Laxmikant V. Kale
Research Accomplishments in Perspective

My research has centered upon parallel computing, and more specifically on improving (computer) performance and (human) productivity in parallel programming. My choice of specific research topics and strategies is guided by the following (admittedly subjective) axioms.

- To have a long-term impact on the state-of-art in parallel computing, the research must be Computer Science centered yet application oriented[95]. Centering one's research on a particular application will typically not lead to development of enabling technologies, applicable across a broad variety of applications. At the same time, without application orientation, one is in danger of developing tools and techniques that aren't useful to real applications. Using application benchmarks to demonstrate CS techniques isn't enough. So, one should develop core "enabling technologies" while focusing on full-fledged "real" applications, via intensely collaborative interdisciplinary research.

- As the systems area of computer science is an engineering discipline, it is necessary to embody the techniques developed as part of our research into reusable software, and to make such software available to other researchers/practitioners.

Consequently, my research has focused on developing the *middle layers* of enabling technologies, with portability and data driven execution as the underlying themes, with dynamic and irregular applications and environments as the context, and with performance, scalability, and ease of programming as the objectives.
An important aspect of this philosophy is that one must move constantly between developing new abstractions, and using them to effectively parallelizing challenging applications.
The following paragraphs summarize components of my past and ongoing research.

Portability: In 1986-87, when the *chare kernel* [165, 166, 167, 168] (predecessor to *Charm*) was being designed with the objective of providing portability, it was deemed an unrealistic objective, especially portability across shared and distributed memory machines. Such portability is now passe.

Data driven objects: Data-driven execution ideas were at the basis of data-flow machines proposed in the '70s. Software-based data driven execution was used in several research projects, including Rediflow (parallel functional languages), our own research [169] on parallel prolog and Agha's elaboration of Hewitt's Actor model. Chare kernel was one of the first pragmatic (e.g. C based) data-driven (active) object based system; [165, 166, 167, 168] it was available on most parallel machines of the day including workstation networks. [110] Other parallel object systems in that timeframe include ABCL, ORCA, PC++, and UC++. One of the first parallel implementations of Actors (HAL, Houck and Agha, ICPP '92) was developed on top of Charm. The performance and modularity benefits of data driven execution were systematically studied and demonstrated by us [170, 96, 92, 99]. We've continued to elaborate and develop the Charm++ parallel object model, which is at the foundation of much of our research. Beyond data driven objects themselves, Charm++ concepts include object groups, information sharing abstractions,[102] object-placement based load balancing [171] and prioritized scheduling[97]. A more recent addition is object arrays [88, 75] . Charm++ has been used extensively for developing applications by us and others.

Migrating objects and threads: To deal with unpredictable dynamic behavior of many applications, it is necessary to adaptively move an appropriate fraction of work away from overloaded processors. Charm++ objects provide a potential mechanism for this purpose. We've developed a framework that supports object migration, as well as thread migration.

Adaptive Run-time Systems and the Principle of Persistence:
A central theme to much of my group's recent research is that of adaptive runtime systems. This is based on a simple idea: let the programmer decompose the problem into parallel pieces, and let the runtime system assign (and dynamically reassign) those pieces to processor, schedule their execution, and mediate communication between them. I believe this represents a good division of labor between the "system" and the programmer. This empowers the RTS to carry out many "smart" optimizations. beleive
As we worked with CSE applications, and parallelized them using the mirgatable-objects model, a heuristic principle that we called the principle of persistence became apparent: the object communication patterns and computation loads tend to persist over time, in spite of dynamic nature of the applications. Based on this, we can build measurement-based dynamic load balancers [164, 75, 75, 87] and communication optimizers [52, 53, 53, 53] that can learn as the application runs. This has operned up a large area of research in adaptive runtime strategies. The load balancers enabled by the principle of persistence migrate existing objects, instead of migrating "seeds" for new objects, as in our load balancers for state-space-search [97, 131, 121]

Run-time systems and multi-paradigm interoperability: It is difficult to persuade broad developer community to switch to a new paradigm(such as Charm++). Further, different paradigms may be appropriate for different components of a single application. We developed the Converse runtime framework [90, 80] to address these issues. Converse provides interoperability between multiple data-driven and multithreaded paradigms as well as with traditional message passing paradigms. Converse is a component based architecture, with components encapsulating commonly needed run-time functionalities. As a result, to develop a portable runtime library for a new paradigm on top of Converse becomes extremely easy. We implemented more than ten such (interoperable) libraries, representing several parallel programming languages and paradigms [80, 89].

Adaptive MPI: Since MPI is the commonly used paradigm on parallel computers, we developed an implementation of MPI on top of Charm++: MPI processes are implemented as user level threads embedded in Charm++ objects. This required solving tricky issues of migrating threads with their stacks, and dealing with global variables [54, 33].
Fault Tolerance: Given the trend of an increasing number of processors in parallel systems, fault handling becomes an important issue. To address this problem, we have developed in the Charm++ framework a series of techniques that allow an application to make progress in its execution despite the presence of system faults. Those techniques include checkpoint-restart mechanisms [149], diskless checkpointing [45], message-logging schemes [48], and proactive processor evacuation [41]. All of these implementations are available for either Charm++ or Adaptive MPI applications.

Performance analysis and visualization tools: A parallel object language such as Charm++ provides a wealth of information to the runtime system about the behavior of the application program. Exploiting this, we developed performance visualization and automatic performance analysis techniques that can identify performance hurdles in an application [107, 83].

Going a step further, we developed closed-loop techniques [86, 87] that analyze the performance data from one run to generate a "hints" database that is utilized by subsequent runs (or iterations) to improve performance.

Parallel molecular dynamics: A large fraction of my research effort during the past few years was devoted to a collaborative project [172, 173, 14] aimed at developing a scalable and efficient parallel molecular dynamics program. The resultant program, NAMD [12], is a production quality program used to produce over a dozen published results in biophysics. It is written using Charm++, and uses Converse [77] to interoperate with libraries written in other languages. We believe it is the fastest parallel molecular dynamics program (speedup of 180 on 220 processors, not surpassed by any other production quality program). Although it has required significant effort on our part to support the nitty-gritty of a real application, it has also been a wellspring of ideas for Computer Science research.

