Scientific Programming 13 (2005) 255–263
IOS Press

Guest-Editorial

Worldwide computing: Adaptive middleware and programming technology for dynamic Grid environments

Carlos A. Varela[a], Paolo Ciancarini[b] and Kenjiro Taura[c]
[a]*Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*
[b]*Dipartimento di Scienze dell'Informazione, University of Bologna, Mura Anteo Zamboni-40126, Bologna, Italy*
[c]*Department of Information and Communication Engineering, University of Tokyo, Japan*

1. Towards a pervasive Grid: A research agenda

Future-generations cyber-infrastructure must enable the dynamic coordinated composition of computing and information services. Resulting applications need to provide high-performance distributed computing to end users in a scalable, reliable and secure manner. This pervasive and ubiquitous grid computing infrastructure will view physical devices and computational agents uniformly, enabling radical improvement of existing applications, and opening the door for applications in entirely *new* domains. For example, *wireless sensor-actuator networks* deployed in buildings and bridges can semi-automatically and cooperatively react to natural or man-made disasters in order to prevent human losses. Temperature, pressure, and stress-sensing devices can inform authorities about the probability of structural damage leading to collapse, can automatically activate fire extinguishing equipment, can help locate survivors and devise exit strategies, and can deviate traffic on emergency situations in semi-automated ways. Another example is *virtual surgical planning*, in which a surgeon simulates several surgery plans on a detailed computational model of a patient's body. The surgeon can interactively analyze the effect of different plans on the patient's body fluid dynamics. Another example is a *distributed camera network* coordinated by a real-time data mining component for airport surveillance and security. Unusual irregular patterns of human behavior can be detected and communicated promptly to authorities. A final example is a *manned and unmanned aerial vehicle network* that can exchange sensed information and plans of action, and combine them with terrain and map databases for decentralized coordinated air traffic control. These applications will directly benefit from coordinated computational resources offered by the future pervasive grid.

To realize this *pervasive grid* vision, it is imperative to develop programming technology and modular middleware to facilitate systems development on highly heterogeneous and dynamic cyber-infrastructure. In closed grid environments, a centralized coordination module often needs to reserve dedicated network and computing resources for specific tasks. Furthermore, users are often expected to manually allocate resources and install any needed software on target computing environments. In contrast, open dynamic grid environments require coordination and resource management protocols to be automated. Resources in open grid environments can be organized into *peer-to-peer networks*. These networks are scalable because information lives in individual nodes and communication is highly, if not completely, decentralized. In an envisioned grid computing scenario, a process or data item will be created by a human user in a single node. The process or data item will get replicated and propagated by middleware

ISSN 1058-9244/05/$17.00

layers to a dynamic network of processing and storage resources. The purpose of the middleware will be to provide high performance through resource profiling and optimization, scalability through dynamic reconfiguration and adaptation, fault-tolerance and data availability through replication, and privacy through encryption. All of these must be provided in a completely *decentralized and automated* manner.

Realistic grid computing environments will be large-scale, dynamic, and heterogeneous. Physical and computational agents will span the globe. New computational nodes will join at any time, for example, because a user points her web browser to a site coordinating a search for extraterrestrial intelligence from astronomical data. Nodes will also leave at any time, for example, because of user-directed actions or because of network or node failures. Furthermore, resources will have very different capabilities and constraints, for example, a small robot in a battle field may have energy constraints. This will prevent the robot from communicating too frequently with its base, thus preferring to send digested abstracts after local communication with nearby robots. Grid computing middleware needs to coordinate such heterogeneous computational and physical agents. Middleware must dynamically adapt applications to changing needs and to evolving underlying computational resources. Research in decentralized coordination and global resource optimization is needed to provide cost-effective services with high reliability and performance to end users.

The fundamental research problems the pervasive grid community faces are:

1. **Decentralized coordination:** Coordination algorithms and protocols must account for physical and computational agents entering, leaving and moving about the system. Decentralized coordination is needed for systems to scale. Decentralized coordination protocols form the foundation for other distributed resource management problems. Modular middleware is critical to plug-in different coordination protocols to adjust to diverse application communication topologies and computation-to-communication ratios.
2. **Resource optimization:** Grid computing middleware needs to continuously map computational agents to underlying network resources. Dynamic resource profiling and allocation(e.g., see [30]) needs new mathematical models that optimize global resource usage with minimal profiling and communication overhead and in the presence of local, partial, and even inaccurate knowledge. One potential interesting avenue of research is novel integrated static and dynamic program analyses for resource optimization.
3. **Programmability:** Usability is critical to the success of grid computing. Research is needed in new coordination models, abstractions, and languages that facilitate describing and reasoning about high-level specifications of distributed resource usage, composition, and constraints. For example, actor-oriented programming models, languages and infrastructure (e.g., see [32,84]) provide intuitive high-level abstractions and efficient middleware components that support these abstractions.
4. **Quality of Service:** Quality of service (QoS) requirements in grid computing application include *scalability* to world-wide computational environments, *reliability* in the presence of node and network failures, and enforcement of *security* policies, including data privacy, user authentication and authorization, and secure remote code execution. There is significant existing grid computing security research and software infrastructure (e.g., see [37,41]) but as grid environments become more open and dynamic, new challenges arise.

