Proc. e-Science 2006, Second IEEE International Conference on e-Science and Grid Computing,
P.M.A. Sloot, G.D. van Albada, M. Bubak, and A. Trefethen (edts), Amsterdam, Netherlands, December, 2006.

A Middleware Framework for Maximum Likelihood Evaluation over Dynamic Grids

Wei-Jen Wang† Kaoutar El Maghraoui† John Cummings‡ Jim Napolitano‡
Boleslaw K. Szymanski† Carlos A. Varela†

†Department of Computer Science ‡Department of Physics, Applied Physics, and Astronomy
Rensselaer Polytechnic Institute, Troy, NY, U.S.A.
{wangw5, elmagk, cummij, napolj, szymab, varelc}@rpi.edu

Abstract

We have designed a maximum likelihood fitter using the actor model to distribute the computation over a heterogeneous network. The prototype implementation uses the SALSA programming language and the Internet Operating System middleware. We have used our fitter to perform a partial wave analysis of particle physics data. Preliminary measurements have shown good performance and scalability. We expect our approach to be applicable to other scientific domains, such as biology and astronomy, where maximum likelihood evaluation is an important technique. We also expect our performance results to scale to Internet-wide runtime infrastructures, given the high adaptability of our software framework.

1. Introduction

While computational power, network bandwidth, and storage capacity are growing exponentially, the same technological progress responsible for that growth has improved data acquisition rates in some fields to even larger degrees. This has led to an embarrassment of riches: we often have more data than we can analyze using traditional techniques. One strategy to address this problem is to make use of idle computational resources scattered throughout the Internet. The main difficulties with this strategy arise from the Internet's heterogeneous and dynamic nature.

The above concerns are simultaneously addressed by using SALSA, an actor oriented programming language [35]. SALSA actors execute on a Java Virtual Machine, essentially turning a heterogeneous unsafe network of physical machines into a homogeneous secure network of virtual machines. An actor encapsulates its state and a thread of execution, and thus may migrate easily, allowing for dynamic reconfiguration of the distributed application as the conditions of the network change. This can be done either by the application itself, or transparently by the middleware when using the Internet Operating System (IOS) [27].

Particle physics is facing the described above embarrassment of riches in the field of spectroscopy. Partial wave analysis (PWA) of the very large data sets soon expected to be available seems intractable by current methods. The maximum likelihood fitting technique used in PWA is amenable to the strategy sketched above, and with suitable generalizations, is applicable to many scientific fields.

The remainder of the paper is structured as follows: Section 2 discusses partial wave analysis, its significance in particle physics, and maximum likelihood evaluation as a computational technique. Section 3 describes the actor implementation of maximum likelihood evaluation. Section 4 introduces the IOS middleware components and its ability to adapt distributed executions to dynamic run-time environments. Section 5 presents performance and scalability results of our prototype. Section 6 discusses the potential generalization of the maximum-likelihood evaluation technique to other scientific domains. Section 7 discusses related work. Finally, Section 8 concludes the paper.

2. Partial Wave Analysis

Although fundamental physics has not yet been able to unify the forces of nature into one encompassing theory, sometimes called a Grand Unified Theory, as we enter the 21st century, we are making progress. Electricity and magnetism were united in the late 1800's

and came to be understood as a quantum gauge field theory in the 1940's. In the 1960's attempts to understand the weak nuclear force as a quantum gauge field theory led to the discovery that quantum electrodynamics (QED) was the result of spontaneous symmetry breaking in a unified electro-weak field theory, thus uniting electromagnetism and the weak force.

