AFRL-SN-WP-TR-1998-1110

BLIZZARD AND PARDYN: INFRASTRUCTURE AND SCALABLE TOOLS FOR MULTI-PARADIGM PARALLEL COMPUTERS

Barton P. Miller
James R. Larus
Mark D. Hill
David A. Wood

Computer Sciences Department
University of Wisconsin-Madison
1210 W. Dayton Street
Madison, Wisconsin 53706-1685

APRIL 1998

FINAL REPORT FOR PERIOD AUGUST 30, 1994 – JANUARY 8, 1998

SENSORS DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7318

NOTICE

USING GOVERNMENT DRAWINGS, SPECIFICATIONS, OR OTHER DATA INCLUDED IN THIS DOCUMENT FOR ANY PURPOSE OTHER THAN GOVERNMENT PROCUREMENT DOES NOT IN ANY WAY OBLIGATE THE US GOVERNMENT. THE FACT THAT THE GOVERNMENT FORMULATED OR SUPPLIED THE DRAWINGS, SPECIFICATIONS, OR OTHER DATA DOES NOT LICENSE THE HOLDER OR ANY OTHER PERSON OR CORPORATION; OR CONVEY ANY RIGHTS OR PERMISSION TO MANUFACTURE, USE, OR SELL ANY PATENTED INVENTION THAT MAY RELATE TO THEM.

THIS REPORT IS RELEASABLE TO THE NATIONAL TECHNICAL INFORMATION SERVICE (NTIS). AT NTIS, IT WILL BE AVAILABLE TO THE GENERAL PUBLIC, INCLUDING FOREIGN NATIONS.

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.

JEFFREY T. BUTLER, Capt, USAF
Project Engineer
Targeting and Attack Radar Branch
Sensors Directorate

DENNIS M. MUKAI, Chief
Targeting and Attack Radar Branch
RF Sensor Technology Division
Sensors Directorate

PAUL S. HADORN, Ph.D., Chief
RF Sensor Technology Division
Sensors Directorate

Do not return copies of this report unless contractual obligations or notice on a specific document requires its return.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
April 3, 1998

3. REPORT TYPE AND DATES COVERED
Final Technical Report, August 30, 1994 - January 8, 1998

4. TITLE AND SUBTITLE
Blizzard and Paradyn: Infrastructure and Scalable Tools for Multi-Paradigm Parallel Computers

5. FUNDING NUMBERS
C: F33615-94-1-1526
PE: 62301E
PR: B550
TA: 01
WU: 01

6. AUTHOR(S)
Barton P. Miller, Mark D. Hill, James R. Larus, David A. Wood

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Computer Sciences Department
University of Wisconsin-Madison
1210 W. Dayton Street
Madison, WI 53706-1685

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Sensors Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright-Patterson Air Force Base, OH 45433-7318
POC: Capt Jeffrey T Butler, AFRL/SNRR, 937-255-6427, ext 4340

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-SN-WP-TR-1998-1110

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

The Blizzard and Paradyn project has built and distributed software infrastructure and tools for developing and studying multiple-paradigm, parallel programs on existing and future parallel computers. The Blizzard shared-memory substrate and Paradyn performance tools enable researchers to develop and evaluate parallel applications languages, compilers, and run-time systems for current and future machines. We use this infrastructure to evaluate innovative hardware ideas and influence the direction of future machines.

Tempest is an interface between user-level software and a system on which to implement communication, including transparent shared memory, extensible shared memory, message passing, and hybrid forms. Applications or compilers are not limited to a single hardware-provided paradigm. Blizzard is an implementation of the Tempest interface for existing message-passing machines (CM-5 and Wisconsin COW, a cluster of 40 dual-processor SPARCs). This software-only solution performs well enough that Blizzard programs compete with native software.

Paradyn is based on a new, inherently scalable dynamic instrumentation facility for detailed profiling of long-running, very large parallel programs. Paradyn monitors instrumentation cost and automatically controls its dynamic instrumentation. Paradyn supports an interface for high-level parallel languages presenting performance data at the language level, allowing a programmer to peel back layers (such as Blizzard, run-time, or parallel math libraries) to understand low-level operations in terms of high-level features.