Parallel quantum chemistry: My interest in molecular dynamics simulation has been extended into the electronic structure details governed by quantum mechanics. Our collaborative project [4] has explored how the Car-Parrinello ab initio molecular dynamics method can achieve new heights of scalability [156] by way a finer decomposition using Charm++ virtualization approach. Achieving effective scaling for the 32 molecule problem to over 1500 processors. The heavy use of Fast fourier transforms also inspired the development of our parallel FFT library [153]. Furthermore, this complex application has been a driver for parallel research in the efficient use of collective communication optimizations [139], and the use of prioritized execution to maximize the efficiency of adaptive overlap of computation with communication.

Computational Astronomy A collaborative project (with Prof. Quinn, UW) aims at attaining new levels of performance for both astronomical simulations involving gravit, gas dynamics, and including multiple timestepping over large scale-ranges. This application presents interesting tradeoffs between different parallel data structures, balancing memory use with efficiency, complex multiphase dynamics load balancing strategies, etc. Preliminary results are promising [32].

Other Collaborative Applications These include structural dynamics (with Prof. Geubelle, UIUC), space-time-meshes (Profs. Haber, Ericsson et al UIUC), and a large DOE project on rocket simulation (headed by Prof. Heath, UIUC). We have validated much or our methodology and software (Charm++, AMPI, and the unstructured mesh framework now called ParFUM) via use in these projects.

Performance Prediction and Parallel Discrete Event Simulation: We have developed POSE[50], an environment for Parallel Discrete Event Simulation (PDES), to address the need for scalable simulation frameworks capable of handling simulations with fine-grained tasks. POSE[146] is being used in a variety of areas such as VHDL-based circuit simulation and application performance prediction on very large parallel machines.

As the need for computing power grows, huge machines with thousands of processors are being designed that are potentially capable of peta-FLOPs performance. PDES can be used in the design stages of such machines to model and predict the performance of target applications on the proposed architecture[63]. Toward this end, we developed BigSim[55], a parallel simulator for performance prediction of real applications on extremely large machines. Application tasks are simulated as if running on target processors, and communication and load balancing are modeled with fixed network parameters[5].

This lead to the development[148] of modular simulations of interconnection networks for these

large parallel machines in the BigNetSim project[39]. Logs of application task execution data generated by BigSim, were used to simulate the application on a machine configured with a variety of network options. Logs could be reused for various networks — the initial task execution only needed to be performed once. This presented a challenging simulation to POSE [37], which has achieved good scalability in spite of the fine granularity.

Dynamic and irregular applications in science and engineering: We are developing a broad "multi-domain partitioning" approach for such applications, based on objects and the migration framework. Strategies for detecting application-induced imbalances, and adaptively reacting to them to restore efficiency are being developed [164]. Consistent with our philosophy, these are being tested on applications in biophysics [79], rocket simulation, and solidification of metals, in collaborative projects. We expect the resultant framework to be a broadly applicable "enabling technology".

Domain Specific Frameworks: Unstructured Meshes A few key "data structures" (structured and unstructured grids, spatial-decomposition trees, and particles) account for a large number of parallel applications. Developing domain-specific frameworks for these will enhance productivity by facilitating reuse. Although we have developed basic frameworks for all of these, the unstructured mesh framework has been highly successful, and is being used in multiple applications. A guiding principle of this framework was to retain the sequential "look-and-feel" for the application programmer, and to give control of application data-structures to them (instead of requiring a system-mandated sub-classing regimen, for example). Our future work here is focused on supporting mesh adaptivity in multiple contexts.

Cluster computing: As clusters are becoming more common and Grid computing becoming a reality, we have started to develop Faucets [44], an economy-drive framework for sharing computational resources on the Grid. We have studied different bidding strategies to provide a good economic model [151]. We have also developed techniques which leverage the Charm [168] and AMPI [71] run-time systems to tolerate intra-cluster latencies encountered in the Grid [40]. Faucets is currently deployed on several clusters in the Department of Computer Science.

State-space search/non-numeric computations: One of the early successes of our parallel object methodology was in the area of state-space search. Based on the ideas of fine-grain prioritization, and scalable load balancing, we've obtained strong results [101] in state space search for any one solution (which was a very difficult problem), iterative deepening, branch and bound [97], and so on.

Web based interactive parallel programs: We have developed a client-server interface that allows parallel programs to interact with outside world. Building on this, we have developed a web interface [174, 47] that allows one to attach to running parallel programs from a browser anywhere, monitor its performance, debug it, and interact with it.
The citations numbers above refer to the publication list in the resume. Further information about these topics can be obtained from http://charm.cs.uiuc.edu

(Papers 1-30, approxmiately, are journal papers or book chapters, papers until 136 are refered conference papers, and the ones after that are thesis and tech. reports. That is the intent: this bibliography is "under construction". So, please pardon any errors of classification).

Publications

[1] Laxmikant V. Kalé, Klaus Schulten, Robert D. Skeel, Glenn Martyna, Mark Tuckerman, James C. Phillips, Sameer Kumar, and Gengbin Zheng. Biomolecular modeling using parallel supercomputers. In S. Aluru, editor, *Handbook of computational molecular biology*, pages 34.1–34.43. Taylor and Francis, 2005.

[2] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kalé, and Klaus Schulten. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 26(16):1781–1802, 2005.

[3] Orion Lawlor, Sayantan Chakravorty, Terry Wilmarth, Nilesh Choudhury, Isaac Dooley, Gengbin Zheng, and Laxmikant Kale. Parfum: A parallel framework for unstructured meshes for scalable dynamic physics applications. *Engineering with Computers*.

[4] Ramkumar V. Vadali, Yan Shi, Sameer Kumar, L. V. Kale, Mark E. Tuckerman, and Glenn J. Martyna. Scalable fine-grained parallelization of plane-wave-based ab initio molecular dynamics for large supercomputers. *Journal of Comptational Chemistry*, 25(16):2006–2022, Oct. 2004.

[5] Gengbin Zheng, Terry Wilmarth, Praveen Jagadishprasad, and Laxmikant V. Kalé. Simulation-based performance prediction for large parallel machines. In *International Journal of Parallel Programming*, volume 33, pages 183–207, 2005.

[6] Orion Sky Lawlor and L. V. Kalé. Supporting dynamic parallel object arrays. *Concurrency and Computation: Practice and Experience*, 15:371–393, 2003.

[7] Tarmar Schlick, Robert Skeel, Axel Brünger, Laxmikant Kalé, John A.Board Jr, Jan Hermans, and Klaus Schulten. Algorithmic chanllenges in computational molecular biophysics. *Journal of Computational Physics*, 151:9–48, 1999.

[8] L. V. Kalé. Programming Languages for CSE: the state of the art. *IEEE Computational Science and Engineering*, 5(2):18–26, April-June 1998.