The great success of quantum gauge field theory in describing electro-weak interactions led to the development of quantum chromodynamics (QCD) in the 1970's. This theory describes the strong force which is felt between objects with "color"—the strong force analog of electric charge—such as quarks. QCD has proven more difficult to understand than QED, however. The technique of perturbation theory, used successfully in QED, works only at high energies in QCD. However, the world we inhabit typically is not at high energies, in a perturbative QCD sense. Tests of QCD in the non-perturbative region are less quantitative, due to the lack of accurate analytic predictions. What we have for this region is a model which predates QCD: the Quark Model that is a classification system for particles made up of quarks (and anti-quarks). The particles properties are determined by the properties of the constituent quarks and the quantum state they are in. Surprising agreement with the observed states is found; despite ignoring all the complications of a non-abelian gauge field theory, the Quark Model describes the observed states well!

One complication ignored is that the force carriers of QCD—gluons—carry color, unlike the neutral photon of QED. Gluons feel the strong interaction, so bound states should exist with gluons as constituents. Entire families of states beyond the quark model seem to be predicted by QCD. Understanding why (and if) the quark model works so well will help us understand the properties of QCD, and extend its tested domain down from the perturbative region.

Experimentally, we need to observe particle states and measure their quantum numbers known as "spin" and "parity". One important technique used to determine the spin-parity of particles is Partial Wave Analysis (PWA). Consider an example: a beam of mesons, say pions, is produced in an accelerator and hits a liquid hydrogen target. Some fraction of the pions interact with a proton in the target, and if the energy of the pion is high enough, a spray of particles is produced. Some of the particles will live long enough to create trails in a suitable detector, but many will decay after an extremely short time, $\sim 10^{-23}$s and will not travel a measurable distance. It is these short lived particles that are of interest. Their existence and properties can be inferred from correlations in the final state particles into which they decay.

There are many ways of reaching this final system, through various allowed intermediate states. Quantum mechanics dictates that all of them need to be considered. Each possibility is described by a complex number known as the quantum mechanical "amplitude" which must be added coherently before squaring the total amplitude to get the probability distributions. Every intermediate state with an assumed spin-parity has its own angular distributions of the final state particles. By varying the amount of each intermediate state to fit the observed distributions, PWA can determine the amount of each spin-parity state present in the data set.

These fits are done using the maximum likelihood method. The likelihood is defined as the product of the probabilities of observing each event given a set of fit parameters. In our case the fit parameters are the complex amplitudes for producing each of the intermediate states. In practice, we minimize the negative of the logarithm of the likelihood, or $-\ln(\mathcal{L})$, which can be written

$$-\ln(\mathcal{L}) = -\sum_{i}^{n} \ln\left(\left|\psi_\alpha^p \psi_\alpha^d(\tau_i)\right|^2\right) - n\psi_\alpha^p \Psi_{\alpha\alpha'} \psi_{\alpha'}^{p*} \quad (1)$$

where the sum over i runs over all events in the data set, and the sums over the repeated αs, the fit parameter index, are implicit. The ψ_α^p are the complex fit parameters, related to the amount of the intermediate state α produced. The $\psi_\alpha^d(\tau_i)$ is the quantum amplitude for the i^{th} event with angles τ_i assuming intermediate state α. The possibility of non-interfering amplitudes has been ignored here. While physically important, it is a detail which further complicates the expression for the likelihood, yet serves no illustrative purpose for the discussion at hand. The second term on the right hand side is the normalization integral, where any known inefficiencies of the detector are taken into account. The total number of events in the data being fit is n; and $\Psi_{\alpha\alpha'}$ is the result of the normalization integral, done numerically before the fit is performed.

Finding the best fit to a typical data set might require hundreds of trials: experimenting with various parameter sets, trying slightly different selections of the original data, repeating fits with various parameter starting values, etc. Each trial may require thousands of evaluations of Eq. 1. For this type of exploratory analysis, it is desirable to keep the time per fit as short as possible. Today's data sets typically are $\approx 10^5$ events in size, and usually require 10-100 parameters. These fits, using our current C++ (serial) code, take hours to complete. This is tolerable today, but the next generation detectors [8, 26] will record data sets

of $10^6 - 10^7$ events. Eq. 1 is dominated by the sum over events, meaning these larger data sets will require weeks *per fit*!