14. SUBJECT TERMS
parallel computing, scalable systems, performance profiling, high-performance architectures

15. NUMBER OF PAGES 11

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

BLIZZARD AND PARDYN: INFRASTRUCTURE AND SCALABLE TOOLS FOR MULTI-PARADIGM PARALLEL COMPUTERS

Barton P. Miller
Mark D. Hill
James R. Larus
David A. Wood

Computer Sciences Department
University of Wisconsin-Madison
1210 W. Dayton Street
Madison, Wisconsin 53706-1685

APRIL 1998

FINAL REPORT FOR PERIOD AUGUST 30, 1994 – JANUARY 8, 1998

SENSORS DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7318

1 OVERVIEW

The Blizzard and Paradyn project investigated next-generation high-performance tools and architectures. Overall, the project met or exceeded it goals, creating a huge wealth of technical results, publications, and widely-distributed software. In both the architecture and tools area, this project has defined the cutting edge and has had global influence on high-performance computing. Some of the technical highlights of this work include:

- Demonstration and deployment of platform-independent, run-time, instruction level instrumentation (Paradyn's Dynamic Instrumentation).
- Demonstration and deployment of techniques that automate the diagnosis of performance bottlenecks (Paradyn's Performance Consultant).
- Demonstration of a portable fine-grain distributed shared memory system (Blizzard).
- Development and deployment of an executable editing library (EEL).

2 ACCOMPLISHMENTS

We report on major accomplishments performed under this contract. We proposed contributions in the following areas (these items are taken verbatim from our proposal). We were successful at, or above, the proposal levels in all of these areas. Each area is annotated with the publication numbers listed in Section 3.

- **A user-level shared-memory abstraction.** The Tempest shared-memory interface will provide user-level code with control over synchronization, communication, and memory management. Because this code is user accessible and modifiable, a compiler can select the explicit message passing, transparent shared memory, or customized memory semantics appropriate for an application. In particular, this interface will facilitate development of compilers and run-time libraries for irregularly structured and other dynamic problems. [3,21,25,27,34,37,43,47]
- **A shared-memory substrate on current message-passing machines.** The Blizzard shared-memory substrate enables researchers to develop Tempest shared-memory programs and compilers on current message-passing machines (CM-5 and Wisconsin COW). Blizzard's transparent shared-memory performance is within a small constant factor of machines with hardware shared-memory support. [17,20,32,33,44,45]
- **Fast, accurate simulation of future parallel machines.** We extended the Wisconsin Wind Tunnel virtual prototyping system to provide a Tempest user-level shared memory interface. This system enables researchers to quickly and accurately evaluate a program's performance on future large-scale parallel machines and provide concrete data to influence the design of these machines. [12,22,30,36]
- **Evaluation of large-scale machines.** Because the Wisconsin Wind Tunnel can multiplex multiple target nodes for each host (i.e., CM-5) node, researchers can evaluate their codes on "virtual machines" that are much larger than the available physical machines. [7,11,13,14,22,23,30]
- **On-the-fly dynamic instrumentation.** This new method of instrumentation is a major advance in performance monitoring that scales to very large parallel machines. Instrumentation decisions are delayed until program execution. Instrumentation is only added when necessary, which eliminates unnecessary overhead for unused instrumentation. The Executable Editing Library (EEL) enables further instrumentation optimizations. Dynamic instrumentation eases the work of the programmer since no special code, compilation, or linking is needed. Existing programs can be measured immediately; continually-running programs (such as servers or large computations) can be measured at any time in their execution. [2,6,16,19,26,35]
- **Decision support for performance measurement.** Decision support, as embodied in the Paradyn's Performance Consultant, provides systematic and automated assistance in searching for performance bottlenecks. It can dynamically select the instrumentation to insert in an application program and system, evaluate the data being collected, and further focus and refine the instrumentation. The Performance Consultant guides a programmer in the search for performance bottlenecks. It also selects visualizations to illustrate current performance problems. [1,15]
- **Integrated application/system/hardware instrumentation.** The dynamic instrumentation provides the ability to incorporate new performance data from the hardware (e.g., bus utilization or switch traffic) and operating system (e.g., paging rate or disk utilization). New performance metrics are specified by a simple description.

