DOE/ER/25001--T5

DE91 005377

CEDAR PROJECT — ORIGINAL GOALS AND PROGRESS TO DATE

Progress Report

For Period January 25, 1990 - January 24, 1991

G. Cybenko, D. Kuck, D. Padua and A. Sameh

Center for Supercomputing Research and Development
University of Illinois at Urbana-Champaign
305 Talbot Lab
104 South Wright Street
Urbana, IL 61801

November 28, 1990

Prepared for

THE U. S. DEPARTMENT OF ENERGY
AGREEMENT NO. DE-FG02-82ER25001

This work encompasses a broad attack on high speed parallel processing. Hardware, software, applications development, and performance evaluation and visualization as well as research topics are proposed. Our goal is to develop practical parallel processing for the 1990's.

Cedar Project — Original Goals and Progress to Date

The Center for Supercomputing Research and Development was founded in late 1984 with major funding beginning in January of 1985. The principal project at that time was the Cedar Project, which was designed to be the focal point for bringing to fruition our work on high performance computation. The goal of this project, as stated in a 1984 report [CSRD464*], was to demonstrate that supercomputers of the future can exhibit general–purpose behavior and be easy to use. The Cedar Project was based on five key developments which taken together offered a comprehensive solution to achieving high performance computation:

1. The development of VLSI components made large memories and small, fast processors available at low cost.
2. Based on many years of work at Illinois and elsewhere, we had designed a shared memory and switch which provide high bandwidth over a wide range of computations and applications areas.
3. The Parafrase project at Illinois had for more than 10 years been aimed at developing software for restructuring ordinary programs to effectively exploit supercomputer architectures. This technology was ripe for a real test.
4. By using a hierarchy of control, we believed that dataflow principles could be used at a high level (macro–dataflow), thus avoiding some of the problems with traditional dataflow methods.
5. Work in numerical algorithms indicated great promise in exploiting multiprocessors without the penalty of high synchronization overhead, which had proved fatal in some earlier studies.

We believe that with the developments over the past six years we have made substantial progress toward our goal. In this section we will outline our progress since the beginning of the Cedar Project and briefly describe specific Cedar work that remains to be done.

CSRD Developed Boards and Physical Equipment

The Cedar hardware (see Figure 1) has been operative for the past two years, and 16 processors have been installed and are functional. The debugging and integration of Cedar has continued until Q4 1990, and we are currently installing all 32 of the intended processors. Continued maintenance will be required for as long as the Cedar system is useful.

We have developed prototype performance evaluation boards, and we are planning to manufacture and integrate a comprehensive set of these boards.

A limited amount of new development will be done on Cedar. For example, the memory system is being improved by building a compatible global memory board using 4 MBit parts. These projects are described in more detail elsewhere in this report.

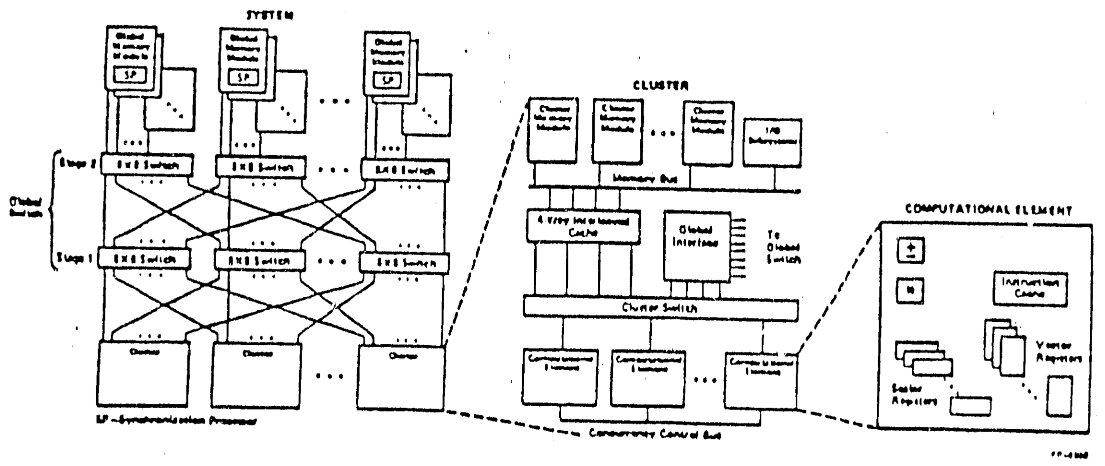


Figure 1. Architectures of the Cedar I

Boards and equipment developed at CSRD for Cedar can be classified into five categories: global hardware boards, support boards, performance boards, diagnostic hardware, and physical equipment.