2. Middleware and infrastructure for Grid computing

2.1. Globally distributed computing

Several research groups are trying to achieve distributed computing on a large scale. Wisconsin's Condor project studies high throughput computing by "hunting" for idle workstation cycles [58]. Berkeley's NOW project effectively distributes computation on a building-wide scale [8], and Berkeley's Millennium project exploits a hierarchical cluster structure to provide distributed computing on a campus-wide scale [19].

The Globus project seeks to enable the construction of larger *computational grids* [37]. Virginia's Legion meta-system integrates research in object-oriented parallel processing, distributed computing, and security to form a programmable world-wide virtual computer [47]. Caltech's Infospheres project envisions a world-wide pool of millions of objects (or agents) much like the pool of documents on the World-Wide Web today [23]. WebOS seeks to provide operating sys-

tem services, such as client authentication, naming, and persistent storage, to wide area applications [83]. UIUC's 2K is an integrated operating system architecture addressing the problems of resource management in heterogeneous networks, dynamic adaptability, and configuration of component-based distributed applications [55].

Berkeley's SETI@Home project [77] uses idle cycles on the Internet to analyze astronomical data in search for patterns that might prove the existence of extraterrestrial life. Stanford's ProteinFolding@Home [66] has over a million participants worldwide volunteering their processing power in a computational attempt to understand protein folding.

2.2. Grid computing middleware

Grid computing has made great progress in the last few years. The basic mechanisms for accessing remote resources have been developed as part of the Globus Toolkit [38] and are now widely deployed and used. The grid community is now concentrating on middleware that enables the composability and interoperability of different service-oriented grid solution components by using Web Service standards.

The emerging generation of grid middleware, based on the Open Grid Services Architecture (OGSA) [39, 40] is tackling interoperability and composability by using XML-based developments including WSDL for service definition, SOAP for data interchange, and UDDI for service registration and discovery. OGSA specifies the requirements for a layered set of protocols and services as follows: (i) **Connectivity layer**, concerned with communication and authentication, (ii) **Resource layer**, concerned with negotiating access to individual resources, and (iii) **Collective layer**, concerned with the coordinated use of multiple resources.

2.3. Process migration and replication

Research on process and data migration for grid applications includes [73,81,82]. A key idea behind these approaches is to detect service degradations and react to such events by dynamically reconfiguring application processes to effectively use available computational resources. Research on data and process replication includes [5,24,25,64,68,71,75,85]. Most of these approaches only consider immutable data replication to avoid having to devise and implement expensive replica consistency protocols. An adaptive middleware layer is needed, capable of migrating and replicating data and processes proactively based on the dynamically changing availability of resources on the grid [4,30,56,57, 78].

While adaptive process and data migration and replication can have a large impact on the performance of grid computing applications, they both assume a reasonable underlying model of resource usage and expected future performance and availability of grid resources. Two mechanisms to predict performance based on profiling resource usage are the Network Weather Service (NWS) [90] and the Globus Meta Discovery Service (MDS) [28]. Recent research has devised and evaluated different mechanisms for resource management in dynamic heterogeneous grid environments – e.g., see [6,7,10,28,43,49,51,92]. However, how to map application resource needs onto dynamically changing computational resources in a transparent and efficient manner is still an open problem. A promising approach is to use econometric market-driven resource exchange based on supply and demand – the more applications that want to use a resource the more expensive it is, and conversely the more resources that are available to applications the cheaper they need to be to remain competitive. Research in this direction includes [88,89].

An important consideration when adapting applications to dynamic grid environments through proactive data and process migration and replication is that the failure semantics of applications changes considerably. Research on fault detection and recovery through checkpointing and replication includes [16–18,33,74]. Notice that an application checkpointing mechanism is necessary for adaptive application migration and can readily be used for fault tolerance as well. More fine-grained process-level rather than application-level checkpointing and migration requires logging messages in transit to properly restore a distributed application state upon failure [33].

2.4. Application-level support for Grid application development

Although some advances have been made in the area of application-level support for grid applications, much work still needs to be done. An MPI implementation using the Globus Toolkit allows execution of cluster computing programs in grid environments [36,52]. One system that in particular targets numerical relativity is the Cactus system [7]. Cactus allows users to write application-specific modules and interface them to the Cactus framework which can run these modules

on the grid. While enabling communication across heterogeneous environments provides an important step towards enabling grid execution of applications written for cluster environments, the latency incurred over wide area networks makes highly synchronized iterative programs run inefficiently in typical grid environments (e.g., see [13]). Due to the economic need to reuse existing application code, most application-level support for grid computing tries to impose as few requirements as possible on developers [59]. The goal is that applications delegate distribution issues completely to middleware, and therefore remain largely unchanged. However, it is also important to devise higher-level programming models to enable faster software development cycles and more efficient and effective middleware-triggered dynamic application reconfiguration [9,14,45,84]. An open research question is how to integrate application-level load balancing (as, e.g., in adaptive parallel applications [29,34,35,69,72]) with middleware-level load balancing [30] without compromising the efficiency of the former or the transparency of the latter.