Together with the Wisconsin Wind Tunnel, this capability helps evaluate hardware instrumentation techniques. [10,18,19,24]

- **Model for high-level parallel language performance.** We developed a model to describe performance data at multiple levels of abstraction, while maintaining the mappings between the levels. Programmers view performance data in terms of the high-level source code, *and* peel back layers of abstraction to see the underlying activities. Implicit (compiler or run time generated) costs can be exposed by viewing performance data at more primitive levels. The mapping information allows a programmer to associate low-level performance activity (e.g., node message passing) with a high-level language construct (e.g., an array distribution in HPF). [29,28,46]

These results are based on the deliverables specified in our proposal. These deliverables are in relation to two proposed computing platforms, the TMC CM-5 and "Machine-X". Machine-X was a machine to be designated during the course of the contract, to allow the tracking of current technologies. This machine turned out to be the Wisconsin COW (Cluster of Workstations), a 40 node cluster of dual-processor SPARC nodes connected via Myrinet, and 10 and 100 Mb/second Ethernet. We also went beyond our deliverables and had some success porting to a "Machine Y," the IBM SP/2.

- *Specification of the Tempest user-level shared memory interface:* The *Tempest* specification is complete. It provides user-level code with control over synchronization, communication, and memory management. Primitives allow the sending and receiving of active messages, the sending and receiving of bulk data transfers, limit user-level control of virtual memory mapping (e.g, like Unix's *mmap*), and fine-grain access control. Fine-grain access control, arguably the most novel aspect of Tempest, provides users with the ability to control access at a granularity similar to a cache block (e.g., 64 bytes).
- *User-level shared memory library:* We implemented this library to enable programs written to "transparent shared memory" (e.g., to a symmetric multiprocessor) to run on Tempest on a variety of non-shared-memory platforms. This library, called *"Stache"* for *"SofTware cACHE,"* caches data from remote nodes by allocating local pages, mapping them to the appropriate virtual addresses, and moving data in and out at the cache block granularity. Like S-COMA systems, and unlike most software DSMs, Stache provides a performance model similar to the ubiquitous hardware symmetric multiprocessor (e.g., false sharing occurs only within cache blocks).
- *Blizzard/CM-5:* We implemented Tempest on the TMC CM-5 in an implementation called *Blizzard/CM-5*. With Stache running on Blizzard/CM-5, users could run shared memory programs on the non-shared memory CM-5 for the first time. We implemented fine-grain access control with either executable editing (see EEL below) or by overloading error correcting code (ECC) bits to generate invalid traps.
- *Blizzard/Machine-X:* We implemented Tempest on the Wisconsin COW in an implementation called *Blizzard/COW.* With Stache running on Blizzard/COW, users could run shared memory programs on the non-shared memory COW. To the best of our knowledge, Blizzard/COW is the first system to allow users to run shared memory programs on a COW with a performance model similar to the ubiquitous hardware symmetric multiprocessor. We implemented fine-grain access control with executable editing, overloading ECC bits, or with a custom board designed as part of a National Science Foundation grant.
- *Blizzard/WWT:* We implemented Tempest of the Wisconsin Wind Tunnel (WWT) parallel simulator. WWT runs a parallel target program on a parallel host and uses execution-driven, distributed, discrete-event simulation to accurately calculate program execution time for the target machine under evaluation. This implementation of Blizzard, dubbed *Blizzard/WWT,* allowed us to evaluate future designs for supporting Tempest.
- *Executable Editing Library (EEL):* We delivered EEL for the CM-5 and Machine-X (Wisconsin COW). We use EEL in some implementations of Blizzard and WWT. We also went beyond our deliverables to develop/port EEL to a UltraSPARC symmetric multiprocessor and an IBM SP/2. The latter was a substantial change due to the instruction set differences between the SP/2 and other machines on which EEL runs.
- *Dynamic instrumentation on the TMC CM-5:* Dynamic Instrumentation is one of the fundamental technologies in the Paradyn performance tools. It provides the ability to instrumentation programs on the fly, requiring no special preparation. Already running programs (such as servers) can easily be instrumented. Dynamic instrumentation provides one of the major techniques for scalable instrumentation because it allows you to be able to defer instrumentation decisions until after execution, asking only for what is relevant to the current study. It was implemented for the CM-5 early in the project and demonstrated in a research exhibit at the Supercomputing '94 Conference. Dynamic Instrumentation on the CM-5 demonstrated the scalability features harnessing the CM-5

broadcast and reduction networks.