Global Hardware Boards

The global hardware boards are the main boards for Cedar operation. The Global Interface Board (GIB) provides the interface from the Alliant FX/8 to the Global Network (GN). This board is 20 by 24 inches and has approximately 1,000 equivalent integrated circuits with 11 layers. The GIB is fabricated mainly in TTL technology to gain the most in functionality; however, its Global Network interface is designed in ECL. The Alliant backplane and computational elements (CEs) were modified for CSRD by Alliant to accept this board. Since the Alliant backplane was modified for four additional slots, four GIBs can be loaded into each Alliant (cluster), giving four CEs access to GN.

In addition to providing the interface to the global memory, the GIB provides Cedar-specific functions. The GIB implements the special Cedar synchronization primitives. This board also performs prefetching of vectors from Global Memory at an issue rate of one request every other clock. Cross Processor Interrupt (CPI) between clusters is also designed into the GIB.

An enhanced GIB, the GIB1A, was also designed for Cedar 1A, and its functionality is much like that of the GIB. This board has enhancements that allow for eight GIB1As to exist in each cluster, providing a global hardware interface for each CE in the cluster. Since early 1989, Cedar 1A has been operating using GIB1As.

On the GN interface side of the board, the GIB can issue requests into the network at the rate of one every two clocks. This matches the bandwidth of the Alliant issue rate. On the receiving side, the GIB has a four-deep FIFO for receiving return requests from GN. These requests can be routed by the GIB to the Alliant, the prefetch data buffer, the synchronization processing unit, or the CPI logic, whichever is appropriate.

The next set of board types constitute the GN. This expandable shuffle network is composed of two board types GNBI and GNBK. Both boards are on 20 by 24-inch form factors, and GNBI has approximately 1,150 ICs and 17 layers, while GNBK has 600 ICs and 8 layers.

GNBI is the main board of the set. This board is essentially an 8 x 8 crossbar packet switch, and each port is 80 bits wide. GNBI is designed using ECL technology to provide high bandwidth. In each clock cycle, GNBI can accept a word on all input ports, resolve conflicts, route them, and present the results to the output ports.

GNBI is made possible with a CSRD-designed ECL gate array that does most of the switch function and data queuing. The gate array is a four-bit 8 x 8 crossbar connection matrix with input and output queuing on each port. Nineteen of these gate arrays exist on each GNBI. The control for this switch uses MSI circuitry and handles contention resolution as well as providing queuing control of incoming data. Each port has a two-deep input queue and an output latch. The board has 1280 differential ECL signals presented to it via cable and provides 640 single-ended ECL signals for backplane driving to the next board, GNBK.

GNBK receives the switched data from GNBI via backplane connections. This board provides one level of queuing (latch) and converts the single-ended signals from the backplane to differential signals for cable driving to the next GN pair, GM, or GIB.

The GMB contains main memory and implements Cedar synchronization. The board is identical in size to the GIB and network boards and has approximately 1200 ICs with 15 layers. All paths on the board are 64-bits wide. It is designed with a mixture of ECL and TTL technology to provide the best compromise between speed and functionality. GMB has 2 Mbytes of RAM, and ECC protection with a 64-bit word size. This board also contains the synchronization processor that completes (together with the GIB) the implementation of the special Cedar synchronization primitives. GMB has three levels of input buffering and three levels of output buffering. It can sustain a 64-bit data streaming rate of one datum in four clocks. The latency is six clocks, a read-modify-write cycle takes eight clocks, and the synchronization processor adds three clocks for performing 32-bit operations. Normally, there is one GMB for every GIB. By reconfiguring the network, this ratio can be modified.