Fortunately, the same dominance of the event sum helps us distribute the work. The sum is split into more manageable pieces and each piece is evaluated on a separate processor. By using the dynamic reconfigurability of SALSA and the load balancing of IOS we can achieve near optimal performance on a heterogeneous cluster of nodes.

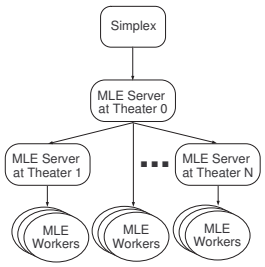

Figure 1. Simplified representation of the fitter.

3 A Maximum Likelihood Fitter

The physics data are currently fit using a fitter written in C++ [13] and based on the MINUIT functional minimization library from CERN [10]. The code makes no attempt at concurrency; the likelihood is evaluated serially on a single processor. The evaluation of the likelihood is done by a virtual stack machine inspired by `hoc` [25].

To experiment with an actor based fitting scheme, we have implemented a simple fitting program in SALSA. SALSA (Simple Actor Language, System, and Architecture) [35] is an actor programming language with high-level constructs for remote messaging, universal naming, migration, and coordination. An actor [1] is a unit of encapsulation for both a state (procedures and data) as well as processing of such a state (a thread of control). All communication between actors is through asynchronous message passing. While processing a message, an actor can carry out any of four basic operations: (1) alter its state, (2) create new actors, (3) send messages to peer actors, or (4) migrate to another run-time environment. Actors are therefore inherently independent, concurrent, and autonomous, which enables efficiency in parallel execution and facilitates mobility [2, 16]. SALSA programs are compiled into Java code, leveraging the existence of virtual machine implementations in multiple heterogeneous platforms and operating systems. We therefore view a heterogeneous network of physical machines as a homogeneous network of Java virtual machines. While Java's main perceived drawback is its lack of performance— due to its bytecode interpretation overhead— advances in just-in-time (JIT) and adaptive compilation, make Java a very attractive platform for scientific applications [7].

The World-Wide Computer (WWC) [2] run-time architecture consists of naming servers and virtual machines running as Java applications on different Internet nodes. The virtual machines, called *theaters*, provide an environment for execution of universal actors using local resources. High-level programming language abstractions enable actors to create remote communication links with peer actors running on other WWC theaters. Furthermore, actors can easily migrate with their full state to other WWC theaters as they become available, supporting load balancing and scalability. The naming servers keep track of universal actor locators, so that communication remains transparent to actor location and migration.

We know from our C++ code that the likelihood evaluation is the most expensive part of a fit, and this expense grows linearly with the size of the data set being fit. The large data sets expected to be obtained shortly will become prohibitively expensive to process. For typical numbers of fit parameters, the likelihood (Eq. 1) is dominated by the sum over events. Our distributed fitter splits the sum into smaller pieces and uses worker actors on distributed theaters to evaluate each piece.

While the C++ code uses more sophisticated minimization routines available in MINUIT, we chose for our tests to write a minimizer using the simplex algorithm [30] that is easy to understand and implement. For our early experiments with fitting using SALSA and IOS, a conceptually simple minimization method allowed us to concentrate on the evaluation of the likelihood function. For an n parameter fit, the simplex is a $n+1$ vertex polygon resting on the likelihood surface. The function to be minimized is evaluated at each vertex, and the "worst" vertex moved to a better position. By repeating this procedure, the simplex can be made to "tumble" down the function surface to the minimum.

The simplified architecture of the prototype fitter is shown in Fig. 1. The `simplex` actor is associated with the main `MLEserver` and instantiates several peer `MLEservers` at different WWC theaters. Each of the `MLEservers` processes some part of the input data set. The size of data to process depends on the number of theaters joining the computations and configuration parameters. A `MLEserver` may have n `MLEworkers` to

evaluate partial sums of the input data it has. The value of n may be set at runtime and parameterizes the degree of parallelism in a single computing host. The main `MLEserver` merges computing results returned by peer servers and itself. It also provides the `simplex` actor with access to the data files to be fit and any additional input such as the results of the normalization integrals in Eq. 1.