- *Dynamic instrumentation on Machine-X:* Dynamic Instrumentation was implemented on the Wisconsin COW and used in several major application studies. This implementation leveraged the SPARC dynamic code generation from the CM-5 implementation. Beyond the proposed goals, Dynamic Instrumentation was also implemented for the Intel x86 architecture under both Solaris and Windows/NT, HP PA-RISC under HP-UX, and IBM Power2 under AIX for the SP2.
- *User-level shared memory instrumentation:* The Dynamic Instrumentation and the Blizzard user-level coherence protocols were successfully integrated to provide detailed performance data from the Blizzard memory system. Communications, cache miss activities, protocol states, and memory block performance data were directly accessible from the Paradyn performance tool.
- *Performance Consultant:* The Performance Consultant is the part of Paradyn that automates the search for performance bottlenecks. It was originally prototyped in 1994 and, during the course of this contract, significant re-engineered and extended in power and functionality. It was extended to be able to concurrently search for different kinds of performance bottlenecks simultaneously, provide feedback about the projected and actual cost of instrumentation. New display techniques were developed so that very large programs (thousands of modules and 10's of thousands of functions) could easily be navigated. Beyond the proposed goals, the Performance Consultant is also available on all the platforms mentioned for Dynamic Instrumentation (except Windows/NT; that version is currently under development).
- Generic high-level language support: There are two key issues for supporting high-level parallel languages. The first issue is to handle the high-level program constructs and operations (such as parallel loops, broadcasts and reductions). The second issue is to be able to view performance data in terms of data structures (instead of control structures). Since data-parallel programming is a fundamental idiom, view performance in terms of parallel data structures (distributed arrays) or parts of these data structures is crucial. Support for both of these features was incorporated into a research version of Paradyn.
- High-level language support for CM: Based on the generic high-level language support in Paradyn, support for CM Fortran (TMC's data parallel Fortran) was developed. We were able to measure applications, and conducted several application studies, including large applications from NASA Ames Research Center. In all cases, we were able to make substantial performance improvements (even on production codes).
- High-level language support for HPF or a Parallel C++: HPF compilers were much later to market than predicted by the industry. The compilers that were produced were buggy and generate code with disappointing performance. As a result, there were few application groups using HPF during the duration of our contract. We switched emphasis in this area to another major programming idiom: distributed message passing. We built support for PVM (on all Paradyn platforms) and MPI (on the SP2). This support has been a major success with many applications being profiled.
- Open visualization interface: Paradyn's open visualization interface and library allows easy access to Paradyn performance data by external visualization tools. Paradyn provides, standard with its distribution, table, bar chart, time histogram, and 3D terrain plots. We also built a general purpose translator module so that we could feed Paradyn performance data directly into the DeVise visualization tools.
- All technical reports, published papers, theses, and dissertations.
- A complete description of all software, including source listings and internal design documents: Two CD ROMs that include the full Blizzard and Paradyn distributions are included with this report.

3 PUBLICATIONS

3.1 Book Chapters

[1] J.K. Hollingsworth and B.P. Miller, "Measurement and Instrumentation" in **Computational Grids: The Future of High-Performance Distributed Computing,** Morgan-Kaufman, I. Foster and C. Kesselman, eds., 1998.

[2] K. Kunchithapadam and B.P. Miller, "Integrating a Debugger and a Performance Tool for Steering" in **Debug-**

ging and Performance Tools for Parallel Computing Systems, IEEE Computer Society Press, M.L Simmons, A.H. Hayes, J.S. Brown, and D.A. Reed, eds., pp. 53-64, 1996.

[3] J. R. Larus, B. Richards, and G. Viswanathan. Parallel programming in C**: A large-grain data-parallel programming language. G. V. Wilson and P. Lu, editors, In **Parallel Programming Using C++**, pages 297–342. MITP, 1996.

[4] J. Hollingsworth, B.P. Miller, and J.E. Lumpp, "Techniques for Performance Measurement of Parallel Programs" in **Parallel Computers: Theory and Practice** IEEE Computer Society Press, T.L. Casavant, P. Tvrdik, and F. Plasil, eds., 1995.

3.2 Journal Papers

[5] M. D. Hill. Multiprocessors should support simple memory consistency models. *IEEE Computer*, 31, 1998.

[6] J.K Hollingsworth and B.P. Miller, "Using Cost to Control Instrumentation Overhead", *Theoretical Computer Science* (to appear, 1998). Invited paper.