2.5. Programming models and systems for adaptive parallelism

Adaptive parallelism refers to parallelism that may change at runtime based on resource availability. Programs with adaptive parallelism are also said *malleable*. Malleability is desirable in Grid environment where availability of resources may change dynamically. It is clearly very difficult for the programmer to write malleable programs without suitable programming models/languages support.

Challenges in supporting such programs involve (1) designing programming models in which writing malleable programs and reasoning about their correctness as well as their localities is easy, and (2) implementing such models with high performance communication and with scalability.

Message passing models such as MPI and PVM are good for the programmer to reason about locality (and thus performance) of parallel programs. On the other hand, directly adding malleability to this model adds significant programming complexity because all the burden of keeping track of memberships and the mapping of data to nodes is on the programmer. Phoenix parallel programming model [79] improves this situation by introducing communication via abstract (virtual) node names.

Message passing programs are difficult to be made malleable because they directly expose "processes" at the programming model level. High-level parallel programming models that hide the mapping between computation/data and physical resources are generally more amenable to adaptive parallelism. This is because the programmer does not use process names for communication and the task of mapping computation/data to resources is in principle up to the system, not the programmer. They include distributed shared memory and object-based models. Their challenges are then how to provide the programmer with a model of locality, with which s/he can reason about communication performance of programs.

An important class of algorithms that has demonstrated an efficient utilization of adaptive parallelism is divide-and-conquer algorithms [15,91]. Such algorithms divide a problem into smaller sub-tasks. Malleability can be achieved relatively easily by dynamic load balancing based on random- or latency-aware task stealing. A nice property is that such dynamic load balancing schemes do not incur much communication overhead if sub-tasks do not have dependencies among them (as is typically the case). Furthermore, under this assumption, fault-tolerance can be achieved relatively simply, by redoing sub-tasks that are lost.

3. Programming and coordination technology

3.1. Models of computation

The complexity of distributed software development and the effectiveness of middleware optimizations are highly dependent on devising high-level programming and coordination models and languages for distributed computing. Nondeterminism, asynchrony, and partial failures make concurrent, distributed and mobile systems much harder to reason about and develop than sequential systems. Theoretical models of concurrency, mobility and distribution help reason about and implement open distributed systems. These models are critical to create appropriate programming abstractions, languages and middleware for global distributed computing. Two major schools of thought for concurrent programming models are process algebras and the actor model.

The main representative of the process algebra school is the π-calculus [63], a simple yet expressive and composable model consisting of *processes* communicating through shared *channels*. The model pro-

vides primitive operations for synchronously reading or writing from channels, for creating new channels with scoped names, and for composing concurrent processes. The calculus has a well-developed theory, including a structural congruence that relates syntactically equivalent process expressions, an operational semantics that provides rules for the evolution of computations, and bisimulations that provide for the semantic equivalence of process expressions. The π-calculus has influenced work in several high-level programming languages, for example, Pict and Nomadic Pict [67, 87]. Other representative calculi include the join calculus [42] and mobile ambients [21].

The actor model of concurrent computation [1,48] defines an *actor* as a unit of concurrency, mobility and distribution in open systems. An actor encapsulates state and reacts to *asynchronous messages* by modifying its internal state, by creating new actors, or by sending messages to other known actors. The model assumes guaranteed message delivery and fairness in scheduling computations. A leading theory of actor computation [3] views actors as functional components and models them as expressions in an extended call-by-value λ-calculus. Open systems are then viewed as composable *actor configurations* and an operational semantics defines valid transitions between configurations. Observational equivalence is used to equate actor systems. Among other applications, actor systems have been used for enterprise integration [80], real-time programming [70], fault-tolerance [2], coordination [20, 44], and distributed artificial intelligence [31]. The actor model has also influenced programming language design – e.g., ABCL, Erlang, THAL and SALSA [12, 54,84,93].

3.2. Coordination models and languages

There has been much recent interest in coordination models and languages [11,26,27,46,65]. Linda [22] is a coordination model that uses a globally shared tuple space as a means of coordination and communication. Additional work in this direction includes adding types to tuple spaces for safety, using objects instead of tuples and making tuple spaces first-class entities [50, 53,61]. Many coordination models rely on *reflection*, the ability to introspect and customize basic behavior (e.g., see [62,76,86,93]). Critical characteristics of a coordination model and language are the separation of concerns it provides (not intermixing coordination and computation code) and its ability to scale to large systems through decentralized communication.

SALSA is a general-purpose actor-oriented coordination language, especially designed to facilitate the development of dynamically reconfigurable open distributed applications. In addition to the actor model's first-class support for unbound concurrency, asynchronous message passing, and state encapsulation, SALSA follows a universal naming model with Internet- and Java-based support for actor migration and location-transparent message passing. Furthermore, to facilitate coordination of concurrent activities, SALSA provides three high-level abstractions for programmers: token-passing continuations, join blocks, and first-class continuations [84]. The SALSA platform provides an important research testbed to evaluate higher-level programming and coordination abstractions. SALSA as a *coordination language*, can be used to develop middleware agents that profile the network, exchange information, make distributed resource management decisions, and reconfigure distributed application components written in other conventional *programming languages* such as C++ or Java [60].