The memory system of Cedar is being improved by a compatible global memory board that uses 4 MBit DRAM chips instead of the 256 KBit chip which will permit the expansion of a single global memory board from 2 MBytes to up to 32 MBytes. The global memory board has significant control- and data-path board-space overhead, so that it makes economic sense to use 4 MBit DRAMS instead of 1 MBit, even if the cost ratio (of 4MBit devices) is as much as two to four times higher. This will extend the lifetime of the system for interesting applications work by allowing larger problems to be run. In addition, if board space permits we would like to restore synchronization functionality that had to be deleted in the first generation global memory board for space reasons, which will allow experimentation with a richer set of synchronization techniques than is now possible. We feel that this is a significant research area for the future in which experimental work could be quite important.

In order to bring these boards together, CSRD-designed backplanes have been produced. There is a backplane for GN that allows GNBI and GNBK to communicate with each other, and which supplies the 600+ amps of power required. Similarly, there is a backplane for GMB that provides the power as well as diagnostic interfaces and cable attachments.

Another Cedar system board is the global clock board, which provides clocks to all global hardware boards as well as to each cluster. The clock board also provides clock synchronization to each cluster to keep all clusters in lock-step.

To allow for eight GIB1A boards to exist in the cluster, CSRD has redesigned the Alliant backplane. This "stretched" backplane was designed by using the Alliant backplane database, adding connectors, and lengthening the backplane using the Calay CAD system.

Support Boards

There have been myriad support boards developed at CSRD. To test hardware, there is a GIB emulator board that emulates the memory and network for GIB debugging and diagnostics. The network board pairs required two special boards that, along with an IBM PC/AT, form a network testbed that allows testing of the network boards under very controlled conditions to debug and test conflict resolution and data path integrity.

CSRD has also designed diagnostic and maintenance boards for the Cedar system. Diagnostic "hooks" are provided on both the GIBs and GMBs. A number of 8751 attached processors monitor the status of GIBs and GMBs and report any errors, via RS232, to a Sun IV workstation. GN has internal diagnostics that provide a simple go/no-go LED indication on each board.

For maintenance, a global reset board and an environmental monitor system have been designed at CSRD. The global reset board provides a precise reset capability for Cedar. The environmental monitor system, which is composed of four small boards, monitors Cedar environmental aspects such as air temperature, cooling fluid temperature, and power supply load and status.

Performance Monitoring Hardware

Throughout the design of the Cedar-specific boards, performance monitoring has been a critical objective. The level of design complexity required for the basic functionality of the interface, interconnection network, and shared memory prevented the incorporation of integral performance monitoring counters and timers on the boards themselves. Consequently, to supplement the large number of performance signals available on the cluster (Alliant) backplane, a significant number of representative signals were identified on each board as observation points, and these signals were buffered and brought out to the edge of each board. The interface board contains 49 signals (e.g., start and stop triggers, access to the data path, prefetch unit state data, synchronization operation data, processor side and network side interface data), the network switch has 128, and the shared global memory has 55 per board.

To collect and process these data, the industry standard VME bus has been adopted by CSRD for the development of all moderate-to-very-sophisticated performance monitoring hardware tools. Prototype board quantities have been constructed for two fundamental hardware performance tools: a hardware Histogrammer and a hardware Tracer. The Histogrammer is capable of histogramming 64K different events at a 170-nanosecond rate. The Tracer is capable of tracing 20 bits of data with a depth of 1 million samples at a 170-nanosecond rate. Both of these boards support interleaving to increase acquisition speed and depth. This allows the Tracers and Histogrammers to be used on a one-per-CE basis for software event rates, and on a two-per-CE basis for hardware event rates. A Sun processor board resides in the VME chassis with the performance analysis hardware to extract data from these boards and to provide analysis capabilities and an ethernet connection to the other computers at CSRD.

For smaller problems and individual signal measuring, a substantially smaller "black box" card is used, and thus small, specific, limited function cards can be developed quickly and inexpensively. Three such modules have been developed to date: two types of data acquisition boards (with associated data probes) for collecting TTL and ECL signals from Cedar, and a dual-channel 16-bit counter-timer with storage capable of storing 32K counts. An interface between the black box and a Multibus chassis (the principal I/O bus of Cedar) has been built, and the necessary software support for black box control has been written.