[9] J. Yelon and L. V. Kalé. Static networks: A powerful and elegant extension to concurrent object-oriented languages. In *Lecture Notes in Computer Science*, volume 1505, pages 143–150. Springer Verlag, 1998.

[10] Tamar Schlick, Robert D. Skeel, Axel T. Brünger, Klaus Schulten, Laxmikant V. Kale, Jan Hermans, and Jr. John A. Board. Computational biophysics today. *Journal of Computational Physics*, 1998. Submitted.

[11] James C. Phillips, Robert Brunner, Aritomo Shinozaki, Milind Bhandarkar, Neal Krawetz, Laxmikant Kalé, Robert D. Skeel, and Klaus Schulten. Avoiding algorithmic obfuscation in a message-driven parallel MD code. In *Computational Molecular Dynamics: Challenges,*

Methods, Ideas, volume 4 of *Lecture Notes in Computational Science and Engineering*, pages 472–482. Springer-Verlag, November 1998.

[12] Laxmikant Kalé, Robert Skeel, Milind Bhandarkar, Robert Brunner, Attila Gursoy, Neal Krawetz, James Phillips, Aritomo Shinozaki, Krishnan Varadarajan, and Klaus Schulten. NAMD2: Greater scalability for parallel molecular dynamics. *Journal of Computational Physics*, 151:283–312, 1999.

[13] L. V. Kale and Sanjeev Krishnan. Charm++: Parallel Programming with Message-Driven Objects. In Gregory V. Wilson and Paul Lu, editors, *Parallel Programming using C++*, pages 175–213. MIT Press, 1996.

[14] Mark Nelson, William Humphrey, Attila Gursoy, Andrew Dalke, Laxmikant Kale, Robert D. Skeel, and Klaus Schulten. NAMD—a parallel, object-oriented molecular dynamics program. *Intl. J. Supercomput. Applics. High Performance Computing*, 10(4):251–268, Winter 1996.

[15] Joshua Yelon and L. V. Kale. Agents: An undistorted representation of problem structure. In *Lecture Notes in Computer Science*, volume 1033, pages 551–565. Springer-Verlag, August 1995.

[16] L. V. Kale, B. H. Richards, and T. D. Allen. Efficient parallel graph coloring with prioritization. In *Lecture Notes in Computer Science*, volume 1068, pages 190–208. Springer-Verlag, August 1995.

[17] L. V. Kalé, B. Ramkumar, A. B. Sinha, and V. A. Saletore. The CHARM Parallel Programming Language and System: Part II – The Runtime system. *Parallel Programming Laboratory Technical Report #95-03*, 1994.

[18] L. V. Kalé, B. Ramkumar, A. B. Sinha, and A. Gursoy. The CHARM Parallel Programming Language and System: Part I – Description of Language Features. *Parallel Programming Laboratory Technical Report #95-02*, 1994.

[19] M.Nelson, W. Humprey A. Gursoy, A¿ Dalke, L.V. Kalè, R Skeel, K. Schulten, and R. Kuffrin. MDScope-A visual COmputing Environment for Structural Biology. *Computer Physics COmmunications*, 91:111–134, October 1995.

[20] John A. Board Jr., Laxmikant V. Kale, Klaus Schulten, Robert D. Skeel, , and Tamar Schlick. Modeling biomolecules: Large sclaes, longer durations. *IEEE Computational Science & Engineering*, 1:19–30, Winter 1994.

[21] David Sehr, L.V. Kale, and David A. Padua. Loop transformation for prolog programs. In *Lecture Notes in Computer Science*, pages 374–389. Springer Verlag, 1993.

[22] L.V.Kalè and B. Ramkumar. *The Reduce-OR-Process Model for Parallel Logic Programming on Nonshared Memory Machines*. John Wiley, June 1992. Editors: M. Wise and Peter Kacssuk.

[23] B.Ramkumar and L.V. Kalè. A join algorithm for combining AND parallel solutions in AND/OR parallel systems. *International Journal of Parallel Processing*, 1992.

[24] B. Ramkumar and L.V. Kale. Machine independent AND and OR parallel execution of logic programs: Part I and II. *IEEE Transactions on Parallel and Distributed Systems*, 5(2), Feb. 1991.

[25] L.V. Kalé. *Journal of Logic Programming*, 11(1):55–84, July 1991.

[26] B.Ramkumar and L.V. Kalé. Machine independent AND and OR parallel execution of logic programs Part II - compiled execution. *IEEE Transactions on Parallel and Distributed Systems*, 5(2):181–192, February 1994.

[27] B. Ramkumar and L.V. Kalé. Machine independent AND and OR parallel execution of logic programs PartI - the binding environment. *IEEE Transactions on Parallel and Distributed System*, 5(2):170–180, February 1994.

[28] L.V. Kalé and V. Saletore. Parallel state-space search for a first solution with consistent linear speedups. *Internaltional Journal of Parallel Programming*, 19(4):251–293, 1990.

[29] W.W. Shu and L.V. Kalé. Chare Kernel - a runtime support system for parallel computations. *Journal of Parallel and Distributed Computing*, 11:198–211, 1990.

[30] L.V. Kalè. An almost perfect heuristic or the N-queens problem. *Information Processing Letters*, 34(4):173–178, April 1990.

[31] L.V. Kalè, D.A. Padua, and D.C. Sehr. OR parallel execution of Prolog programs with side effects. *The Journal of Supercomputing*, 2(2):209–223, October 1988.

[32] Filippo Gioachin, Amit Sharma, Sayantan Chackravorty, Celso Mendes, Laxmikant V. Kale, and Thomas R. Quinn. Scalable cosmology simulations on parallel machines. In *7th International Meeting on High Performance Computing for Computational Science*, July 2006.

[33] Chao Huang, Gengbin Zheng, Sameer Kumar, and Laxmikant V. Kalé. Performance evaluation of adaptive MPI. In *Proceedings of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming 2006*, March 2006.

[34] Tarun Agarwal, Amit Sharma, and Laxmikant V. Kalé. Topology-aware task mapping for reducing communication contention on large parallel machines. In *PPL Technical Report 05-18*, October 2005.

[35] Sameer Kumar, Chao Huang, Gheorghe Almasi, and Laxmikant V. Kalé. Achieving strong scaling with NAMD on Blue Gene/L. In *Proceedings of IEEE International Parallel and Distributed Processing Symposium 2006*, April 2006.