While programming and coordination models and languages help raise the level of abstraction for building grid computing systems, they do not guarantee high performance in distributed execution. Compilation technology and middleware research is needed to ensure efficient execution of distributed applications in dynamic grid environments.

4. Introduction to the special issue on dynamic Grids and worldwide computing

Keahey, Foster, Freeman and Zhang introduce the concepts of *virtual workspaces* and resource capabilities as abstraction layers between grid resources and grid applications. These abstractions enable more efficient matchmaking of distributed resources and applications and also have the potential for improving quality of service (QoS) for grid applications. The authors present two potential instantiations of the virtual workspace abstraction: (i) a **dynamic account,** representing a leased UNIX account mapped to a grid user identity, and (ii) a **virtual machine,** encapsulating and virtualizing access to physical machines. The taxonomy of virtual workspaces, including an XML Schema for representing them, can prove to be an important contribution in the definition, discovery, matching, and deployment of distributed grid resources. The paper presents an authoritative futuristic vision of dynamic grid computing.

Pike, Dorward, Griesemer and Quinlan present a system for massively parallel analyses of very large semi-structured data sets. The system contains a front-end that performs independent computation on locally stored data chunks, and a back-end that combines the results from the independent computations into a single result. While the system leverages results from previous work – mostly the Google File System (GFS) and MapReduce/Workqueue – for fault-tolerance and work distribution, it incorporates two key innovations: (1) a high-level domain specific language, called *Sawzall*, to specify the front-end computations, and (2) a set of so-called *aggregators* to efficiently and conveniently compose the partial results. The system has been successfully used in relatively simple examples to highlight its expressive power and value.

Andreozzi, Montessi and Moretti present *XMatch*, a query language enabling the expression of user requests in terms of the expected satisfaction over XML-based representation of available resources. XMatch enables users to describe their expectations in regards to Grid resources in terms of their preferences. It provides an evaluation method that is the basis for the maximization of the global preference.

Herrera, Huedo, Montero and Llorente describe their experiences on porting scientific applications written in several styles to the Grid environment. They use a proposed standard for distributed resource management API (DRMAA), for a variety of programming idioms, including parameter-sweep applications, master-worker applications, and workflow.

Jeannot, Caron, Desprez, DelFabbro and Nicod propose an extension to Grid RPC systems so that data can persist across multiple remote procedure calls. RPC is a popular programming style in Grid environments, but is often inefficient because of the lack of data persistence. That is, all data necessary for the server to perform their computation must be sent on every RPC, and all data necessary for the client to continue the computation must be sent back to the client on every RPC. They propose an extended API and a scheduler to lower the cost of data transmission in the Grid RPC model.

5. Conclusion

Grid systems enable to integrate and even virtualize geographically dispersed resources offered, and used, by distributed and dynamic virtual organizations. These resources can belong to domains owned by different organizations, making thus difficult their orchestrated working. In fact, current Grid implementations are able to federate worldwide distributed resources across different domains. Nevertheless, there are a number of important issues to be addressed in order to deal with dynamic and adaptive Grid systems. An aspect to consider is the user-oriented selection of Grid resources meant as the capability of identifying the resources better fitting the users expectations. Another aspect concerns the proper configuration of the selected resources so that the user application can profitably interact with them. The interoperability among different Grid implementations is also an open issue due to the lack of adoption of common standards for the management and use of the provided resources. Finally, the definition and study of suitable programming models for exploiting in a coordinated way several Grid resources is a meaningful aspect as well. The aspects mentioned above represent the domain of investigation of this special issue.

Acknowledgements

We would like to thank the authors for their excellent contributions to this special issue, and B. Szymanski and R. Perrott for their guidance and suggestions in putting the issue together. Two articles were invited and three were selected among eleven submissions. All of them were peer-reviewed and we thank the following people for volunteering their time to make this a truly special issue: Sibel Adali, Kento Aida, Marian Bubak, Rajkumar Buyya, Travis Desell, Jim Doherty, John Field, Dan Grigoras, Kenji Kaneda, Jacek Kitowski, Shiding Lin, Malik Magdon-Ismail, Kaoutar El Maghraoui, Dave Musser, Hidemoto Nakada, Tatsuo Nakajima, Keith Power, Marcel Rosu, Eunice Santos, Atsuko Takefusa, Osamu Tatebe, Roman Wyrzykowski, and Zheng Zhang.

References

[1] G. Agha, *Actors: A Model of Concurrent Computation in Distributed Systems*, MIT Press, 1986.

[2] G. Agha, S. Frolund, R. Panwar and D. Sturman, A linguistic framework for dynamic composition of dependability protocols. In Dependable Computing for Critical Applications III, pages 345–363. International Federation of Information Processing Societies (IFIP), Elsevier Science Publisher, 1993.