4 Middleware-Driven Parallel Execution over Dynamic Grids

The dynamicity of grid environments—new nodes joining, old nodes leaving, node loads fluctuating, resources changing availability—makes it necessary to adapt applications based on information obtained during their distributed execution to improve their performance. Maximum likelihood computations can also significantly change the computation required at different time steps due to different input data sets or different convergence properties of the minimization routine in use. The load imbalance inherent in the data and algorithm, along with the dynamicity of the grid, make it a requirement to adapt distributed computations *dynamically* since these imbalances are not possible to predict at compile-time.

While several application-level frameworks have been created to dynamically adjust computational load (e.g., by redistributing a mesh of data values to different processors when a significant performance degradation is detected) [33, 14, 20, 19, 6], this strategy imposes a significant burden on scientific code developers and is better tackled at a *middleware* level. Middleware is a software layer between applications and their distributed execution environments dealing with non-functional concerns, e.g., resource management, fault-tolerance, security, and quality of service (see e.g., [3, 36]).

Middleware can be organized as a virtual network of agents which gather and exchange information on the physical resources dynamically and in a completely decentralized manner. The application components can also be profiled to detect their communication topology and make better placement, relocation, granularity refinement, or replication decisions.

We have developed a modular middleware framework, called the Internet Operating System (IOS) [27], to trigger application reconfiguration decisions based on a generic weighted resource management model [16]. The IOS architecture (see Fig. 2) consists of a peer-to-peer network of middleware agents, each of which is composed of:

- **a profiling module**: that gathers information at two levels: (*i*) at an *application level* to determine the communication topology of application entities as well as their resource usage over time, and (*ii*) at a *physical level* to determine local resource availability;
- **a protocol module**: that enables communication of middleware agents to exchange local and remotely profiled information, as well as work stealing requests and forwarding; and
- **a decision module**: that uses the information given by the profiling and protocol modules to trigger application reconfiguration decisions at runtime.

While the IOS middleware is programming language and programming model neutral, we already have interfaced it to MPI and SALSA programs. Therefore, the maximum likelihood fitter described in Section 3 can make use of IOS' dynamic reconfiguration algorithms directly.

5. Experimental Results

We have initially made experiments using a static load-balancing approach which distributes computing tasks in optimal conditions. Three large data sets were fit — $9053 * 2^6$ events as `MIX1`, $9053 * 2^7$ events as `MIX2`, and $9053 * 2^8$ events as `MIX3`, where each event is represented by seven complex numbers. Typically, the maximum likelihood application requires about 2000 function calls for an acceptable solution. In case of `MIX1` for example, each function call in a uni-processor machine takes about one minute, indicating the total execution time is more than 33 hours.

Results from our experiments indicate, as we expected, that the maximum likelihood problem with large input data is scalable; see Fig. 3 and 4. *Parallel efficiency*, used in Fig. 3 and 4, is defined as $(T_{sequential}/(n * T_{parallel}))$, where $T_{sequential}$ is the sequential execution time per function call, n is the number of processors, and $T_{parallel}$ is the execution time per function call with n processors. We use two different clusters to run the tests — a small and heterogeneous cluster, *Cluster one* [1], and another relatively large and homogeneous cluster, *Cluster two* [2]. In the scalability experiment, each processor is assigned an actor to

[1] Cluster one contains four IBM Workstations (six processors) and four SUN Blades (eight processors), where an IBM Workstation processor is twice as fast as a SUN Blade processor.

[2] Cluster two consists of three 4-dual-core Opteron machines and fourteen 4-single-core Opteron machines.