[36] Orion Lawlor, Hari Govind, Isaac Dooley, Michael Breitenfeld, and Laxmikant Kale. Performance degradation in the presence of subnormal floating-point values. In *Proceedings of the International Workshop on Operating System Interference in High Performance Applications*, September 2005.

[37] Nilesh Choudhury, Yogesh Mehta, Terry L. Wilmarth, Eric J. Bohm, and Laxmikant V. Kalé. Scaling an optimistic parallel simulation of large-scale interconnection networks. In *Proceedings of the Winter Simulation Conference*, 2005.

[38] Xiangmin Jiao, Gengbin Zheng, Orion Lawlor, Phil Alexander, Mike Campbell, Michael Heath, and Robert Fiedler. An integration framework for simulations of solid rocket motors. In *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, Tucson, Arizona, July 2005.

[39] Terry L. Wilmarth, Gengbin Zheng, Eric J. Bohm, Yogesh Mehta, Nilesh Choudhury, Praveen Jagadishprasad, and Laxmikant V. Kale. Performance prediction using simulation of large-scale interconnection networks in pose. In *Proceedings of the Workshop on Principles of Advanced and Distributed Simulation*, pages 109–118, 2005.

[40] Gregory A. Koenig and Laxmikant V. Kale. Using message-driven objects to mask latency in grid computing applications. In *19th IEEE International Parallel and Distributed Processing Symposium*, April 2005.

[41] Sayantan Chakravorty, Celso Mendes and L. V. Kale. Proactive fault tolerance in large systems. In *HPCRI Workshop in conjunction with HPCA 2005*, 2005.

[42] L. V. Kalé, Mark Hills, and Chao Huang. An orchestration language for parallel objects. In *ProceediNg of Seventh Workshop on Languages, Compilers, and Run-time Support for Scalable Systems (LCR 04)*, Houston, Texas, October 2004.

[43] Jayant DeSouza and Laxmikant V. Kalé. MSA: Multiphase specifically shared arrays. In *Proceedings of the 17th International Workshop on Languages and Compilers for Parallel Computing*, West Lafayette, Indiana, USA, September 2004.

[44] Laxmikant V. Kalé, Sameer Kumar, Jayant DeSouza, Mani Potnuru, and Sindhura Bandhakavi. Faucets: Efficient resource allocation on the computational grid. In *Proceedings of the 2004 International Conference on Parallel Processing*, August 2004.

[45] Gengbin Zheng, Lixia Shi, and Laxmikant V. Kalé. Ftc-charm++: An in-memory checkpoint-based fault tolerant runtime for charm++ and mpi. In *2004 IEEE International Conference on Cluster Computing*, San Dieago, CA, September 2004.

[46] Laxmikant V. Kale, Gengbin Zheng, Chee Wai Lee, and Sameer Kumar. Scaling applications to massively parallel machines using projections performance analysis tool. In *Future Generation Computer Systems Special Issue on: Large-Scale System Performance Modeling and Analysis*, number to appear, 2005.

[47] Rashmi Jyothi, Orion Sky Lawlor, and L. V. Kale. Debugging support for Charm++. In *PADTAD Workshop for IPDPS 2004*, page 294. IEEE Press, 2004.

[48] Sayantan Chakravorty and L. V. Kale. A fault tolerant protocol for massively parallel machines. In *FTPDS Workshop for IPDPS 2004*. IEEE Press, 2004.

[49] Gengbin Zheng, Terry Wilmarth, Orion Sky Lawlor, Laxmikant V. Kalé, Sarita Adve, David Padua, and Philippe Geubelle. Performance modeling and programming environments for petaflops computers and the blue gene machine. In *NSF Next Generation Systems Program Workshop, 18th International Parallel and Distributed Processing Symposium(IPDPS)*, Santa Fe, New Mexico, April 2004. IEEE Press.

[50] Terry Wilmarth and L. V. Kalé. Pose: Getting over grainsize in parallel discrete event simulation. In *2004 International Conference on Parallel Processing*, pages 12–19, August 2004.

[51] Laxmikant V. Kalé. Performance and productivity in parallel programming via processor virtualization. In *Proc. of the First Intl. Workshop on Productivity and Performance in High-End Computing (at HPCA 10)*, Madrid, Spain, February 2004.

[52] Sameer Kumar and L. V. Kale. Opportunities and Challenges of Modern Communication Architectures: Case Study with QsNet. Technical Report 03-15, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, 2003.

[53] Sameer Kumar and L. V. Kale. Scaling collective multicast on fat-tree networks. In *ICPADS*, Newport Beach, CA, July 2004.

[54] Chao Huang, Orion Lawlor, and L. V. Kalé. Adaptive MPI. In *Proceedings of the 16th International Workshop on Languages and Compilers for Parallel Computing (LCPC 2003), LNCS 2958*, pages 306–322, College Station, Texas, October 2003.

[55] Gengbin Zheng, Gunavardhan Kakulapati, and Laxmikant V. Kalé. Bigsim: A parallel simulator for performance prediction of extremely large parallel machines. In *18th International Parallel and Distributed Processing Symposium (IPDPS)*, Santa Fe, New Mexico, April 2004.

[56] Laxmikant V. Kalé, Sameer Kumar, Gengbin Zheng, and Chee Wai Lee. Scaling molecular dynamics to 3000 processors with projections: A performance analysis case study. In *Terascale Performance Analysis Workshop, International Conference on Computational Science(ICCS)*, Melbourne, Australia, June 2003.

[57] Jayant DeSouza and L. V. Kalé. Jade: A parallel message-driven Java. In *Proc. Workshop on Java in Computational Science, held in conjunction with the International Conference on Computational Science (ICCS 2003)*, Melbourne, Australia and Saint Petersburg, Russian Federation, June 2003.

[58] L. V. Kale, Sameer Kumar, and Krishnan Vardarajan. A Framework for Collective Personalized Communication. In *Proceedings of IPDPS'03*, Nice, France, April 2003.

[59] Laxmikant V. Kalé. The virtualization model of parallel programming : Runtime optimizations and the state of art. In *LACSI 2002*, Albuquerque, October 2002.

[60] James C. Phillips, Gengbin Zheng, Sameer Kumar, and Laxmikant V. Kalé. NAMD: Biomolecular simulation on thousands of processors. In *Proceedings of SC 2002*, Baltimore, MD, September 2002.

[61] Orion Sky Lawlor and L. V. Kalé. A voxel-based parallel collision detection algorithm. In *Proceedings of the International Conference in Supercomputing*, pages 285–293. ACM Press, June 2002.