[3] G. Agha, I.A. Mason, S.F. Smith and C.L. Talcott, A foundation for actor computation, *Journal of Functional Programming* **7** (1997), 1–72.

[4] G. Agha and C. Varela, Worldwide computing middleware, in: *Practical Handbook on Internet Computing*, M. Singh, ed., CRC Press, 2004, pp. 38.1–21.

[5] W.E. Allcock, J. Bester, J. Bresnahan, A.L. Chervenak, I.T. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel and S. Tuecke, Data management and transfer in high-performance computational grid environments, *Parallel Computing* **28**(5) (2002), 749–771.

[6] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel and J. Shalf, The Cactus Worm: Experiments with dynamic resource selection and allocation in a grid environment, *International Journal of High-Performance Computing Applications* **15**(4) (2001), 345–358.

[7] G. Allen, T. Dramlitsch, I. Foster, N. Karonis, M. Ripeanu, E. Seidel, and B. Toonen, Supporting efficient execution in heterogeneous distributed computing environments with Cactus and Globus. In Supercomputing 2001 (SC 2001), Denver, November 2001.

[8] T.E. Anderson, D.E. Culler and D.A. Patterson, A Case for Networks of Workstations: NOW, *IEEE Micro* **15**(1) (February 1995), 54–64.

[9] D. Angulo, R. Aydt, F. Berman, A. Chien, K. Cooper, H. Dail, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, M. Mazina, J. Mellor-Crummey, D. Reed, O. Sievert, L. Torczon, S. Vadhiyar and R. Wolski, *Toward a framework for preparing and executing adaptive grid programs*, in International Parallel and Distributed Processing Symposium (IPDPS 2002), 2002.

[10] D. Angulo, I. Foster, C. Liu and L. Yang, *Design and evaluation of a resource selection framework for grid applications*, in IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, July 2002.

[11] F. Arbab and C. Talcott, eds, Fifth International Conference on Coordination Languages and Models (COORDINATION '2002), number 2315 in LNCS, Berlin, Springer-Verlag, 2002.

[12] J. Armstrong, *The development of Erlang*, in Proceedings of the ACM SIGPLAN International Conference on Functional Programming, ACM Press, 1997, 196–203.

[13] S. Barnard, R. Biswas, S. Saini, R. Van der Wijngaart, M. Yarrow, L. Zechter, I. Foster and O. Larsson, *Large-scale distributed computational uid dynamics on the Information Power Grid using Globus*, in The Seventh Symposium on the Frontiers of Massively Parallel Computation, 1999, 60–71.

[14] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon and R. Wolski, The GrADS project: Software support for high-level grid application development, *International Journal of High-Performance Computing Applications* **15**(4) (2002), 327–344.

[15] R.D. Blumofe and P.A. Lisiecki, *Adaptive and reliable parallel computing on networks of workstations*, in Proceedings of USENIX '97 Conference, 1997.

[16] G. Bosilca, A. Bouteiller, F. Cappello, S. Djilali, G. Fdak, C. Germain, T. Hrault, P. Lemarinier, O. Lodygensky, F. Magniette, V. Nri and A. Selikhov, *MPICH-V: Toward a scalable fault tolerant MPI for volatile nodes*, in Supercomputing 2002, Baltimore, USA, November 2002, 1–18.

[17] A. Bouteiller, F. Cappello, T. Hrault, G. Krawezik, P. Lemarinier and F. Magniette, *MPICH-V2: A fault tolerant MPI for volatile nodes based on the pessimistic sender based message logging*, in Supercomputing 2003, Phoenix, USA, November 2003.

[18] A. Bouteiller, P. Lemarinier, G. Krawezik and F. Cappello, *Coordinated checkpoint versus message log for fault tolerant MPI*, in Cluster 2003, Hong Kong, China, December 2003.

[19] P. Buonadonna, A. Geweke and D.E. Culler, *An implementation and analysis of the Virtual Interface Architecture*, in Proceedings of Supercomputing '98, Orlando, FL, November 1998.

[20] C. Callsen and G. Agha, Open Heterogeneous Computing in ActorSpace, *Journal of Parallel and Distributed Computing* (1994), 289–300.

[21] L. Cardelli and A.D. Gordon, *Mobile Ambients*, in Foundations of System Specification and Computational Structures, LNCS 1378, Springer Verlag, 1998, 140–155.

[22] N. Carriero and D. Gelernter, *How to Write Parallel Programs*, MIT Press, 1990.

[23] K.M. Chandy, A. Rifkin, P.A.G. Sivilotti, J. Mandelson, M. Richardson, W. Tanaka and L. Weisman, *A World-Wide Distributed System Using Java and the Internet*, in Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing, New York, USA, Aug. 1996.

[24] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger and B. Tierney, *Giggle: A framework for constructing scalable replica location services*, in Supercomputing 2002 (SC2002), November 2002.

[25] A.L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman and R. Schwartzkopf, *Performance and scalability of a replica location service*, in International IEEE Symposium on High Performance Distributed Computing, June 2004.

