Girija J. Narlikar

CMU Computer Science Dept.
5000 Forbes Avenue
Pittsburgh, PA 15213
Fax: (412) 268-5576

girija@cs.cmu.edu
http://www.cs.cmu.edu/~girija
Phone: (412) 268-3337 [O]
(412) 802-0459 [H]

Research Interests

Multithreading, distributed systems and applications, scheduling, parallel languages and systems, design and implementation of practical algorithms, performance modeling, scalable systems for mining and information retrieval.

Education

Carnegie Mellon University August 1993–present.
Ph.D. in Computer Science expected February 1999.
Dissertation: "Space-Efficient Multithreading".
Research advisor: Guy Blelloch.
Thesis Committee: Guy Blelloch, Thomas Gross, Bruce Maggs, and Charles Leiserson (MIT).
M.S. in Computer Science received May 1995.

Indian Institute of Technology, Bombay. August 1989–May 1993.
B.Tech. in Computer Science and Engineering received May 1993.

Research Experience

- **Carnegie Mellon University** September 1993–present.

  Graduate Student Researcher in the SCANDAL project with Prof. Guy Blelloch. The project is involved in developing applicative and portable parallel languages such as NESL, and fast parallel algorithms. My contributions have focused mainly on space-efficient multithreading, NESL implementation, and N-body algorithms:

  – Designed provably space and time efficient scheduling algorithms for multithreading systems as part of my Ph.D. thesis. Also built fast runtime systems based on these algorithms. Summary of my thesis research is on page 4.
  – Built a portable interpreter for the NESL programming language. Designed and implemented a Java-based graphics library to make NESL programs available as applets over the web.
  – Analyzed, implemented and compared three popular N-body algorithms for different levels of accuracy, numbers of particles, and types of forces.
  – Conducted an experimental comparison of message passing and distributed shared memory on a network of workstations for a variety of applications, as part of an operating systems project.

- **NEC Research Institute** June 1998 and Summer 1997.

  Consultant and Summer Intern with Satish Rao. Designed and implemented a distributed object system on a cluster of PCs with a fast interconnect. The goal of the library was to combine the convenience of a global name space for objects with the efficiency and predictability of bulk synchronous programming.

- **DEC Systems Research Center** Summer 1995.

  Summer Intern with Chandramohan Thekkath. Designed and implemented low-overhead runtime optimizations for a distributed shared memory library on a network of Alphas, and evaluated their effectiveness for a set of applications.

- **Indian Institute of technology, Bombay** August 1992-April 1993.

  Senior Thesis with Dhananjay Dhamdhere. Formalized an algorithm for Incremental Dataflow Analysis.

- **Center for Development of Advanced Computing (CDAC), India** Summer 1992.

  Summer Intern. Ported gcc to a transputer platform (T-800) and compared performance with the native compiler.

- **Indian Institute of technology, Bombay** Spring 1992.

  Designed and implemented a query database to computerize the Central Stores Purchasing System at IIT Bombay.

Teaching Experience

- **Carnegie Mellon University** Fall 1995.

  Teaching Assistant for senior-level Operating Systems class. Guided students in the implementation of a kernel (including virtual memory management) and a file system. Helped create and grade exams and projects.

- **Carnegie Mellon University** Spring 1994.

  Teaching Assistant for undergraduate Data Structures and Complexity class. Taught four recitation classes a week, helped create and grade exams.

- **Indian Institute of Technology, Bombay** Fall 1991.

  Teaching Assistant for introductory programming class. Conducted recitations and graded exams and projects.

Refereed Publications

- G. Narlikar and G. Blelloch. *"Space efficient implementation of nested parallelism"*, accepted for publication in ACM Transactions on Programming Languages and Systems (TOPLAS). Earlier version appeared in Proc. ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), June 1997.
- G. Narlikar. *"Scheduling threads for low space requirement and good locality"*, to appear in Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA), June 1999.
- M. Goudreau, K. Lang, G. Narlikar and S. Rao. *"BOS is Boss: A Case for Bulk-Synchronous Object Systems"*, to appear in Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA), June 1999.
- G. Narlikar and G. Blelloch. *"Pthreads for dynamic and irregular parallelism"*, Proc. SC98: High Performance Networking and Computing, November 1998. *(Best student paper award.)*
- J. Hardwick, G. Narlikar, and J. Sipelstein. *"Interactive Simulations on the Web: Compiling NESL into Java"*, Concurrency: Practice and Experience, Vol.9(11):1075-1089, November 1997. Also appeared in Proc. ACM Workshop on Java for Science and Engineering Computation, June 1997.
- G. Blelloch, P. Gibbons, Y. Matias, and G. Narlikar. *"Space efficient scheduling of parallelism with synchronization variables"*, Proc. ACM Symposium on Parallel Algorithms and Architectures (SPAA), June 1997.
- G. Blelloch and G. Narlikar. *"A practical comparison of N-body algorithms"*, Parallel Algorithms. Series in Discrete Mathematics and Theoretical Computer Science, Volume 30, 1997.