[62] James Phillips, Gengbin Zheng, and Laxmikant V. Kalé. Namd: Biomolecular simulation on thousands of processors. In *Workshop: Scaling to New Heights*, Pittsburgh, PA, May 2002.

[63] Gengbin Zheng, Arun Kumar Singla, Joshua Mostkoff Unger, and Laxmikant V. Kalé. A parallel-object programming model for petaflops machines and blue gene/cyclops. In *NSF Next Generation Systems Program Workshop, 16th International Parallel and Distributed Processing Symposium(IPDPS)*, Fort Lauderdale, FL, April 2002.

[64] Laxmikant V. Kalé, Sameer Kumar, and Jayant DeSouza. A malleable-job system for timeshared parallel machines. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2002)*, May 2002.

[65] Neelam Saboo, Arun Kumar Singla, Joshua Mostkoff Unger, and L. V. Kalé. Emulating petaflops machines and blue gene. In *Workshop on Massively Parallel Processing (IPDPS'01)*, San Francisco, CA, April 2001.

[66] Milind Bhandarkar and L. V. Kale. An Interface Model for Parallel Components. In *Proceedings of the Workshop on Languages and Compilers for Parallel Computing (LCPC), Cumberland Falls, KY*, August 2001.

[67] O. Lawlor and L. V. Kalé. Supporting dynamic parallel object arrays. In *Proceedings of ACM 2001 Java Grande/ISCOPE Conference*, pages 21–29, Stanford, CA, Jun 2001.

[68] Robert K. Brunner, James C. Phillips, and Laxmikant V. Kale. Scalable Molecular Dynamics for Large Biomolecular Systems. In *Proceedings of Supercomputing (SC) 2000, Dallas, TX, November 2000. Nominated for Gordon Bell Award.*, November 2000.

[69] L. V. Kale, Milind Bhandarkar, and Robert Brunner. Run-time Support for Adaptive Load Balancing. In J. Rolim, editor, *Lecture Notes in Computer Science, Proceedings of 4th Workshop on Runtime Systems for Parallel Programming (RTSPP) Cancun - Mexico*, volume 1800, pages 1152–1159, March 2000.

[70] Eric deStruler, Jay Hoeflinger, L. V. Kale, and Milind Bhandarkar. A New Approach to Software Integration Frameworks for Multi-physics Simulation Codes. In *Proceedings of IFIP TC2/WG2.5 Working Conference on Architecture of Scientific Software, Ottawa, Canada*, pages 87–104, October 2000.

[71] Milind Bhandarkar, L. V. Kale, Eric de Sturler, and Jay Hoeflinger. Object-Based Adaptive Load Balancing for MPI Programs. In *Proceedings of the International Conference on Computational Science, San Francisco, CA, LNCS 2074*, pages 108–117, May 2001.

[72] Milind Bhandarkar and L. V. Kalé. A Parallel Framework for Explicit FEM. In M. Valero, V. K. Prasanna, and S. Vajpeyam, editors, *Proceedings of the International Conference on High Performance Computing (HiPC 2000), Lecture Notes in Computer Science*, volume 1970, pages 385–395. Springer Verlag, December 2000.

[73] Milind Bhandarkar, Gila Budescu, William F. Humphrey, Jesus A. Izaguirre, Sergei Izrailev, Laxmikant V. Kale, Dorina Kosztin, Ferenc Molnar, James C. Phillips, and Klaus Schulten. Biocore: A collaboratory for structural biology. *In Agostino G. Bruzzone, Adelinde Uchrmacher, and Ernest H. Page, editors, Proceedings of the SCS International Conference on Web-Based Modeling and Simulation*, pages 242–251, 1999.

[74] Parthasarathy Ramachandran and L. V. Kalé. Mulitlingual debugging support for data-driven and thread-based parallel languages. In *Lecture Notes in Computer Science: Proc. of 12th International Workshop on Languages and Compilers for Parallel Computing (LCPC '99)*, pages 236–250. Springer-Verlag, August 1999.

[75] Robert K. Brunner and Laxmikant V. Kalé. Adapting to load on workstation clusters. In *The Seventh Symposium on the Frontiers of Massively Parallel Computation*, pages 106–112. IEEE Computer Society Press, February 1999.

[76] Laxmikant Kale, Robert Brunner, James Phillips, and Krishnan Varadarajan. Application performance of a linux cluster using converse. In *Proceedings of 3rd Workshop on Runtime Systems for Parallel Programming, Rolim et al (Eds)*, Lecture Notes in Computer Science 1586, pages 483–495. IPPS/SPDP, Springer, 1999.

[77] L. V. Kalé, M. Bhandarkar, R. Brunner, N. Krawetz, J. Phillips, and A. Shinozaki. Namd: A case study in multilingual parallel programming. In *Languages and Compilers for Parallel Computing*, number 1366 in Lecture Notes in Computer Science, pages 367–381. Springer-Verlag, 1998.

[78] Robert Brunner, Laxmikant Kalé, and James Phillips. Flexibility and interoperability in a parallel molecular dynamics code. In *Object Oriented Methods for Inter-operable Scientific and Engineering Computing*, pages 80–89. SIAM, October 1998.

[79] L. V. Kalé, Milind Bhandarkar, and Robert Brunner. Load balancing in parallel molecular dynamics. In *Fifth International Symposium on Solving Irregularly Structured Problems in Parallel*, volume 1457 of *Lecture Notes in Computer Science*, pages 251–261, 1998.

[80] Robert Brunner L. V. Kale, Milind Bhandarkar and Joshua Yelon. Multiparadigm, Multilingual Interoperability: Experience with Converse. In *Proceedings of 2nd Workshop on Runtime Systems for Parallel Programming (RTSPP) Orlando, Florida - USA*, Lecture Notes in Computer Science, March 1998.

[81] L. V. Kalé, Milind Bhandarkar, Robert Brunner, N. Krawetz, J. Phillips, and A. Shinozaki. Namd: A case study in multilingual parallel programming. In *Proc. 10th International Workshop on Languages and Compilers for Parallel Computing*, Minneapolis, Minnesota, August 1997.

[82] L. V. Kalé, Milind Bhandarkar, and Terry Wilmarth. Design and implementation of parallel java with a global object space. In *Proc. Conf. on Parallel and Distributed Processing Technology and Applications*, pages 235–244, Las Vegas, Nevada, July 1997.

[83] Amitabh Sinha and L. V. Kale. Towards Automatic Peformance Analysis. In *Proceedings of International Conference on Parallel Processing*, volume III, pages 53–60, August 1996.