[26] P. Ciancarini and C. Hankin, eds, First International Conference on Coordination Languages and Models (COORDINATION '96), number 1061 in LNCS, Berlin, Springer-Verlag, 1996.

[27] P. Ciancarini and A. Wolf, eds, Third International Conference on Coordination Languages and Models (COORDINATION '99), number 1594 in LNCS, Berlin, Springer-Verlag, April 1999.

[28] K. Czajkowski, S. Fitzgerald, I. Foster and C. Kesselman, *Grid information services for distributed resource sharing*, in 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), August 2001.

[29] H.L. de Cougny, K.D. Devine, J.E. Flaherty, R.M. Loy, C. Özturan and M.S. Shephard, Load balancing for the parallel adaptive solution of partial differential equations, *Appl. Numer. Math.* **16** (1994), 157–182.

[30] T. Desell, K. El Maghraoui and C. Varela, *Load balancing of autonomous actors over dynamic networks*, in Proceedings of the Hawaii International Conference on System Sciences, HICSS-37 Software Technology Track, January 2004, 1–10.

[31] J. Ferber and J. Briot, *Design of a concurrent language for distributed artificial intelligence*, in Proceedings of the International Conference on Fifth Generation Computer Systems, volume 2, Institute for New Generation Computer Technology, 1988, 755–762.

[32] J. Field and C. Varela, Toward a programming model for building reliable systems with distributed state, *Electronic Notes in Theoretical Computer Science* **68**(3) (March 2003), 1–19. http://www.elsevier.nl/locate/entcs/volume68.html.

[33] J. Field and C. Varela, *Transactors: A programming model for maintaining globally consistent distributed state in unreliable environments*, in Conference on Principles of Programming Languages (POPL 2005), 2005, 195–208.

[34] J.E. Flaherty, R.M. Loy, C. Özturan, M.S. Shephard, B.K. Szymanski, J.D. Teresco and L.H. Ziantz, Parallel structures and dynamic load balancing for adaptive finite element computation, *Appl. Numer. Math.* **26** (1998), 241–263.

[35] J.E. Flaherty, R.M. Loy, M.S. Shephard, B.K. Szymanski, J.D. Teresco and L.H. Ziantz, Adaptive local refinement with octree load-balancing for the parallel solution of three-dimensional conservation laws, *J. Parallel Distrib. Comput.* **47** (1997), 139–152.

[36] I. Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal and S. Tuecke, A wide-area implementation of the Message Passing Interface, *Parallel Computing* **24**(12) (1998), 1735–1749.

[37] I. Foster and C. Kesselman, The Globus Project: A Status Report, in: *Proceedings of the Seventh Heterogeneous Computing Workshop* (*HCW '98*), J. Antonio, ed., IEEE Computer Society, March 1998, pp. 4–18.

[38] I. Foster and C. Kesselman, *Globus: A toolkit-based grid architecture*, in The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 259–278.

[39] I. Foster, C. Kesselman, J. Nick and S. Tuecke, Grid services for distributed systems integration, *IEEE Computer* **35**(6) (June 2002), 37–46.

[40] I. Foster, C. Kesselman, J. Nick and S. Tuecke, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.

[41] I. Foster, C. Kesselman, G. Tsudik and S. Tuecke, *A security architecture for computational grids*, in Proc. 5th ACM Conference on Computer and Communications Security Conference, 1998, 83–92.

[42] C. Fournet and G. Gonthier, *The re exive CHAM and the join-calculus*, In Proceedings of the ACM Symposium on Principles of Programming Languages, ACM Press, Paris, January 1996.

[43] J. Frey, T. Tannenbaum, I. Foster, M. Livny and S. Tuecke, Condor-G: A computation management agent for multi-institutional grids, *Cluster Computing* **5**(3) (2002), 237–246.

[44] S. Frolund, *Coordinating Distributed Objects: An Actor-Based Approach to Synchronization*, MIT Press, 1996.

[45] D. Gannon, R. Bramley, G. Fox, S. Smallen, A. Rossi, R. Ananthakrishnan, F. Bertrand, K. Chiu, M. Farrellee, M. Govindaraju, S. Krishnan, L. Ramakrishnan, Y. Simmhan, A. Slominski, Y. Ma, C. Olariu and N. Rey-Cenevaz, Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications, *Journal of Cluster Computing* (2002).

[46] D. Garlan and D. Le Metayer, eds, *Second International Conference on Coordination Languages and Models* (*COORDINATION '97*), number 1282 in LNCS, Berlin, Springer-Verlag, 1997.

[47] A.S. Grimshaw, W.A. Wulf and "the Legion team", The legion vision of a worldwide virtual computer, *Communications of the ACM* **40**(1) (January 1997), 39–45.

[48] C. Hewitt, Viewing control structures as patterns of passing messages, *Journal of Artificial Intelligence* **8**(3) (June 1977), 323–364.

[49] A. Iamnitchi and I. Foster, *On fully decentralized resource discovery in grid environments*, in International Workshop on Grid Computing, Denver, Colorado, USA, November 2001.