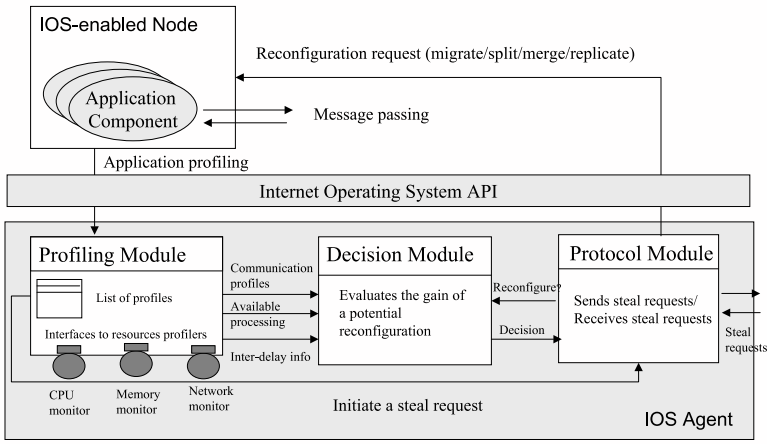

Figure 2. Internet Operating System(IOS) Architecture.

perform computing tasks, and each actor has a cache size of 10000 events. Fig. 4 shows that the execution time per function call decreases roughly by half as the total number of processors doubles, and the parallel efficiency decreses faster for smaller data input. Fig. 3 also shows scalability but reveals a difference in the fact that there exist two different slopes for each curve. The relatively weak computing power of Processors 7 to 14 is the main cause of the drastic decrease of the parallel efficiency.

Cache size is one possible factor that affects performance. By changing the cache size, we found that increasing the cache size uniformly in Cluster two (homogeneous) can improve performance for more than 25%. Surprisingly, increasing the cache size in Cluster one (heterogeneous) has trivial impact or even negative impact (2% slower) on the performance. We attribute this to the fact that a uniform cache size does not work well in a more heterogeneous environment and thus precludes smart memory use by the operating systems.

These preliminary experimental results are very promising and demonstrate that maximum likelihood evaluations can be effectively parallelized to execute over distributed and heterogeneous environments. While we have not explicitly tested the *dynamicity* of the run-time environments using this particular maximum likelihood fitter, we have shown using a similar iterative application in astronomy [17] that IOS is capable of adjusting to dynamic environments to improve performance. In Fig. 5, every 6 iterations, the environment was artificially changed, from 8 to 12 to 16 to 15 to 10 back to 8 processors. The last 8 processors removed were the initial 8 processors. A non-reconfigurable application would not scale beyond 8 processors, nor be able to move to the new processors when the initial ones were removed. The performance of the application increased by 22% on average with the availability of new resources.

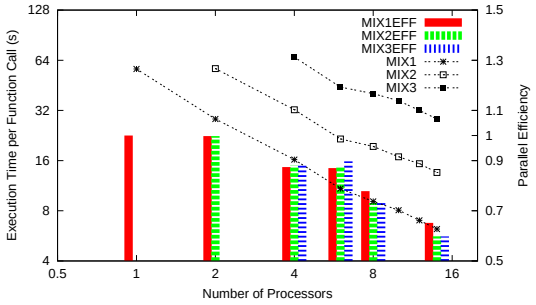

Figure 3. Scalability experiment results for different input sizes at Cluster one (heterogeneous).

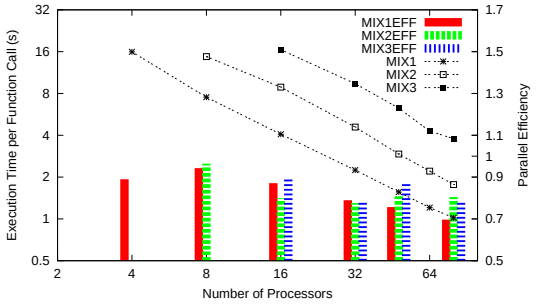

Figure 4. Scalability experiment results for different input sizes at Cluster two (homogeneous).