Diagnostic Hardware

The ability to detect and identify hardware problems in a working system was established as a very important design aspect in the early phase of Cedar development. The large number of boards in this system, coupled with the problems of online diagnostic and isolation techniques, makes the process of board-swapping for fault isolation wholly unacceptable. We chose the approach of attached diagnostic processors for groups of boards and designed error status and handling circuitry into each board type to report errors (via the diagnostic processors) to a host work station.

We chose to keep the diagnostic processor design as simple as possible to increase reliability and ease the design requirements for both the diagnostic itself as well as the error reporting circuitry on each of the board types. To this end, we chose to use a single chip microcontroller that provides a bit-serial interface to the outside world, and surrounded it with the "glue" logic necessary for interface to each board type. In addition, the ROM-resident code for the diagnostic processors is contained in the microcontroller as well as the static RAM for the storage of error status.

At present, we have all required diagnostic processor boards for both the GIB and GMB boards (the diagnostics processors are embedded in the global network boards). The diagnostic is fully operational in the Cedar 1A system. All the application-specific code has been generated and debugged for the diagnostic processors. The serial interface mentioned above is currently being used to dump diagnostic information to a Sun IV workstation.

We have chosen to use a Sun as the workstation to tie all the diagnostic processors together. Additionally, Alliant-based code has been developed to exercise the Cedar system, and work continues on the development of system code to perform on-line checks and fault isolation.

Physical Equipment

Physical equipment designed at CSRD includes three different extender board types as well as the powering, cooling, and housing of Cedar. Four cabinets have been designed at CSRD. Two of these hold the four Alliant clusters (Cedar 1A), one holds the network, and the fourth, smaller cabinet, contains the memory system. Powering is provided by large switching supplies with a total output current capability in excess of 2000 amps. Cooling is provided for by chilled water heat exchangers, of which there are three in each cabinet. The capacity of each exchanger is 7.5 Kwatts.

Operating System

Xylem (Cedar)

We have taken a very conservative approach to implementing an operating system for Cedar. Instead of starting from scratch, we chose to adopt Unix and to start with Alliant Concentrix (which is based on Berkeley 4.2). Our system is called Xylem[1]. The main areas where Concentrix needed to be modified were to provide: (1) support for the utilization of multiple clusters by a single program, (2) management of the physical main memory hierarchy (global vs. cluster), (3) support for shared virtual memory between

[1] "A complex tissue in the vascular system of higher plants...functioning chiefly in conduction but also in support and storage..."

clusters, and (4) inter-cluster synchronization and communication mechanisms. Xylem has evolved in such a way that each Alliant cluster still appears to execute an independent Concentrix OS while also managing Xylem-specific programs that utilize the features of the Cedar system. This year has seen progress in a number of areas, described in the following paragraphs.

User-level inter-cluster and inter-processor synchronization mechanisms have been further developed. We have run-time library routines for doing mutual exclusion and post-wait event style synchronization. Internal interfaces within these routines have been simplified and overall they have been made more robust, including fixing a bug that could lead to deadlock during interrupt routines.

The saving and restoring of Cedar-specific hardware registers during interrupts and context switching has been fixed. This was done incorrectly in the first implementation and so the use of Cedar synchronization hardware was unreliable. There were small timing windows during which the contents of these registers could be corrupted by the OS, resulting in the execution of operations that were not intended. This problem has been cured, and a number of diagnostic tests are now available to verify correct operation of the hardware while under control of the OS.

The hardware group has corrected the problems we had with the high-resolution (10-microsecond) clocks on each cluster. Previously these would appear to drift, as if running at slightly different rates. As a result, it was difficult to tell the same time in tasks on different clusters. Elaborate software mechanisms were in the kernel to detect this drift and then resynchronize the software. This resynchronization process would necessarily tie up clusters for long periods of time, and it happened often enough during normal scheduling activities to impact user program performance. A new algorithm has been designed to sync the clocks only once during system initialization. At the same time, rather than computing pairwise time deltas between clusters, we will use the notion of synchronizing against a global time base. As a result, it will be much easier for programs to measure time and coordinate events across clusters, with no system interference. For example, a periodic system utility can run once a day to verify that the clocks have indeed maintained synchronization.

