ELSEVIER

Future Generation Computer Systems 18 (2002) v–vii

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.com/locate/future

Guest editorial

# The best papers from CCGrid 2001

Cluster and grid computing are experiencing incredible growth in virtually all areas of application, e.g., academic, scientific, engineering and commercial. Cluster computing, of course, is the concept of building a parallel computer using a "cluster" of commodity processors and networks. Grid computing is the concept of building *virtual organizations* or *distributed supercomputers* or *high-throughput computers* using networked, distributed resources world-wide. While the simplest clusters have the same processors on a single network switch, many clusters will have different processors connected by a switch hierarchy. Such *heterogeneity* will allow clusters to evolve as new processors and networks become available, and also enable *scalability* as their size increases. It is important to realize that such heterogeneity can make a cluster look more like a grid.

Given the all-inclusive nature of what is a grid resource, cluster computers will be integral to many grid configurations. Clusters and grids are clearly "joined at the hip" and share many of the same technical challenges. One of the most fundamental issues is performance, e.g., how to manage communication and scheduling in heterogeneous and dynamic environments, or how to manage data movement to improve latency tolerance and load balancing. This can be translated into the management of communication systems, file systems, and execution models. Clusters and grids will present a nonuniform communication hierarchy and topology to the application builder. This includes not only communication between processors but also between processors and files and specialized instruments—in short, any type of *data movement*. While applications can certainly be hand-coded to manage specific capabilities and topologies, the real work lies in systems that can deliver reasonable performance without explicit application-specific knowledge.

Simply building applications in such environments will also be a brave, new world. Web services and network-enabled RPC coupled with portals and scripting tools for component software models will enable flexible applications to be built for clusters and grids. However, this will also demand a host of services from the infrastructure, e.g., information services for resource discovery, allocation, scheduling, and monitoring. A grid service must be able to describe itself with a commonly understood metadata schema (most likely in XML). This service must either be discoverable in a well-known information service or have a well-known *inspection port* whereby potential users can determine the suitability of its functionality and the interfaces it supports.

Security and fault tolerance are also imperative. While isolated clusters can be secured with traditional administrative concepts, any cluster participating in a grid must be able to provide security on a per-user, or even per-call, basis since it will essentially be in an open-ended system. This can require not only certificates that authenticate clients and servers, but also *proxy certificates* that enable the delegation of trust through a chain or tree of operations at different sites. Again, for smaller clusters, fault tolerance can be addressed by traditional means. For larger installations, however, and clusters that participate in a grid, fault tolerance must have more extensive support. Hence, time-outs must be used and communication tools must be guaranteed not to "hang" without providing some type of independent *event messaging* that allows corrective actions to be taken.

In light of these many facts, the First IEEE/ACM International Symposium on Cluster Computing and

0167-739X/02/$ – see front matter 
PII: S0167-739X(01)00079-6

the Grid (CCGrid 2001) was convened on May 15–18, 2001, to serve as the premier world forum for cluster and grid-related issues. CCGrid2001 is a new millennium symposium that has emerged as a follow up and merger of the Asia–Pacific International Symposium on Cluster Computing (APSCC 2000) held in Beijing, China and the International Workshop on Cluster Computing Technologies, Environments, and Applications (CC-TEA) series held in the United States for the past 4 years (1997–2000). CCGrid 2001 was hosted by the Queensland University of Technology (QUT) in Brisbane, Australia, and sponsored by the IEEE Task Force on Cluster Computing (TFCC). It was organized by the General Co-Chairs George Mohay (QUT) and Rajkumar Buyya (Monash University) with the Program Chair Paul Roe (QUT). Complete information is available at http://www.ccgrid.org.

Of the 48 technical papers presented at CCGrid 2001, this volume contains expanded, journal-length versions of the 10 best papers representing a cross-section of the issues facing cluster and grid computing—nominally five in the area of grid computing and five in the area of cluster computing—even through some papers overlap both areas significantly. While making such choices necessarily involves some subjective judgement, the following "VASE" criteria were applied (with a numerical rating from 1 to 10) to aid the selection process:

- *Vision*: did the paper present an overall vision for where a field is going?
- *Approach*: how convincing was the work presented in support of the vision?
- *Significance*: how important was the work presented to the long-term vision?
- *Exposition*: how well-written and understandable was the paper?

Many of the papers selected investigate aspects of performance, and specifically data movement. They also report on file systems, execution models, program development techniques and security.