[84] L. V. Kale and Milind Bhandarkar. Structured Dagger: A Coordination Language for Message-Driven Programming. In *Proceedings of Second International Euro-Par Conference*, volume 1123-1124 of *Lecture Notes in Computer Science*, pages 646–653, September 1996.

[85] L. V. Kale and Joshua Yelon. Threads for Interoperable Parallel Programming. In *Proc. 9th Conference on Languages and Compilers for Parallel Computers*, San Jose, California, August 1996.

[86] Sanjeev Krishnan and L. V. Kale. Automating Parallel Runtime Optimizations Using Post-Mortem Analysis. In *Proc. 10th ACM International Conference on Supercomputing*, pages 221–228, Philadelphia, PA, May 1996.

[87] Sanjeev Krishnan and L. V. Kale. Automating Runtime Optimizations for Load Balancing in Irregular Problems. In *Proc. Conf. on Parallel and Distributed Processing Technology and Applications*, San Jose, California, August 1996.

[88] Sanjeev Krishnan and L. V. Kale. A parallel array abstraction for data-driven objects. In *Proceedings of Parallel Object-Oriented Methods and Applications Conference*, Santa Fe, NM, February 1996.

[89] Milind Bhandarkar and L. V. Kale. MICE: A Prototype MPI Implementation in Converse Environment. In *Proceedings of the second MPI Developers Conference*, pages 26–31, South Bend, Indiana, July 1996.

[90] L. V. Kale, Milind Bhandarkar, Narain Jagathesan, Sanjeev Krishnan, and Joshua Yelon. Converse: An Interoperable Framework for Parallel Programming. In *Proceedings of the 10th International Parallel Processing Symposium*, pages 212–217, April 1996.

[91] Sanjeev Krishnan and L. V. Kale. A parallel adaptive fast multipole algorithm for n-body problems. In *Proceedings of the International Conference on Parallel Processing*, pages III 46 – III 50, August 1995.

[92] L. V. Kale and Attila Gursoy. Modularity, reuse and efficiency with message-driven libraries. In *Proc. 27th Conference on Parallel Processing for Scientific Computing*, pages 738–743, February 1995.

[93] E. Kornkven and L. V. Kale. Efficient implementation of high performance fortran via adaptive scheduling – an overview. In *Proceedings of the International Workshop on Parallel Processing*, Bangalore, India, December 1994.

[94] E. Kornkven and L. V. Kale. Efficient implementation of high performance fortran via adaptive scheduling – an overview. In *Proceedings of the International Workshop on Parallel Processing*, Bangalore, India, December 1994.

[95] L. V. Kale. Application oriented and computer science centered HPCC research. In *Proceedings of Developing a CS Agenda for High-Performance Computing*, pages 98–105, March 1994. Position Paper.

[96] A. Gursoy and L.V. Kalé. Dagger: Combining the Benefits of Synchronous and Asynchronous Communication Styles. In H. G. Siegel, editor, *Proceedings of the 8th International Parallel Processing Symposium*, pages 590–596, Cancun, Mexico, April 1994.

[97] A. Sinha and L.V. Kalé. A load balancing strategy for prioritized execution of tasks. In *Seventh International Parallel Processing Symposium*, pages 230–237, Newport Beach, CA., April 1993.

[98] L.V. Kale and Sanjeev Krishnan. Medium grained execution in concurrent object-oriented systems. In *Workshop on Efficient Implementation of Concurrent Object Oriented Languages, at OOPSLA 1993*, September 1993.

[99] Attila Gursoy and L. V. Kale. Simulating Message-Driven Programs. In *Proceedings of International Conference on Parallel Processing*, volume III, pages 223–230, August 1996.

[100] L. V. Kale and Attila Gursoy. Performance benefits of message driven execution. In *Intel Supercomputer User's Group*, St. Louis, October 1993.

[101] L.V. Kale, B. Ramkumar, V. Saletore, and A. B. Sinha. Prioritization in parallel symbolic computing. In T. Ito and R. Halstead, editors, *Lecture Notes in Computer Science*, volume 748, pages 12–41. Springer-Verlag, 1993.

[102] A. Sinha and L.V. Kalé. Information Sharing Mechanisms in Parallel Programs. In H.J. Siegel, editor, *Proceedings of the 8th International Parallel Processing Symposium*, pages 461–468, Cancun, Mexico, April 1994.

[103] L.V. Kalé and S. Krishnan. CHARM++: A Portable Concurrent Object Oriented System Based on C++. In A. Paepcke, editor, *Proceedings of OOPSLA'93*, pages 91–108. ACM Press, September 1993.

[104] L. V. Kale and Sanjeev Krishnan. A comparison based parallel sorting algorithm. In *Proceedings of the 22nd International Conference on Parallel Processing*, pages 196–200, St. Charles, IL, August 1993.

[105] Amitabh Sinha and L.V. Kale. A load balancing strategy for prioritized execution of tasks. In *Workshop on Dynamic Object Placement and Load Balancing, in co-operation with ECOOP's 92*, Utrecht, The Netherlands, April 1992.

[106] A. Gursoy, L.V. Kale, and S.P. Vanka. Unsteady fluid flow calculations using a machine independent parallel programming environment. In R. B. Pelz, A. Ecer, and J. Hauser, editors, *Parallel Computational Fluid Dynamics '92*, pages 175–185. North-Holland, 1993.

[107] L.V. Kalé and Amitabh Sinha. Projections: A preliminary performance tool for charm. In *Parallel Systems Fair, International Parallel Processing Symposium*, pages 108–114, Newport Beach, CA, April 1993.

[108] L.V. Kale. Parallel problem solving. In Vipin Kumar, P. S. Gopolakrishnan, and L. N. Kanal, editors, *Parallel Algorithms for Machine Intelligence and Vision*, pages 146–181. Springer-Verlag, 1989.

[109] D. Sehr and L.V. Kale. Estimating the inherent parallelism in Prolog programs. In *Proceedings of the 1992 International Conference on Fifth Generation Computer Systems*, Tokyo, Japan, June 1992.

[110] W. Fenton, B. Ramkumar, V.A. Saletore, A.B. Sinha, and L.V. Kale. Supporting machine independent programming on diverse parallel architectures. In *Proceedings of the International Conference on Parallel Processing*, pages 193–201, St. Charles, IL, August 1991.

[111] David C. Sehr, Laxmikant V. Kalé, and David A. Padua. Fortran-style transformations for functional programs (Extended abstract). In *Proceedings of the 1991 International Conference on Parallel Processing*, pages 282–283, St. Charles, IL, August 1991.