Another problem that required a considerable amount of attention was the diagnosis of bugs in the physical memory allocators and process swapping algorithms in the kernel. Some of the indicators of these bugs are that processes might get swapped out and then never get swapped back in, or processes might get stuck in memory (i.e., not swapped out) such that other processes could not be swapped even though they we ready and had sufficient priority. These problems were caused by a combination of incorrectly accounting for how physical memory was being used, and incorrect computation of working set sizes by the system swapper. We are confident of our diagnosis, and most of these problems have been corrected, but there are still accounting errors to be isolated. Algorithms in the swapper have been made more fault tolerant of such inconsistencies.

Instrumentation of Xylem

The current implementation of Xylem is partially instrumented. For example, facilities exist to account for every page fault in terms of what kind of memory was faulted (e.g., cluster vs. global, shared vs. private), and how it was resolved (image from memory or disk,) [CSRD857** & CSRD858*]. Accounting for system-wide paging activities has recently been installed, and we are looking forward to observing the paging behavior and tuning this aspect of system performance.

We have designed a comprehensive instrumentation approach using the hardware facilities available on the Cedar GIB board. This hardware will let us collect trace events and send them to a dedicated processor. Without adding much overhead in the kernel, we intend to collect timestamps on all kinds of system wide activities, such as interrupts, context switches, and page faults. Some low level subroutines are already in place to make trace events and write them to the hardware buffer. With these capabilities fully functional, we will be able to determine what is really happening inside the OS.

Single User Batch

Recently, a single user batch facility has been implemented and installed on Cedar to make it possible to submit a job to be run in single user mode automatically. At predetermined or regularly scheduled times, the normal time-sharing system will shut itself down and initiate single user batch processing. If there are any jobs in the single user batch queue, they will be run one at a time, and when finished, normal time-sharing will resume. This will remove a large burden on our user community. Many of these people need the machine in a single user mode, but still operate under Xylem, primarily to do benchmarking runs. Previously, they had to sign up in advance for time slots in the evening or early morning, and then come to the machine room and work at the console during their assigned time.

Debugging Support in Parallel Programming Environments

This study involves some of the problems of debugging parallel programs, particularly those at a low level (i.e., machine code level). Problems unique to parallel programming environments include identifying and probing execution threads and synchronization of the debugger within itself and with the user's program. This work also intends to identify a small set of specific facilities that the operating system kernel can provide to support parallel debugging tools. An initial implementation of interactive parallel debugging facilities has been done in the Xylem environment. Different approaches are being tried, and deficiencies in kernel support have been identified. Ideally, this work will build a base upon which high-level debuggers can be built.

Research

Recently, we have been discussing new ideas for the design of a multi-processor operating system. We observe that Unix-based operating systems have gotten larger and more complex over the past decade, as much as one to two orders of magnitude compared to the early Bell Labs versions. Yet with this tremendous increase in size and complexity, there is very little improvement in functionality or performance. As a result, our thinking leans toward a very simple design, which is portable to a number of shared and hierarchical memory machines.

For example, unlike most research efforts, we are considering a kernel that does not require or utilize virtual memory management hardware. This is prompted by two considerations: (1) users continually request that we eliminate virtual memory because it always interferes with performance, and (2) we are targeting Cray machines, which do not have virtual memory hardware.

Currently, we envision the multiprocessor kernel as only providing processor and physical memory management, scheduling, swapping, and trap handling. I/O and network activity may be passed off (via messages) to attached host processors.

Languages and Compilers

Our work on languages and compilers has been guided by three main objectives: (1) to provide the programmers of the Cedar System with high quality compilers and restructurers; (2) to conduct research on compilers and restructurers leading to the development of algorithms that are useful for our work on Cedar, but will also be applicable on a wide spectrum of machines; and (3) to study compilation and restructuring techniques for languages other than Fortran.