[50] S. Jagannathan, Customization of first-class tuple spaces in a higher-order language, in: *PARLE '91, volume 2, number 506 in LNCS*, E.H.L. Arts, J. van Leeuwen and M. Rem, eds, Springer-Verlag, 1991.

[51] N. Karonis, B. de Supinski, I. Foster, W. Gropp, E. Lusk and J. Bresnahan, *Exploiting hierarchy in parallel computer networks to optimize collective operation performance*, in 14th International Parallel Distributed Processing Symposium (IPDPS'00), Cancun, Mexico, May 2000, 377–384,

[52] N. Karonis, B. Toonen and I. Foster, MPICH-G2: A grid-enabled implementation of the Message Passing Interface, *J. Parallel Distrib. Comput.* **63**(5) (2003), 551–563.

[53] T. Kielmann, Designing a coordination model for open systems. In Ciancarini and Hankin [26], 267–284.

[54] W. Kim, *THAL: An Actor System for Efficient and Scalable Concurrent Computing*, PhD thesis, University of Illinois at Urbana-Champaign, May 1997.

[55] F. Kon, R.H. Campbell, M.D. Mickunas, K. Nahrstedt and F.J. Ballesteros, *2K: A Distributed Operating System for Dynamic Heterogeneous Environments*, in Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing (HPDC'9), Pittsburgh, August 2000, 201–208.

[56] H. Lamehamedi, Z. Shentu, B.K. Szymanski and E. Deelman, *Simulation of dynamic data replication strategies in data grids*, in Proceedings of the 12th Heterogeneous Computing Workshop, 2003.

[57] H. Lamehamedi, B.K. Szymanski and E. Deelman, *Data replication strategies in grid environments*, in Proceedings of the Int. Conference on Algorithms and Architectures for Parallel Processing, 2002, 378–383.

[58] M. Litzkow, M. Livny and M. Mutka, *Condor – a hunter of idle workstations*, in Proceedings of the 8th International Conference of Distributed Computing Systems, June 1988, 104–111.

[59] K. El Maghraoui, J. Flaherty, B. Szymanski, J. Teresco and C. Varela, *Adaptive computation over dynamic and heterogeneous networks*, in Proc. of the Fifth International Conference on Parallel Processing and Applied Mathematics (PPAM'2003), number 3019 in LNCS, Czestochowa, Poland, September 2003.

[60] K. El Maghraoui, T. Desell, B.K. Szymanski, J.D. Teresco and C.A. Varela, Towards a middleware framework for dynamically reconfigurable scientific computing, in: *Grid Computing and New Frontiers of High Performance Processing*, L. Grandinetti, ed., Elsevier, 2005.

[61] S. Matsuoka and S. Kawai, *Using tuple space communication in distributed object-oriented languages*, in ACM Conference Proceedings, Object Oriented Programming Languages, Systems and Applications, San Diego, CA, 1988, 276–284.

[62] S. Matsuoka, T. Watanabe and A. Yonezawa, *Hybrid group re ective architecture for object-oriented concurrent re ective programming*, in Proceedings of the European Conference on Object-Oriented Programming, number 512 in LNCS, 1991, 231–250.

[63] R. Milner, J. Parrow and D. Walker, A calculus of mobile processes, parts I–II, *Information and Computation* **100**(1) (1992), 1–77.

[64] M. Nibhanapudi and B.K. Szymanski, *High Performance Cluster Computing, volume I of Architectures and Systems*, chapter BSP-based Adaptive Parallel Processing, Prentice Hall, New York, 1999, 702–721.

[65] R. De Nicola and G. Meredith, eds, *Sixth International Conference on Coordination Languages and Models* (*COORDINATION '2004*), number 2949 in LNCS, Berlin, Springer-Verlag, 2004.

[66] V. Pande et al., Atomistic protein folding simulations on the submillisecond timescale using worldwide distributed computing. *Biopolymers*, 2002. Peter Kollman Memorial Issue.

[67] B.C. Pierce and D.N. Turner, Pict: A Programming Language Based on the Pi-Calculus, in: *Proof, Language and Interaction: Essays in Honour of Robin Milner*, G. Plotkin, C. Stirling and M. Tofte, eds, MIT Press, 2000, pp. 455–494.

[68] K. Ranganathan and I. Foster, *Identifying dynamic replication strategies for high performance data grids*, in International Workshop on Grid Computing, Denver, Colorado, USA, November 2002.

[69] J.-F. Remacle, J.E. Flaherty and M.S. Shephard, An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible ow problems, *SIAM Review* **45**(1) (2003), 53–72.

[70] S. Ren, G.A. Agha and M. Saito, A modular approach for programming distributed real-time systems, *Journal of Parallel and Distributed Computing* **36** (1996), 4–12.

[71] M. Ripeanu and I. Foster, *A decentralized, adaptive, replica location service*, in 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, Scotland, July 2002.