[7] S. S. Mukherjee and M. D. Hill. Making network interfaces less peripheral. *IEEE Computer*, 1998. A talk-only version of this paper appears in *Hot Interconnects V*, 1997.

[8] K.L. Karavanic, J. Myllymaki, M. Livny and B.P. Miller, "Integrated Visualization of Parallel Program Performance Data", *Parallel Computing* **23**, 1-2 (1997), pp. 181-198. Special issue environments and tools for parallel scientific computing.

[9] B. Falsafi and D. A. Wood, "Modeling cost/performance of a parallel computer simulator", *ACM Transactions on Modeling and Computer Simulation 7*, 1, January 1997.

3.3 Conference Papers

[10] G. Ammons and J. R. Larus. Improving data-flow analysis with path profiles. *SIGPLAN '98 Conference on Programming Language Design and Implementation (PLDI)*, June 1998.

[11] M. Plakal, D. J. Sorin, A. E. Condon, and M. D. Hill, "Lamport Clocks: Verifying a Directory Cache-Coherence Protocol", *10th Annual ACM Symposium on Parallel Architectures and Algorithms*, June 1998.

[12] D. Sorin, V. Pai, S. Adve, M. Vernon, and D. Wood. Analytic Evaluation of Shared-Memory Parallel Systems with ILP Processors. *25th International Symposium on Computer Architecture*, June 1998.

[13] S. S. Mukherjee and M.D. Hill. The impact of data transfer and buffering alternatives on network interface design. *Fourth IEEE Symposium on High-Performance Computer Architecture*, February 1998.

[14] I. Schoinas and M.D. Hill. Address translation mechanisms in network interfaces. *Fourth IEEE Symposium on High-Performance Computer Architecture*, February 1998.

[15] K.L. Karavanic and B.P. Miller, "Experiment Management Support for Performance Tuning", *Supercomputing '97*, San Jose, California, November 1997.

[16] J.K. Hollingsworth, B.P. Miller, M.J.R. Gonçalves, O. Naim, Z. Xu, and L. Zheng, "MDL: A Language and Compiler for Dynamic Program Instrumentation", *1997 International Conference on Parallel Architectures and Compilation Techniques*, November 1997, San Francisco, California.

[17] S. Chandra, M. Dahlin, B. Richards, R. Y. Wang, T. E. Anderson, and J. R. Larus. Experience with a language for writing coherence protocols. *USENIX Conference on Domain-Specific Languages*, Santa Barbara, California, October 1997.

[18] G. Ammons, T. Ball, and J.R. Larus. Exploiting hardware performance counters with flow and context sensitive profiling. *SIGPLAN '97 Conference on Programming Language Design and Implementation (PLDI)*, June 1997.

[19] Z. Xu, J.R. Larus, and B.P. Miller, "Shared-Memory Performance Profiling", *6th ACM Symposium on Principles and Practice of Parallel Programming*, Las Vegas, June 1997.

[20] Y. Zhou, L. Iftode, J. P. Singh, K. Li, B. R. Toonen, I. Schoinas, M. D. Hill, and D. A. Wood, "Relaxed consis-

tency and coherence granularity in DSM systems: a performance evaluation", *Sixth ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*, June 1997.

[21] S. Chandra and J. R. Larus. Optimizing Communication in HPF programs for Fine-Grain Distributed Shared Memory. *6th ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming*, Las Vegas, June 1997.

[22] Babak Falsafi and David A. Wood. "Reactive NUMA: A Design for Unifying S-COMA and CC-NUMA". *24th Annual International Symposium on Computer Architecture*, pages 229–240, June 1997.

[23] B. Falsafi and D. A. Wood. Scheduling communication on an SMP node parallel machine. *Third IEEE Symposium on High-Performance Computer Architecture*, pages 128–138, February 1997.

[24] T. Ball and J. R. Larus. Efficient path profiling. *29th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 29)*, page 46–57, December 1996.

[25] Guhan Viswanathan and James R. Larus. "Compiler-directed Shared-Memory Communication for Iterative Parallel Applications". *Supercomputing '96*, November 1996.

[26] J.K Hollingsworth and B.P. Miller, "An Adaptive Cost Model for Parallel Program Instrumentation", *Euro-Par '96*, Lyon, France, August 1996.