*Optimizing execution of component-based applications using group instances* presents a *filter/stream framework* for data-intensive grid applications. The specific question investigated is the scheduling of filter groups and whether an existing filter should be reused or a new filter instantiated.

*KelpIO: a telescope-ready domain-specific I/O library for irregular block-structured applications* investigates the concept of using domain-specific I/O libraries, rather than language extensions, such that high-level source code optimizers can produce efficient I/O operations. The term "telescope-ready" refers to the notion of "telescoping languages" where one set of scripts and functionality "telescopes" into another using interprocedural compilation strategies to maintain performance.

*TACO—exploiting cluster networks for high-level collective operations* introduces a C++ template library for topologies and collections. While ordinary topologies, such as lists, trees, and meshes are possible, any topology can be defined by the user. A parallel traversal of distributed graphs is used to implement collective operations. Collections are also dynamic, i.e., new members may join existing collections.

*Latency hiding in dynamic partitioning and load balancing of grid applications* investigates a dynamic partitioning scheme to balance workloads, hide latencies, and minimize communication for adaptive mesh-based applications on the NASA Information Power Grid infrastructure. The MinEX partitioner operates in three phases, contraction, partitioning, and refinement, based on the criterion minimizing the application execution time rather than simply balancing the load at a given time.

*OPIOM: off-processor IO with Myrinet* looks at the hardware aspects of data movement by making disks closer to the network. Rather than serving disk data through a processor and its local memory to a remote consumer over the network, the OPIOM system inserts a new SCSI service in the Linux SCSI stack such that the PCI address in a SCSI request is actually that of a buffer in the Myrinet SRAM. This increases performance and decreases host overhead without additional hardware and cost.

*Armada: a parallel file system for computational grids*, on the one hand, describes a distributed file system for grids where the application can extensively control its behavior. A *harbor* consists of low-level functions that is typically hosted close to the physical location of a data store but can, in fact, be anywhere in between a client and the data. A *ship* in the Armada defines user-level functionality that can be moved between (executed at) different harbors.

*OVM: out-of-order execution parallel virtual machine* presents an execution model that may prove to be very important in distributed heterogenous systems. OVM is similar to the SPMD programming style where the user must specify the work decomposition. However, the user does not explicitly declare the work distribution, the communication, or the synchronization. Since OVM is based on asynchronous RPC, the runtime system can manage this and accommodate systems with deep memory/latency hierarchies.

*XML-based visual specification of multidisciplinary applications* presents a web-based environment for building, executing and monitoring distributed applications. The *Arcade* environment provides a visual and script-based specification interface for heterogeneous modules. XML is used to specify *Project* objects that consist of modules and the dependency graph between these modules. Using this information, a set of data, execution, resource, and security managers can control execution.

*Design of a generic platform for efficient and scalable cluster computing based on middleware technology* presents a similar approach based on the notion of an *Intelligent Agent Platform*. Computational tasks are intelligent agents that can utilize coordination, scheduling, load balancing and migration services to maintain performance.

Finally, *sabotage-tolerance mechanisms for volunteer computing systems* describes a technique to protect "volunteer" computing systems (such as SETI@home) from being sabotaged by a malicious participant. This technique does not rely on checksums or cryptographic techniques. Instead, the established technique of voting is integrated with the new techniques of spot-checking, backtracking, and blacklisting based on the concept of *credibility-based fault tolerance*. This new technique can significantly reduce and limit the overhead and use of redundancy required to protect the computation.

We would like to thank editorial staff members of Elsevier Science: Doutzen Abma, Ruud Koole, and Inge Pompen for their encouragement and enthusiastic support during the preparation of this special issue.

We now invite you to read this collection of papers—not as the final word on any of the topics they cover, but as a point-of-departure for the further work that must be done.

Craig A. Lee
*Computer Systems Research Department*
*The Aerospace Corporation*
*El Segundo, CA, USA*
http://www.aero.org
*E-mail address:* lee@aero.org (C.A. Lee)

Paul Roe
*Faculty of Information Technology*
*Queensland University of Technology*
*Brisbane, Qld, Australia*
http://www.fit.qut.edu.au/~proe.
*E-mail address:* p.roe@qut.edu.au (P. Roe)

Rajkumar Buyya
*School of Computer Science and Software Engineering, Monash University Melbourne,*
*Vic., Australia*
http://www.buyya.com
*E-mail address:* rajkumar@csse.monash.edu.au (R. Buyya)