[112] A. Gursoy and L.V. Kale. High-level support for divide-and-conquer parallelism. In *Proceedings of Supercomputing '91*, pages 283–292, November 1991.

[113] L.V. Kalé and B. Ramkumar. Implementing a parallel Prolog interpreter on multiprocessors. In *5th International Parallel Processing Symposium*, pages 543–548, Anaheim, CA, April 1991.

[114] L.V. Kalé and Vikram Saletore. Efficient parallel execution of IDA* on shared and distributed memory multiprocessors. In *Proceedings of the Sixth Distributed Memory Computing Conference (DMCC6)*, Portland, Oregon, April 1991.

[115] Vikram A. Saletore. A distributed and adaptive dynamic load balancing scheme for parallel processing of medium-grain tasks. In *Proceedings of the Fifth Distributed Memory Computing Conference (5th DMCC'90)*, volume II, Architecture Software Tools, and Other General Issues, pages 994–999, Charleston, SC, April 1990. IEEE.

[116] L. V. Kalé and B. Ramkumar. Joining AND-parallel solutions in AND/OR parallel systems. In *Proceedings of the North American Conference on Logic Programming*, pages 614–631, October 1990.

[117] L.V. Kale. The Chare Kernel parallel programming language and system. In *Proceedings of the International Conference on Parallel Processing*, volume II, pages 17–25, August 1990.

[118] V. Saletore and L.V. Kale. Consistent linear speedups for a first solution in parallel state-space search. In *Proceedings of the AAAI*, pages 227–233, August 1990.

[119] B. Ramkumar and L. V. Kalé. A Chare Kernel implementation of a parallel Prolog compiler. In *Proceedings of the Second Conference on Principles and Practice of Parallel Programming*, pages 99–108, Seattle, WA, March 1990.

[120] B. Ramkumar and L. V. Kalé. An abstract machine for the reduce or process model for parallel Prolog. In *Proceedings of Knowledge Based Computer Systems Conference*, pages 267–276, Bombay, India, December 1989.

[121] W. W. Shu and L. V. Kalé. A dynamic load balancing strategy for the Chare Kernel system. In Supercomputing89 [175], pages 389–398.

[122] B. Ramkumar and L. V. Kalé. Compiled execution of the Reduce-Or process model on multiprocessors. In *The North American Conference on Logic Programming*, pages 313–331, Oct 1989.

[123] V. Saletore and L. V. Kalé. Obtaining first solutions fast in parallel problem solving. In *The North American Conference on Logic Programming*, pages 390–408, Oct 1989.

[124] L. V. Kalé and W. Shu. The Chare Kernel base language: Preliminary performance results. In ICPP89 [176], pages 118–121.

[125] V. P. Pethe, C. P. Rippey, and L. V. Kalé. A specialized expert system for judicial decision support. In *Proceedings of the Second International Conference on AI and Law (ICAIL-89)*, pages 190–194, June 1989.

[126] L. V. Kalé. A brief perspective on parallel programming. In *IEEE Tencon Regional Conference (Invited)*, pages 1085–1088, Bombay, India, November 1989.

[127] L. V. Kalé, B. Ramkumar, and W. Shu. Parallel Prolog on Intel's iPSC/2. In *Proceedings of the Fourth Conference on Hypercube Concurrent Computers and Applications*, March 1989.

[128] L. V. Kalé. The mesh superceded? In *Proceedings of ACM Computer Science Conference*, pages 180–187, Feb 1989. Also available as TR#1362, dated July 1987.

[129] L. V. Kalé. A tree representation for parallel problem solving. In *Proceedings of AAAI*, pages 677–681, August 1988.

[130] L. V. Kalé, B. Ramkumar, and W. Shu. A memory independent binding environment for AND and OR parallel execution of logic programs. In *The Joint International Conference/Symposium on Logic Programming*, pages 1223–1240, Seattle, WA, 1988.

[131] L. V. Kalé. Comparing the performance of two dynamic load distribution methods. In *Proceedings of the 1988 International Conference on Parallel Processing*, pages 8–11, St. Charles, IL, August 1988.

[132] M. Gooley, L. V. Kalé, D. Padua, D. Sehr, B. Ramkumar, U. S. Reddy, W. Shu, and B. Wah. Prolog research at University of Illinois. In *Compcon*, pages 68–73, San Francisco, Feb 1988.

[133] L. V. Kalé and B. Ramkumar. D-trees: A class of dense regular interconnection topologies. In *Proceedings of the Frontiers of Massively Parallel Computation*, pages 207–210, Oct 1988.

[134] L. V. Kalé. Parallel execution of logic programs: the REDUCE-OR process model. In *Proceedings of Fourth International Conference on Logic Programming*, pages 616–632, May 1987.

[135] L. V. Kalé. 'Completeness' and 'Full parallelism' of parallel logic programming schemes. In *Proceedings of 1987 Symposium on Logic Programming*, pages 125–133, San Francisco, CA, 1987.

[136] L.V. Kalé. Optimal communication neighborhoods. In *Proceedings of the 1986 International Conference on Parallel Processing*, pages 823–826, St. Charles, Illinois, August 1986.

[137] L. V. Kalé. Lattice-mesh: a multi-bus topology. In *Proceedings of the 1985 International Conference on Parallel Processing*, pages 700–703, St. Charles, Illinois, August 1985.

[138] L. V. Kalé and D. S. Warren. A class of architectures for Prolog machine. In *Proceedings of the Conference on Logic Programming*, pages 171–182, Uppsala, Sweden, July 1984.

[139] Sameer Kumar, Yan Shi, Eric Bohm, and L. V. Kale. Scalable, fine grain, parallelization of the car-parrinello ab initio molecular dynamics method. Technical report, UIUC, Dept. of Computer Science, 2005.

[140] Orion Sky Lawlor. *Impostors for Parallel Interactive Computer Graphics.* PhD thesis, University of Illinois at Urbana-Champaign, December 2004.

[141] Yogesh Mehta. Low diameter regular graph as a network topology in direct and hybrid interconnection networks. Master's thesis, Dept. of Computer Science, University of Illinois, 2005.

[142] Tarun Agarwal. Strategies for topology-aware task mapping and for rebalancing with bounded migrations. Master's thesis, Dept. of Computer Science, University of Illinois, 2005.

[143] Gengbin Zheng. *Achieving High Performance on Extremely Large Parallel Machines: Performance Prediction and Load Balancing.* PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 2005.