Fortran Translation

Fortran, the most frequently used language in supercomputing today, is the most natural language on which to base a major portion of our effort mainly because most supercomputer applications are dominated by numerical computations, including most of the algorithm research work at CSRD. In addition, Fortran will most likely be used on supercomputers for a long time to come.

Cedar Fortran

Our work on Fortran began with the design of extensions to Alliant Fortran for parallel processing on Cedar. Alliant Fortran includes extensions for vector operations. These are essentially those in the current Fortran 8X proposal. We decided to adopt the extensions and concentrate our efforts on extensions for concurrency. To this end, we developed constructs for task parallelism and loop parallelism. For task parallelism we designed constructs similar to those of Cray Fortran. These constructs are adequate, and the applications group at CSRD felt comfortable with them. One major difference between Cray Fortran and our (**ctskstart**) task spawning routine is that the latter allocates not one processor, but several to the newly created task, in order to allow the cluster tasks to use the Alliant hardware in the execution of concurrent loops. We have also developed a micro-task spawning mechanism centered around the **mtskstart** routine. This routine creates a microtask and queues it according to a priority specified in one of its parameters. Helping (or implicit) tasks remove microtasks from the queue and execute them to completion before going back to the queue for more work. Microtasking has a lower spawning overhead than regular tasking. However, microtasking is in some cases harder to use since it does not allow synchronization operations other than those used to wait for the completion of a microtask (**mtskwait** and **mtskwaitall** in Cedar Fortran).

Extensions for loop parallelism included **doall** and **doacross** loops. These resemble the regular Fortran **do** loop, but their iterations may proceed in parallel. Concurrent loops may be confined inside a cluster or execute across clusters. In the first case they make use of the Alliant hardware for the scheduling of iterations on processors. When concurrent loops are executed across clusters, there are two alternatives. In the first, whole (virtual) clusters are used to execute the loop in a self-scheduled manner. In the second alternative, several individual processors from two or more clusters are used to execute the loop, also in a self-scheduled manner. The scheduling in these two last cases is handled by software under the microtasking scheme. Concurrent loops executed by whole clusters are supposed to enclose concurrent loops of the cluster-confined type to allow the use of all the processors in a cluster.

Cedar Fortran includes extensions to specify at which memory hierarchy level individual variables and common blocks are to reside, and includes extensions to handle the Cedar memory synchronization hardware primitives. A detailed description of Cedar Fortran can be found in [PaLa86], [Guzz87], and [GPLH90].

From the point of view of implementation, the Cedar Fortran compiler comprises a source-to-source preprocessor, a slightly modified version of the Alliant Fortran compiler, and an object-to-object postprocessor. The preprocessor executes before the Alliant Fortran Compiler and translates Cedar Fortran constructs to Alliant Fortran constructs or to subroutine calls. The postprocessor executes after the Alliant Fortran compiler and translates Unix object files into Xylem object files. The only modifications to the Alliant Fortran compiler were those needed to generate the machine code for vector prefetching from global memory via the Global Interface Board. A major portion of the compiler work was the development of new run-time library routines needed for the Cedar Fortran constructs, and the modification of the Alliant run-time routines to allow them to run under Xylem. The Cedar Fortran compiler with its run-time library is now operational and has been extensively tested. Much effort has been invested lately in decreasing the overhead associated with the run-time library and providing efficient mechanisms for data communication and synchronization. To this end, we are experimenting with strategies for dispatching and scheduling DOALL loops across the four Cedar clusters. For example, we have found that static dispatching brings the overhead down to 40 uses. Self-scheduling using the Cedar synchronization hardware is also being tested.

Fortran Restructurer

A second component of the Fortran translation system is a restructurer that translates from Fortran 77 to Cedar Fortran. The present version of the restructurer is KAP, a commercial product developed by KAI. We have modified this source-to-source restructurer to generate Cedar Fortran as its output. In its current state, the implementation automatically identifies concurrent loops and vector operations. The system performs strip-mining as necessary to permit the exploitation of the three levels of parallelism available in Cedar. It also allocates variables in one of the memory hierarchy levels either automatically or by using user-supplied statements.