[72] M.S. Shephard, J.E. Flaherty, C.L. Bottasso, H.L. de Cougny, C. Özturan and M.L. Simone, Parallel automatic adaptive analysis, *Parallel Comput* **23** (1997), 1327–1347.

[73] O. Sievert and H. Casanova, A simple MPI process swapping architecture for iterative applications, *International Journal of High Performance Computing Applications* (2003).

[74] P. Stelling, I. Foster, C. Kesselman, C. Lee and G. von Laszewski, *A Fault Detection Service for Wide Area Distributed Computations*, in Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing, Chicago, IL, 28-31 July 1998, 268–278.

[75] H. Stockinger, A. Samar, B. Allcock, I. Foster, K. Holtman and B. Tierney, File and object replication in data grids, *Journal of Cluster Computing* **5**(3) (2002), 305–314.

[76] D.C. Sturman and G. Agha, *A protocol description language for customizing failure semantics*, in Proceedings of the 13th Symposium on Reliable Distributed Systems, IEEE Computer Society Press, October 1994.

[77] W.T. Sullivan, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye and D. Anderson, *A New Major SETI Project based on project SERENDIP data and 100,000 Personal Computers*, in Proceedings of the Fifth International Conference on Bioastronomy, Editrice Compositori, Bologna, Italy, 1997.

[78] B. Szymanski, C. Varela, J. Cummings and J. Napolitano, *Dynamically reconfigurable scientific computing on large-scale heterogeneous grids*, in Proc. of the Fifth International Conference on Parallel Processing and Applied Mathematics (PPAM'2003), LNCS 3019, Czestochowa, Poland, September 2003.

[79] K. Taura, K. Kaneda and T. Endo, *Phoenix: a Parallel Programming Model for Accommodating Dynamically Joininig/Leaving Re-sources*, in Proc. of PPoPP, ACM, 2003, 216–229.

[80] C. Tomlinson, P. Cannata, G. Meredith and D. Woelk, The extensible services switch in Carnot, *IEEE Parallel and Distributed Technology* **1**(2) (May 1993), 16–20.

[81] S.S. Vadhiyar and J.J. Dongarra, *A performance oriented migration framework for the grid*, In CCGrid, IEEE Computing Clusters and the Grid, Tokyo, Japan, may 2003.

[82] S.S. Vadhiyar and J.J. Dongarra, SRS – a framework for developing malleable and migratable parallel applications for distributed systems, *International Journal of High Performance Applications 2003 and Super-computing 2003* (2003).

[83] A. Vahdat, T. Anderson, M. Dahlin, D. Culler, E. Belani, P. Eastham and C. Yoshikawa, *WebOS: Operating System Services For Wide Area Applications*, in Proceedings of the Seventh IEEE Symposium on High Performance Distributed Computing, July 1998.

[84] C. Varela and G. Agha, Programming dynamically reconfigurable open systems with SALSA. ACM SIGPLAN Notices, *OOPSLA'2001 Intriguing Technology Track Proceedings* **36**(12) (December 2001), 20–34. http://www.cs.rpi.edu/?cvarela/oopsla2001.pdf.

[85] S. Vazhkudai, S. Tuecke and I. Foster, *Replica selection in the Globus Data Grid*, in First IEE/ACM International Conference on Cluster Computing and the Grid (CCGrid 2001), May 2001, 106–113.

[86] T. Watanabe and A. Yonezawa, An actor-based meta-level architecture for group-wide re ection, in: *Foundations of Object-Oriented Languages*, number 489 in LNCS, J.W. deBakker, W.P. deRoever and G. Rozenberg, eds, Springer-Verlag, 1990, pp. 405–425.

[87] P. Wojciechowski and P. Sewell, *Nomadic Pict: Language and Infrastructure Design for Mobile Agents*, in First International Symposium on Agent Systems and Applications (ASA'99)/Third International Symposium on Mobile Agents (MA'99), Palm Springs, CA, USA, 1999.

[88] R. Wolski, J. Plank, J. Brevik and T. Bryan, *G-commerce: Market formulations controlling resource allocation on the computational grid*, in International Parallel and Distributed Processing Symposium, March 2001.

[89] R. Wolski, S. Plank, T. Bryan and J. Brevik, Analyzing market-based resource allocation strategies for the computational grid, *International Journal of High Performance Computing Applications* **15**(3) (2001), 258–281.

[90] R. Wolski, N.T. Spring and J. Hayes, The Network Weather Service: A distributed resource performance forecasting service for metacomputing, *Future Generation Comput. Syst.* **15**(5–6) (October 1999), 757–768.

[91] G. Wrzesińska, R.V. van Nieuwpoort, J. Maassen and H. Bal, *Fault-tolerant, malleability, and migration for divide-and-conquer applications on the grid*, in Proceedings of IPDPS 2005, 2005.

[92] L. Yang, J.M. Schopf and I. Foster, *Conservative scheduling: Using predicted variance to improve scheduling decisions in dynamic environments*, In Supercomputing 2003 (SC2003), November 2003.

[93] A. Yonezawa, ed., *ABCL An Object-Oriented Concurrent System*, MIT Press, Cambridge, Mass., 1990.