a 64-processor Beowulf cluster with a high-performance interconnect and later extend it with a storage solution from IBM. He demonstrates each system component's role via a prototype problem from the numerical solution of transient partial differential equations. The problem shows how the judicious combination of a numerical algorithm, its efficient implementation, and the right hardware can achieve the two fundamental goals of parallel computing: solving large problems faster and solving larger problems than can be solved on serial computers.

Dean E. Dauger and Viktor K. Decyk describe cluster computing as practiced in the Plasma Physics Group at the University of California, Los Angeles. They discuss plug-and-play computing, focusing on the Macintosh platform running OS X. Motivated by a desire to avoid system administration, they chose the Macintosh for its ease of use and straightforward system administration. This article is a must-read for anyone who wants to apply clustering in an open-source environment, but faces limited resources for actually managing the cluster.

Phil Hatcher, Matthew Reno, Gabriel Antoniu, and Luc Bougé describe their research on using Java for cluster computing, presenting two approaches for making it work. The first views the cluster as a single computer running a single Java virtual machine (JVM) that can implicitly spread Java threads (lightweight processes or tasks) among the cluster's nodes. The second approach uses the mpiJava framework to run a JVM on every node in the cluster. The results show that good-to-excellent performance can be achieved in an environment that isn't typically associated with high-performance computing.

James D. Terseco, Jamal Faik, and Joseph E. Flaherty describe resource-aware cluster computing, including the challenges many smaller institutions face. If someone develops an application on a local cluster, for example, possibly comprising a relatively small number of nodes, and then migrates it to a large-scale cluster at a national laboratory, the authors' software techniques could prove useful for making the application aware of the change in computing resources and adapting to the new environment.

I hope to organize another issue (or annual issue) on this topic, simply because it can't be covered in only four articles; please contact me if you'd like to contribute. Cluster computing is in a continued state of evolution: a good thing that just keeps getting better! The articles included here provide tremendous insight into the general issues of clustering and what it takes to get supercomputer-class results.

Acknowledgments

I thank Norman Chonacky and Denis Donnelly for their help in attracting articles for this issue, and the reviewers, Steve Chiu (Benedict College), Wei-Keng Liao (Northwestern University), and Nick Karonis (Northern Illinois University) for their assistance.

George K. Thiruvathukal is an associate professor and graduate program director in the Department of Computer Science at Loyola University Chicago, and president/CEO of Nimkathana Corporation, a small-business concern based in Chicago. He has worked extensively on cluster and Grid computing and operates two Linux clusters running the Gentoo Linux operating system. He has a PhD in computer science from the Illinois Institute of Technology. Contact him at gthiruv@luc.edu or gkt@nimkathana.com; www.thiruvathukal.com.