[27] S. Chandra and J. R. Larus. HPF on Fine-Grain Distributed Shared Memory: Early Experience. U. Banerjee, A. Nicolau, D. Gelernter, and D. Padua, editors, *Ninth Workshop on Languages and Compilers for Parallel Computing*. August 1996.

[28] R.B. Irvin and B.P. Miller, "Mechanisms for Mapping High-Level Parallel Performance Data", 1996 *Workshop on Challenges for Parallel Processing*, IEEE Computer Society Press (Chicago, August 1996).

[29] R.B. Irvin and B.P. Miller, "Mapping Performance Data for High-Level and Data Views of Parallel Program Performance", *Int'l Conference on Supercomputing*, Philadelphia, May 1996.

[30] Steven K. Reinhardt, Robert W. Pfile, and David A. Wood. Decoupled Hardware Support for Distributed Shared Memory. *23rd Annual International Symposium on Computer Architecture*, May 1996.

[31] Rahmat S. Hyder and David A. Wood. Synchronization Hardware for Networks of Workstations: Performance vs. Cost. *1996 International Conference on Supercomputing*, May 1996.

[32] Satish Chandra, Brad Richards, and James R. Larus. Teapot: Language Support for Writing Memory Coherence Protocols. *SIGPLAN '96 Conference on Programming Language Design and Implementation (PLDI)*, May 1996.

[33] Trishul Chilimbi, Thomas Ball, Stephen Eick, and James Larus. StormWatch: A Tool for Visualizing Memory System Protocols. *Proceedings of Supercomputing '95*, December 1995.

[34] S. S. Mukherjee, S. D. Sharma, M. D. Hill, J. R. Larus, A. Rogers, and J. Saltz. Efficient support for irregular applications on distributed-memory machines. *Fifth ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming* (PPOPP), pages 68–79, July 1995.

[35] J. R. Larus and E. Schnarr. Eel: Machine-independent executable editing. *SIGPLAN '95 Conference on Programming Language Design and Implementation (PLDI)*, pages 291–300, June 1995.

[36] D. C. Burger and D. A. Wood. Accuracy vs. performance in parallel simulation of interconnection networks. *9th International Parallel Processing Symposium*, April 1995.

[37] M.D. Hill, J.R. Larus, and David A. Wood. Tempest: A Substrate for Portable Parallel Programs. *COMPCON '95*, pages 327–332, San Francisco, California, March 1995. IEEE Computer Society.

[38] A.R. Lebeck and D.A. Wood. Active memory: A new abstraction for memory-system simulation. 1995 *ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 220–230, May 1995.

3.4 Workshops

[39] Shubhendu S. Mukherjee, Steven K. Reinhardt, Babak Falsafi, Mike Litzkow, Steve Huss-Lederman, Mark D. Hill, James R. Larus, and David A. Wood. Wisconsin Wind Tunnel II: A Fast and Portable Parallel Architecture Simulator. *Workshop on Performance Analysis and Its Impact on Design (PAID)*, June 1997.

[40] S. Chandrasekaran and M. D. Hill. Optimistic simulation of parallel architectures using program executables. *Proceedings of Tenth Workshop on Parallel and Distributed Simulation (PADS '96)*, May 1996.

[41] Eric Schnarr and James R. Larus. Instruction Scheduling and Executable Editing. *Workshop on Compiler Support for System Software (WCSSS '96)*, Tucson, Arizona, February 1996.

3.5 Theses and Technical Reports

[42] D. J. Sorin, M. Plakal, M. D. Hill, and A. E. Condon. Lamport clocks: Reasoning About Shared Memory Correctness. Technical Report 1367, Computer Sciences Department, University of Wisconsin–Madison, 1998.

[43] S. Chandra, December 1997, *Ph.D. Thesis*, "Software Techniques for Customizable Distributed Shared Memory".

[44] B. Falsafi, December 1997, *Ph.D. Thesis*, "Fine-grain Protocol Execution Mechanisms and Scheduling Policies".

[45] I. Schoinas, December 1997, *Ph.D. Thesis*, "Fine-grain Distributed Shared Memory on Clusters of Workstations".

[46] R.B. Irvin, October 1995, *Ph.D. Thesis*, "Performance Measurement Tools for High-Level Parallel Programming Languages".

[47] Steven K. Reinhardt. Tempest Interface Specification (Revision 1.2.1). *Technical Report 1267*, Computer Sciences Department, University of Wisconsin–Madison, February 1995.