Other Publications

- G. Narlikar. *"A Parallel, Multithreaded Decision Tree Builder"*, Tech. Report CMU-CS-98-184, December 1998.
- G. Narlikar and G. Blelloch. *"Pthreads for dynamic parallelism"*, Tech. Report CMU-CS-98-114 (expanded version of SC98 paper), April 1998.
- G. Narlikar and G. Blelloch. *"A framework for space and time efficient scheduling of parallelism"*, Tech. Report CMU-CS-96-197, December 1996.
- G. Blelloch and G. Narlikar. *"A comparison of two N-body algorithms"*, Proc. DIMACS Implementation Challenge Workshop, October 1994.

Technical Presentations

- *"Pthreads for irregular and dynamic parallelism"*, SC98: High Performance Networking and Computing, November 1998. Also presented at Sun Microsystems, March 1998 and the NOW Project Retreat, January 1998.
- *"Space-efficient multithreading"*, Yale Multithreading Workshop, June 1998.
- *"Interactive Simulations on the Web: Compiling NESL into Java"*, ACM 1997 Workshop on Java for Science and Engineering Computation, June 1997.
- *"Space efficient implementation of nested parallelism"*, ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), June 1997, Las Vegas. Also presented at DEC Systems Research Center, October 1997.
- *"Parallel N-body methods"*, guest lecture for graduate level Parallel Computing class at Carnegie Mellon University, March 1997.
- *"A comparison of two N-body algorithms"*, DIMACS Implementation Challenge Workshop, October 1994, Rutgers University.

Honors

- Best Student Paper Award at "SC98: High Performance Networking and Computing", November 1998.
- Graduate Student Member of Sigma Xi Scientific Research Society, June 1997–present.
- Graduate Research Fellowship, CMU Computer Science Department, August 1993–present.
- Deshmukh Gold Medal for graduating with the highest GPA in the Computer Science and Engineering Department at IIT Bombay, May 1993.
- National Talent Scholarship awarded by the Government of India, 1987-1993.

Other activities

- Member of Technical Papers Committee, SC99.
- Reviewer for the Journal of the ACM, Theory of Computing Systems, Currency: Practice and Experience, IPPS, PPoPP, SPAA.
- Student member of ACM, IEEE, IEEE Computer Society.
- Volunteer for International Graduate Student Orientation at CMU, 1998.
- Organized free classical music concerts as a coordinator of SPIC-MACAY, 1996-1997.
- Enjoy hiking, tennis, racketball, running, cooking.

References

Prof. Guy Blelloch
CMU Computer Science Dept.
5000 Forbes Avenue
Pittsburgh, PA 15213.
(412) 268-6245
Guy.Blelloch@cs.cmu.edu

Prof. Bruce Maggs
MIT Lab. for Computer Science
545 Technology Square
Cambridge, MA 02139.
(617) 253-6035
bmm@lcs.mit.edu

Dr. Satish Rao
NEC Research Institute
4 Independence Way
Princeton, NJ 08540.
(609) 951-2714
satish@research.nj.nec.com

Prof. Thomas Gross
CMU Computer Science Dept.
5000 Forbes Avenue
Pittsburgh, PA 15213.
(412) 268-7661
Thomas.Gross@cs.cmu.edu

Summary of Research Activities

1. Thesis research: space efficient multithreading

Programs with irregular or dynamic parallelism benefit from the use of lightweight, fine-grained threads. Lightweight threads significantly simplify the programmer's job, allow for automatic load balancing, and dynamically adapt to a changing number of processors. However, the program depends heavily on the threads implementation for high performance. Therefore, unless the threads scheduler is carefully implemented, the program may end up using too much space, or suffer poor performance due to excessive memory allocation, high scheduling overheads, or poor locality.

My thesis research has focused on two aspects of thread scheduling: the design and analysis of provably space-efficient scheduling algorithms for fine-grained multithreaded programs, and the engineering and evaluation of fast, multithreading runtime systems based on these algorithms.

(a) Space-efficient scheduling algorithms

I designed two online, asynchronous scheduling algorithms for multithreading systems that support nested parallelism. The algorithms are provably space and time efficient, that is, they provide low upper bounds on the space and time requirements of a multithreaded computation. The algorithms guarantee that a nested parallel program with S_1 serial space requirement and D critical path length (depth) requires at most $S_1 + O(pD)$ space on p processors. This is a significant improvement over previous asynchronous, space-efficient schedulers that guarantee an upper bound of pS_1, since D is small compared to S_1 for most parallel programs. My algorithms maintain this low space bound by prioritizing threads according to their sequential execution order, and temporarily stalling big allocations of space. To ensure scalability for both the algorithms, I also parallelized the scheduler itself, and analyzed the space and time requirements including scheduling overheads. Both algorithms provide a user-adjustable trade-off between space and time performance which I analyze and experimentally verify. The second algorithm improves upon the first by also trying to schedule threads close in the computation graph on the same processor, to result in good locality and low scheduling contention.

In joint work with Phillip Gibbons and Yossi Matias from Bell Labs, and my advisor Guy Blelloch, we designed the first asynchronous, space-efficient scheduling algorithm for parallel languages with synchronization variables. Such languages include languages with futures such as Multilisp or Cool, as well as other languages like Id.