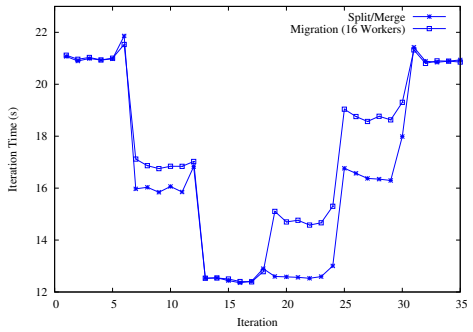

Figure 5. Autonomous reconfiguration using migration and malleability.

6. Toward a Generic MLE Framework

The importance of the method of maximum likelihood in the sciences cannot be overstated. "From a theoretical point of view, the most important general method of estimation known so far is the *method of maximum likelihood*" [12]. While the method has been used for decades to fit small data sets, modern techniques have allowed the collection of data volumes that exceed the capabilities of current fitting techniques. While our prototype fitter has been designed and implemented with a particular problem—Partial Wave Analysis—in mind, we are aware that the techniques we are developing are useful to a broader population of researchers. The lessons we are learning will allow us to abstract out the appropriate parts of the problems at hand and produce a more general application to assist studies in diverse fields of inquiry. Following are brief descriptions of ongoing collaborations that might illustrate the potential impact of a generic MLE framework.

Biology Application: The origins of bacterial pathogenicity. Intimate cooperative associations between bacterial symbionts and their hosts have given rise to a variety of adaptations. In each of such symbioses, the microbial partner is specific to the host organism and may be a necessary part of the normal life cycle of the host, providing some critical function or capability. Comparisons between complete genomes of various parasitic and free-living states of symbiotic bacteria may help uncover mechanisms of infection and colonization, and may shed light on the evolution of virulence and pathogenesis [28]. To address these questions effectively, complete genomic sequences of bacteria that have multiple life history stages must be analyzed to identify patterns in the evolution of genome composition and organization. Many of the problems in phylogenetic analysis are known to be NP-complete [22], yet for smaller genomes they are tractable with effective heuristic algorithms, some of which are based on tree re-arrangement methods [15]. The quality of these trees (optimality value) could be measured by a maximum likelihood model-based approach (in reality the negative logarithm of the likelihood is minimized), giving rise to the computational problem of exactly the same type as discussed in the physics application. For analyses based on single molecular loci, given pre-specified alignments, these techniques can analyze effectively thousands of taxa [24]. The automatic load balancing and fault tolerance capabilities of the software platform supported by our system are essential in such application, as the genomic rearrangement problems require unpredictable and unequal effort. Biologically, the benefits of analyzing such complex data sets will not only generate a more complete phylogenetic survey among taxa, but will also give us the ability to detect specific groups of genes (i.e., islands of pathogenicity) that contain unique genetic signatures that can be used to identify differences in virulent and benign bacteria.

Astronomy Application: The origins and structure of the Milky Way. The Sloan Digital Sky Survey (SDSS) has already accumulated 9000 square degrees of imaging data in five wavelength ranges by June 2005. Additional 4000 square degrees of imaging data and individual spectra for 240,000 stars in the Milky Way will be collected by 2008. This huge amount of data will enable evaluation of complex astronomical models but will also require tremendous amount of computational power. Many of the questions arising in the study of the Milky Way galaxy, such as the structures existing in this galaxy and their spatial distribution, require a probabilistic algorithm for locating geometric objects in spatial databases [32] that can be treated as a mixture density estimation problem. For a given cross-sectional observation region, the densities for the background and structure conditioned on being in this region can be normalized. After such normalization, the observed density of stars is drawn from a mixture distribution parameterized by (i) the fraction of structure stars in the data (compared to background), (ii) the parameters that govern the structure distribution which specify the position of the structure, as well as other important properties, and (iii) the parameters of the Milky Way halo. Such formulated, the problem can again be solved using a maximum likelihood approach. Specifically, the likelihood is the probability (density) of obtaining the observed star distribution after repeated independent sampling from the mixture

distribution. Such probabilistic framework gives a natural sampling algorithm for separating structure from background [31]. The scientific significance of being able to separate the structure from the background stars lies in the resulting ability to analyze and test various theories for the dynamics of the background and the structure separately.