4 PERSONNEL

4.1 Faculty

1. Barton P. Miller, Professor
2. Mark D. Hill, Associate Professor
3. James R. Larus, Associate Professor
4. David A. Wood, Associate Professor
5. Pei Cao, Assistant Professor
6. Steven Huss-Lederman, Associate Scientist

4.2 Research Staff

1. Michael Litzkow (M.S., University of Wisconsin)
2. Marcelo Gonçalves (Ph.D., Princeton University)
3. Oscar Naim (Ph.D., University of Southampton, England)
4. Brian Wylie (Ph.D., University of Edinburgh, Scotland)

4.3 Graduate Students

4.3.1 Ph.D. Students (completed)

1. Satish Chandra, Ph.D., December 1997, "Software Techniques for Customizable Distributed Shared Memory".
2. Babak Falsafi, Ph.D., December 1997, "Fine-grain Protocol Execution Mechanisms and Scheduling Policies".
3. Ioannis Schoinas, Ph.D., December 1997, "Fine-grain Distributed Shared Memory on Clusters of Workstations".

4.3.2 Ph.D. Students (passed Preliminary Exam - ABD)

4. Trishul Chilimbi, "Cache-conscious Data Placement for Pointer-Based Codes", July 1997.
5. Karen Karavanic, "Experiment Management Support for Performance Tuning", August 1997.
6. Eric Schnarr, "Fast Microarchitecture Simulation Using Run-Time Code Generation", October 1997.
7. Krishna Kunchithapadam, "Improving the Instruction-Cache Performance of Large Applications", May 1995.

4.3.3 M.S. Students (completed)

8. Sashik Chandrashenkaran, M.S., December 1994.
9. Rahmat Hyder, M.S. May 1993.
10. Chengjie Liu, M.S., December 1996.
11. Christopher Lukas, M.S., December 1996.
12. Shankar Pasupathy, M.S., May 1998.
13. Robert Pfile, M.S., August 1995.

4.3.4 Pre-Exam Students

14. Ariel Tamches.
15. Zhichen Xu.
16. Chun Zhang.

4.4 Undergraduate Students

1. Tung Fai Chan
2. Daniel Nash
3. Russell Weisfield

5 PRESENTATIONS

Held semiannual two-day workshops on Tempest/Blizzard that were attended by representatives from DEC, HP, IBM, Intel, SGI, Sun, and Thinking Machines.

Held annual "Paradyn Week" meetings to summarize the current year's work and results. Attended by industry groups (Sun, SGI, IBM, Microsoft, Intel), government labs (LANL, LLNL, Sandia, Argonne), and various U.S. and foreign university researchers.

Tempest: A Substrate for Portable Parallel Programs
- 11/97 Northwestern University
- 4/96 U.C. Berkeley
- 3/96 DEC SRC
- 2/96 U.C. San Diego
- 11/95 University of Wisconsin--Milwaukee
- 9/95 University of Michigan
- 8/95 Massachusetts Institute of Technology
- 8/95 Stanford University
- 6/95 Fujitsu
- 6/95 Japanese Joint Symposium on Parallel Processing (invited talk)
- 6/95 University of Washington
- 4/95 University of Massachusetts

4/95 Harvard University
3/95 University of Arizona
11/94 Georgia Tech.
11/94 Princeton University
11/94 AT&T Bell Labs, Murray Hill
10/94 Sun Microsystems
9/94 IBM Yorktown
9/94 CMU
9/94 University of Colorado

Beyond CC-NUMA: Whither Shared Memory?
6/97, SPAA '97,

Experience with Building Parallel Systems
4/97 IPPS/PARCON '97,

HPCA'97 Panel on Hardware vs. Software support for Distributed Shared Memory, February 1997.

Panelist, ICCD'96 Panel on Hardware support for Distributed Shared Memory, October 1996.

Panelist, 2nd NOW/Cluster workshop, with ASPLOS VII, Panel on Communication Models, October 1996.

Panelist, NSF Workshop on Critical Issues in Computer Architecture Research, May 1996.

LCM: Memory System Support for Parallel Language Implementation
Parallel Object Oriented Methods Applications (POOMA), Santa Fe, New Mexico, December 1994.