So far we have demonstrated good performance on Cedar for program kernels and some algorithms. Additional work is needed to exploit efficiently the architectural features of Cedar. To make the restructurer a powerful tool for the Cedar system we need to accomplish the following:

1. Make it possible for the restructurer to accept Cedar Fortran as input. There are several reasons why this is desirable. One is the obvious appeal of not having to think in terms of two dialects of Fortran. The second, and probably the most important, is that programmers may want to indicate parallelism explicitly before restructuring takes place, and Cedar Fortran is the most natural way to do this. Accepting Cedar Fortran implies that the input to the parallelizer might be a parallel program. The transformation of parallel programs presents some difficulties that are under study [MiPa90].

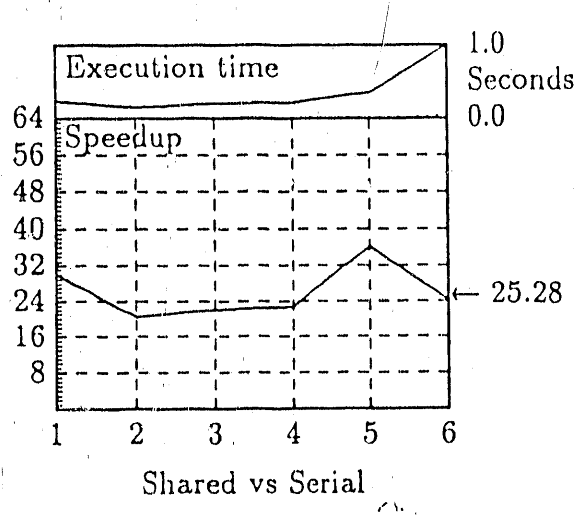
2. Spreading (the transformation that generates task parallelism) has not yet been implemented. An effective implementation of spreading requires a good interprocedural analysis subsystem, which has not yet been implemented either. Interprocedural analysis has been discussed in [LiYe88a,b&c]. Spreading was studied in detail in [Veid85].

3. Finally, we have to apply the restructurer to a broad range of algorithms and application programs. This enables us to study and refine its operation in practice. In a series of experiments we have identified a number of improvements to compilation techniques and strategies that cause significant performance gains in large codes. The following section illustrates some experiments. Incorporating these experimental results into the restructurer's capabilities is an ongoing effort.

Experiments with the Restructurer

We have timed various applications from the Perfect Benchmarks ™ on Cedar and measured the performance of individual loop nests (by summing the wall-clock time of all the activations of each lexically outermost loop). The measurements of OCEAN and FLO52 are summarized by the graphs in Figure 2, where each tic on the x-axis represents a different loop nest within the program. Of course, each loop nest has a different impact on the overall performance of a program, so we present them sorted in increasing order by their serial execution time. Loop nests plotted at the far right cover more of the application's execution time than those to the left.

The y-axis contains two regions: at the bottom we report a speedup ratio for each nest, and on top we accumulate the execution time of all nests from left to right. On the top region, we represent the maximum time accumulated by any of the nests in any version of the program. The solid curve in this region represents accumulated serial execution time. The dotted curve in this region is the execution time of the parallelized version. To the right of each speedup plot an arrow reports the overall speed-up of the whole program.



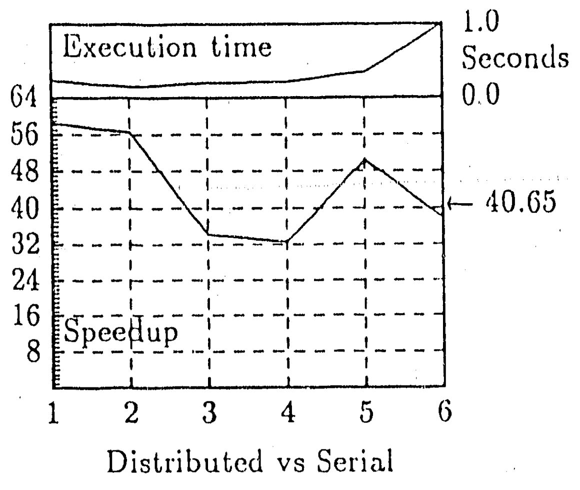


Figure 2: Speedup of Perfect Benchmarks ™ OCEAN and FLO52

The serial version in Figure 2 uses the traditional serial optimizations (like common subexpression elimination) applied by the Alliant code generator. The (SDOALL) version of the program is the result of applying the Cedar Restructurer to the serial version, commanding it to place all interface variables in GLOBAL memory, thus enabling a maximum number of SDOALLs.