(b) Multithreading runtime systems

To determine how useful my space-efficient scheduling algorithms were in practice for realistic benchmarks, I implemented them in the context of two separate runtime systems. First, I built a runtime system that executed C programs, with the threads written in a continuation-passing style, on an SGI Power Challenge. The system used one of my asynchronous scheduling algorithms to schedule the threads. The results of executing parallel programs on this system show that my scheduling algorithm significantly reduces memory usage compared to previous space-efficient techniques, without compromising performance.

Although POSIX threads (Pthreads) have become a popular standard for shared memory programming, I found the current Pthread schedulers unsuited for executing programs with fine-grained, dynamic parallelism. I therefore implemented both my space-efficient scheduling algorithms as part of a popular Pthreads package on Solaris. Although the algorithms guarantee upper bounds for nested parallel programs, they also appear to work well in practice for program with arbitrary synchronizations, such as locks or condition variables. This is the first space-efficient system that supports a functionality as general as that of Pthreads. I used several parallel programs, including numerical codes, physical simulations, and a data classifier, to evaluate the schedulers. The programs, written in a fine-grained style, were simpler to code than their hand-partitioned, coarse-grained counterparts, and yet provided equivalent performance. However, my space-efficient schedulers were required in place of the Pthread library's original scheduler to achieve this performance.

2. Distributed systems

Although my thesis research has focused on parallel computing on shared memory systems, I have also worked on various aspects of distributed computing.

(a) Bulk-synchronous distributed objects

Bulk-synchronous libraries provide a simple and efficient model for parallel programming with predictable performance. However, the programmer has to manage all the data movement at a low level, similar to the message-passing style. In collaboration with Satish Rao (NEC Research Institute), I designed and implemented a distributed object layer over a bulk-synchronous library on a network of PCs. The object layer provides a global object name space that simplifies the task of programming, and uses software caching to minimize communication and programming effort. Since it is built over a bulk-synchronous library, the object system can also provide efficient and predictable performance. We have implemented a number of applications using the object layer to demonstrate its applicability and efficiency.

(b) Distributed shared memory

I worked with Chandramohan Thekkath (DEC Systems Research Center) on adding runtime optimizations to an existing distributed shared memory (DSM) library called SAM. SAM uses objects to share data between processors, and provides primitives for asynchronous access to these shared objects. Although the programming model is fairly user-friendly, performance can be slowed down by a large number of asynchronous requests for remote data. We designed runtime optimizations to reduce communication, and to overlap communication with computation. We took advantage of the fact that data access patterns for many applications change slowly over time. For example, we dynamically kept track of the set of active readers for each DSM object, so that on writes to the object, its copies in the caches of the active readers could be updated before the next read request was made. Other optimizations include modifications to the cache coherence policy depending on the object access patterns. The effectiveness of these optimizations was experimentally verified using existing parallel benchmarks written in SAM.

3. N-body algorithms

This joint work with Guy Blelloch compares three popular algorithms for the 3D N-body problem: the Barnes-Hut algorithm, Greengard's Fast Multipole Method (FMM), and the Parallel Multipole Tree Algorithm (PMTA), to determine which of the algorithms performs best in practice. Although FMM has a better asymptotic running time ($O(N)$ instead of $O(NlogN)$ for uniform distributions), the algorithm is more complicated and it is not immediately clear above what values of N it performs better in practice. This is the first research that analyzes and compares the relative performances of the three popular N-body algorithms. We first experimentally studied the dependence of accuracy on the variable parameters in the three algorithms. We then mathematically analyzed and compared the floating point operation counts of the algorithms at similar levels of accuracy, for both charged and uncharged random distributions. At a high level of accuracy, we found that the FMM performed the least number of operations for $N > 10^4$, assuming both charged and uncharged distributions of points. However, at a lower level of accuracy, for uncharged distributions, the FMM did not outperform Barnes-Hut even for $N > 10^8$. For charged distributions of particles, both the FMM and PMTA were comparable at low accuracy.

4. Compilers

(a) Compiling Nesl into Java

The SCANDAL project in CMU is involved with the implementation of a high level, applicative language called NESL. Currently, it uses a C-based vector library as its backend; porting NESL to a new platform therefore involves porting this vector library. NESL is well-suited to code algorithm animations. However, since the NESL implementation was limited to X11-based unix systems, we required an insecure X11 connection to make the animations available over the web. In joint work with Jonathan Hardwick and Jay Sipelstein at CMU, we designed and implemented an alternate system for compiling NESL into Java. In addition to increasing the portability of NESL, this system has enabled us to make existing simulations and algorithm animations available in applet form on the web. We compared the performance of the new system using a set of benchmarks on both PCs and workstations. Current Java virtual machines running the generated code achieve about half the performance of a native implementation of NESL. We found that the use of Java as an intermediate language is a viable way to improve the portability of existing high-level programming languages for scientific computation and visualization.

(b) Incremental dataflow analysis

As part of my senior thesis under the guidance of Dhananjay Dhamdhere (IIT), I worked on an algorithm for incremental dataflow analysis. The goal was to minimize the time required by the optimization phase of a compiler when small modifications are made to the program.