7 Related Work

Over the last few years, several efforts have focused on building middleware-level software tools to enable resource sharing and problem solving on large grids within the scientific community. The Globus/OGSA project seeks to enable the construction of large *computational grids* [21] by providing a set of services that solve common grid deployment problems. Condor is a distributed resource management system that is designed to support high-throughput computing by harvesting idle resource cycles [34]. Virginia's Legion meta-system integrates research in object-oriented parallel processing, distributed computing, and security to form a programmable world-wide virtual computer [23]. 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 [11]. [36] presents more detailed discussion about middleware research over dynamic grid environments.

Other research efforts have focused on providing grid-enabling frameworks for typical scientific applications. One system that in particular targets numerical relativity is the Cactus system [4]. Cactus allows users to write application-specific modules and interface them to the Cactus framework which can run these modules on the grid. NetSolve [9] is a system that has been designed to solve computational science problems through a client-agent-server architecture. GridSolve [37] is an extension to NetSolve targeted specifically at grid environments. The Grid Physics Network (GriPhyN) project [5] uses grid technologies that provide virtual data management and workflow management to support physicists. The Particle Physics Data Grid [29] also employs grid technologies to support physics research. GridPP [18] is an ongoing collaboration of Particle Physicists and Computing Scientists from 19 UK universities. It aims at building grid-enabled interfaces and software tools to provide a problem-solving environment for particle physicists.

8. Conclusion

We have developed a prototype distributed maximum likelihood fitter and applied it to the particle physics problem of partial wave analysis. This fitter uses the SALSA programming language and the Internet Operating System middleware to achieve a dynamically reconfigurable system. By reconfiguring itself while running it is able to adapt to the changing conditions on a heterogeneous network such as the Internet. We have demonstrated excellent scalability properties of the program. Our plans for future work include the development of a generic framework for maximum likelihood fitting for diverse problem domains and studies of IOS-driven autonomous reconfiguration under different run-time environments.

Acknowledgements

We would like to express our gratitude to Chris Kona, Heschi Kreinick, and Zack Goldstein for implementing the initial versions of the SALSA maximum likelihood programs. We would also like to acknowledge the National Science Foundation (NSF CAREER Award No. CNS-0448407), IBM (SUR Awards 2003 and 2004) and RPI (Seed Funding Grant) for partial support for this research.

References

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

[2] G. Agha, N. Jamali, and C. Varela. Agent Naming and Coordination: Actor Based Models and Infrastructures. In A. Ominici, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents*, chapter 9, pages 225–248. Springer-Verlag, 2001. invited book chapter.

[3] G. Agha and C. Varela. Worldwide computing middleware. In M. Singh, editor, *Practical Handbook on Internet Computing*. CRC Press, 2004. invited book chapter.

[4] 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.

[5] P. Avery and I. Foster. The GriPhyN project: Toward petascale virtual data grids. Available: http://www.griphyn.org, 2001.

[6] C. L. Bottasso, J. E. Flaherty, C. Ozturan, M. S. Shephard, B. K. Szymanski, J. Teresco, and L. Ziantz. The quality of partitions by an iterative load balancer. In *Proc. 3rd Workshop on Languages, Compilers and Run-Time Systems for Scalable Computers*, 1996.

[7] J. M. Bull, L. A. Smith, L. Pottage, and R. Freeman. Benchmarking java against c and fortran for scientific applications. In *Proceedings of ACM Java Grande/ISCOPE Conference*, pages 97–105, 2001.