[144] Sayantan Chakravorty. Implementation of parallel mesh partition and ghost generation for the finite element mesh framework. Master's thesis, Dept. of Computer Science, University of Illinois, 2005. http://charm.cs.uiuc.edu/papers/SayantanMSThesis.html.

[145] Sameer Kumar, Laxmikant V. Kale, and Craig Stunkel. Architecture for supporting hardware collectives in output-queued high-radix routers. Technical Report 05-02, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, Mar 2005.

[146] Terry L. Wilmarth. *POSE: Scalable General-purpose Parallel Discrete Event Simulation.* PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 2005.

[147] Vikas Mehta. Leanmd: A charm++ framework for high performance molecular dynamics simulation on large parallel machines. Master's thesis, University of Illinois at Urbana-Champaign, 2004.

[148] Praveen Kumar Jagadishprasad. Parallel simulation of large scale interconnection networks used in high performance computing. Master's thesis, University of Illinois at Urbana-Champaign, 2004.

[149] Chao Huang. System support for checkpoint and restart of charm++ and ampi applications. Master's thesis, Dept. of Computer Science, University of Illinois, 2004.

[150] Gregory Allen Koenig. An efficient implementation of Charm++ on Virtual Machine Interface. Master's thesis, University of Illinois at Urbana-Champaign, 2003.

[151] Sindhura Bandhakavi. Analyzing bidding strategies for schedulers in a simulated multiple-cluster market driven environment. Master's thesis, University of Illinois at Urbana-Champaign, 2003.

[152] Rashmi Jyothi. Debugging support for charm++. Master's thesis, University of Illinois at Urbana-Champaign, 2003.

[153] Ramkumar Vadali. Parallelizing cpaimd using charm++. Master's thesis, University of Illinois at Urbana-Champaign, 2003.

[154] Jonathan A. Booth. Balancing priorities and load for state space search on large parallel machines. Master's thesis, University of Illinois at Urbana-Champaign, 2003.

[155] Mani Potnuru. Automatic out-of-core exceution support for charm++. Master's thesis, University of Illinois at Urbana-Champaign, 2003.

[156] Ramkumar Vadali, L. V. Kale, Glenn Martyna, and Mark Tuckerman. Scalable parallelization of ab initio molecular dynamics. Technical report, UIUC, Dept. of Computer Science, 2003.

[157] Milind A. Bhandarkar. *Charisma: A Component Architecture for Parallel Programming*. PhD thesis, Dept. of Computer Science, University of Illinois, 2002.

[158] Orion Lawlor, Milind Bhandarkar, and Laxmikant V. Kalé. Adaptive mpi. Technical Report 02-05, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, 2002.

[159] Puneet Narula. An adaptive mesh refinement (amr) library using charm++. Master's thesis, University of Illinois at Urbana-Champaign, 2002.

[160] Sameer Kumar. An adaptive job scheduler for timeshared parallel machines. Master's thesis, Dept. of Computer Science, University of Illinois, 2001. http://charm.cs.uiuc.edu/papers/AdaptiveJobThesis01.html.

[161] Orion Lawlor. A grid-based parallel collision detection algorithm. Master's thesis, Dept. of Computer Science, University of Illinois, 2001.

[162] Neelam Saboo and L. V. Kalé. Improving paging performace with object prefetching. Technical Report 01-02, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, July 2001.

[163] Robert K. Brunner. Versatile automatic load balancing with migratable objects. TR 00-01, January 2000.

[164] Robert K. Brunner and Laxmikant V. Kalé. Handling application-induced load imbalance using parallel objects. In *Parallel and Distributed Computing for Symbolic and Irregular Applications*, pages 167–181. World Scientific Publishing, 2000.

[165] L. V. Kalé and W. Shu. The Chare Kernel language for parallel programming: A perspective. Technical Report UIUCDCS-R-88-1451, Department of Computer Science, University of Illinois, August 1988.

[166] L. V. Kalé and W. Shu. The Chare Kernel base language: Preliminary performance results. In ICPP89 [176], pages 118–121.

[167] W. W. Shu and L. V. Kalé. Chare Kernel - a runtime support system for parallel computations. *Journal of Parallel and Distributed Computing*, 11:198–211, 1990.

[168] L.V. Kalé. The Chare Kernel parallel programming language and system. In *Proceedings of the International Conference on Parallel Processing*, volume II, pages 17–25, August 1990.

[169] L. V. Kalé and D. S. Warren. A class of architectures for Prolog machine. In *Proceedings of the Conference on Logic Programming*, pages 171–182, Uppsala, Sweden, July 1984.

[170] A. Gursoy and L.V. Kale. Tolerating latency with dagger. In *Proceedings of the Eigth International Symposium on Computer and Information Sciences*, Istanbul, Turkey, November 1993.

[171] W. W. Shu and L. V. Kalé. A dynamic load balancing strategy for the Chare Kernel system. In Supercomputing89 [175], pages 389–398.

[172] John A. Board Jr., Laxmikant V. Kale, Klaus Schulten, Robert D. Skeel, , and Tamar Schlick. Modeling biomolecules: Larger scales, longer durations. *IEEE Computational Science & Engineering*, 1:19–30, Winter 1994.

[173] M. Nelson, W. Humprey A. Gursoy, A. Dalke, L.V. Kalé, R. Skeel, K. Schulten, and R. Kuffrin. MDScope-A visual Computing Environment for Structural Biology. *Computer Physics Communications*, 91:111–134, October 1995.

[174] Parthasarathy Ramachandran and L. V. Kalé. Web-based interaction and monitoring for parallel programs. Technical Report 99-05, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, 1999.

[175] *Proceedings of Supercomputing '89*, November 1989.

[176] *Proceedings of the 1989 International Conference on Parallel Processing*, St. Charles, IL, August 1989.

Software developed and distributed via the web

- Charm++ : parallel object language, as a C++ library
- Converse: Portable parallel runtime framework.
- ParFUM: Parallel Framework for Unstructured Mesh computations
- NAMD: Parallel Molecular Dynamics program (available from : http://ks.uiuc.edu).
- Master-slave library, with a sophisticated load balancer.
- mdPerl: parallel Perl based on Converse
- Tempo: message passing with user level threads.
- Generic branch-and-bound : A C++ class library that supports template based implementation of the parallel branch and bound algorithm, on top of Charm++ with prioritized load balancing.
- A "standard library" of parallel components, built on top of Converse, and Charm++; In progress.

  All the above software, except NAMD, is distributed from http://charm.cs.uiuc.edu.