Efficient Path Profiling
Silicon Graphics, May 1997.
University of California, Berkeley, May 1997.
Sun Microsystems, April 1997.
University of California, San Diego, April 1997.
University of Toronto and IBM Toronto, March 1997.
Microsoft Research Laboratory, February 1997.
Hewlett Packard Research Laboratory, August 1996.

EEL: Machine-Independent Executable Editing
Intel Corporation, September 1996.
Princeton University, April 1996.
Sun Microsystems, December 1995.
Microsoft Research, June 1995.

The Paradyn Parallel Performance Tools
Georgia Tech, Atlanta, March 1997.
University of Kentucky, Lexington, September 1997.
UCLA Computer Science Department, January 1997.
Silicon Graphics, February 1995.
Stanford University, April 1995.
University of Colorado, Boulder, April 1995.
Cal Tech Center for Research in Parallel Computing, April 1995.
Hebrew University, Jerusalem, June 1996.

IBM Haifa Research Center, Haifa, June 1996.

Performance Measurement for High-Level Parallel Languages
IBM Haifa Research Center, Haifa, June 1996.

Panelist, Workshop on Debugging and Performance Tuning for Parallel Computing Systems, Cap Cod, October 1994.

Integrating Steering and Performance Measurement
Invited talk, Workshop on Debugging and Perf. Tuning for Parallel Computing Systems, Cap Cod, October 1994

Evaluation of Complex Systems
ARPA CSS PI Meeting, San Antonio TX, February 1996.

Performance Tuning as Experiment Management
Invited lecture, Gordon Research Conference, Plymouth NH, July 1995.

Making Real Programs Explode": A Simple Application of Random Testing
Tuskegee University, Tuskegee Alabama, March 1997.
IBM Haifa Research Center, Haifa, June 1996.
University of California, San Diego, June 1995.
Silicon Graphics, June 1995.
DEC Western Research Lab, May 1995.
Sun Microsystems, April 1995.
Microsoft Labs, April 1995.
IBM Almaden Research Lab, April 1995.
Stanford University (EE380), April 1995.

Super Symbol Tables: Breaking the Parallel Tools Log-Jam
Invited Talk, Workshop for Challenges for Parallel Processing, Chicago, July 1996.

User-Specifiable, Extensible, Dynamic Program Instrumentation
Invited Talk, Workshop on Environments and Tools for Parallel Computing, Lyon, France, August 1996.

W(h)ither Goes Parallel Debugging
Invited Talk, Workshop on Software Tools for High Performance Computing Systems, Cape Cod, October 1996.

User-Specifiable, Extensible, Dynamic Program Instrumentation
Intel, Santa Clara, November 1996.
Sun Microsystems, Mountain View, November 1996.

Paradyn Parallel Performance Tools: Dynamic Instrumentation on the Kernel
DARPA Quorum PI Meeting, Dallas Texas, December 1996.

Application Instrumentation
DARPA PI meeting, Washington DC, June 1997.

6 CONSULTATIVE ACTIVITIES

- Work on Tempest/Blizzard are strongly influenced parallel machine designs of IBM and Sun. We cannot elaborate due the proprietary nature of these unannounced efforts.

- EEL has been licensed by scores of universities and several companies.

7 DISCOVERIES AND INVENTIONS

1) Title: Discontiguous subblock prefetching with valid bit vectors
UW-Madison log #719
Disclosure #P98152US
Inventors: Douglas C. Burger and David A. Wood
Status: disclosed
This invention was also reported to the National Science Foundation.

2) Title: Device interface using cache transfers
UW-Madison Log #333
Disclosure #96105US
Inventors: David Wood, Steven Reinhardt, Shubhendu Mukherjee, Babak Falsafi, Mark Hill, and Robert Pfile
Status: US Patent pending

3) Title: Cost-Aware WWW Caching Algorithms GD-Size(1)
UW-Madison log #587
Disclosure #P97138US
Inventors: Pei Cao and Sandy Irani (UC Irvine)
Status: disclosed

4) Title: Cost-Aware WWW Caching Algorithms GD-Size(HOPS)
UW-Madison log #604
Disclosure #P97139US
Inventors: Pei Cao and Sandy Irani (UC Irvine)
Status: disclosed

5) Title: "Self-Sorting Notebook" Interface to Complex High-Dimensional Data
UW-Madison log #690
Disclosure #P98114US
Inventors: Barton Miller and Karen Karavanic
Status: disclosed, under review for patent filing
This invention was also reported to DOE and NASA.