[8] D. S. Carman. Gluex: The search for gluonic excitations at jefferson laboratory. *AIP Conf. Proc.*, 814:173–182, 2006.

[9] H. Casanova and J. Dongarra. NetSolve: A network-enabled server for solving computational science problems. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(3):212–223, Fall 1997.

[10] CERN Program Library. Available: `http://cernlib.web.cern.ch/cernlib/`, 2006.

[11] 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, U.S.A., Aug 1996.

[12] H. A. Cramer. *Mathematical Methods of Statistics*. Princeton University Press, 1958.

[13] J. P. Cummings and D. P. Weygand. An object-oriented approach to partial wave analysis, 2003.

[14] 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.*, 16:157–182, 1994.

[15] R. Desalle, G. Giribet, and W. Wheeler. *Molecular Systematics and Evolution: Theory and Practice*. Birkhauser Verlag, 2002.

[16] T. Desell, K. E. 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*, pages 1–10, January 2004.

[17] T. Desell, K. E. Maghraoui, and C. Varela. Malleable Components for Scalable High Performance Computing . In *Proceedings of the HPDC'15 Workshop on HPC Grid programming Environments and Components (HPC-GECO/CompFrame)*, pages 37–44, Paris, France, June 2006. IEEE Computer Society.

[18] D.I. Britton et al. GridPP: From prototype to production. In *UK e-Science All Hands Conference*, Nottingham, UK, September 2006.

[19] 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.*, 26:241–263, 1998.

[20] 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-di mensional conservation laws. *J. Parallel Distrib. Comput.*, 47:139–152, 1997.

[21] I. Foster and C. Kesselman. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[22] L. Foulds and R. L. Graham. Steiner tree problem in phylogeny is np-complete. *Advances in Applied Mathematics*, (3):43–49, 1982.

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

[24] D. Janies and W. Wheeler. *Parallel efficiency of POY*. Birkhauser Verlag, 1997.

[25] B. W. Kernighan and R. Pike. *The UNIX Programming Environment*. Prentice Hall Professional Technical Reference, 1984.

[26] W. Li. Bepcii/besiii upgrade and the prospective physics, 2006.

[27] K. E. Maghraoui, T. J. Desell, B. K. Szymanski, and C. A. Varela. The Internet Operating System: Middleware for adaptive distributed computing. *International Journal of High Performance Computing Applications (IJHPCA), Special Issue on Scheduling Techniques for Large-Scale Distributed Platforms*, 2006. To appear.

[28] M. Nishiguci and B. Jones. *Microbial biodiversity within the Vibrionaceae*. Cole-Kluwer Academic Publishers, 2003.

[29] Particle Physics Data Grid (PPDG) project . Available: `http://www.ppdg.net`, 2004.

[30] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[31] J. Purnell, M. Magdon-Ismail, and H. Newberg. A probabilistic approach to finding geometric objects in spatial datasets of the milky way. In *Proc. 15th International Symposium on Methodoligies for Intelligent Systems (ISMIS 2005)*, Saratoga Springs, NY, May 2005.

[32] C. Reina, P. Bradley, and U. Fayyad. Clustering very large databases using mixture models. In *Proc. 15th International Conference on Pattern Recognition*, 2000.

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

[34] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[35] C. Varela and G. Agha. Programming dynamically reconfigurable open systems with SALSA. *ACM SIGPLAN Notices. OOPSLA'2001 Intriguing Technology Track Proceedings*, 36(12):20–34, Dec. 2001.

[36] C. A. Varela, P. Ciancarini, and K. Taura. Worldwide computing: Adaptive middleware and programming technology for dynamic Grid environments. *Scientific Programming Journal*, 13(4):255–263, December 2005. Guest Editorial.

[37] A. YarKhan, K. Seymour, K. Sagi, Z. Shi, and J. Dongarra. Recent developments in gridsolve. *International Journal of High Performance Computing Applications (IJHPCA)*, 20(1):131–141, 2006.