The obvious conclusion to be reached from the graphs of Figure 2 is that the overall performance of an application nearly equals the performance of its most important loop(s). We will refer to the loops which accumulated the top 90% of execution time as the *significant* loops of the program. In OCEAN, the Cedar Restructurer's SDOALL version did well on many of the low-run-time loops, but not as well with the significant ones. Consequently, overall speedup was poor (0.74). In FLO 52, the SDOALL version performed well on the most significant fifth, and the mixed performance elsewhere did not detract from a respectable speedup of 6.7 for the whole program.

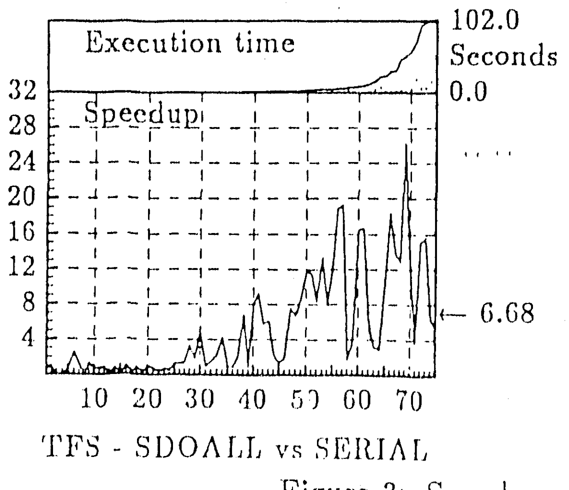
The OCEAN program makes complex use of induction variables in many of its significant loops, the Cedar Restructurer had trouble following some of this. It also calls a handful of low-level subroutines in many loops, forcing the loops serial. Occasionally, inner loops of a loop nest contained calculations with no subroutine calls and these were successfully stripmined and run as SDOALLSs, while the outer loop remained serial because of subroutine. Interprocedural analysis was not applied in this experiment because of the problems with our version of KAP in its in-line expansion pass. We have recently fixed some of these problems, and are planning to use in-line expansion in the next experiment. We also plan to study the development of a more sophisticated induction variable analysis.

The significant loops of the FLO52 program contain many easily vectorizable calculations. All of the significant loop nests had at least an inner loop which was parallelizable, and some nests could be totally parallelized. In the FLO52 program, three significant loop nests had an outer loop that was forced to remain serial because there were too many inner loops. Increasing the data structure sizes allowed two of these to be parallelized. Presumably the other could be parallelized with even larger data structures. The recurrence solvers that we use have not been optimized for Cedar, which was the source of poor performance on two significant loops.

In the FLO52 code, the Cedar Restructurer produced parallel versions of many of the significant loops, but their speedup is less than optimal. Why is that? The most important reason may be that the average running time for most of the loops, even in the serial version, is very short and barely overcomes the SDOALL loop overhead costs. Multi-version loops might cut our losses here.

In one benchmark exercise we solved a banded linear system of 255 equations using the Conjugate Gradient algorithm. Our test data required approximately 65,000 iterations in the key loops, enough to exploit all the parallelism in Cedar. The kernels were transformed into SDOALL/CDOALL/vector loop nests. Figure 3 shows the speedups for the main kernels of this code. The basic statistic is the *minimum* execution time over several hundred repetitions. This statistic eliminates a few executions inadvertently delayed by other system software.

These runs were made on a half-sized Cedar with 16 CEs, each having 4 stages in its vector pipeline, so 64 calculations can occur in parallel. But the effective parallelism in the shared memory network never exactly matches the processor parallelism. Speedup ratios vary from kernel to kernel because each has a different ratio between the



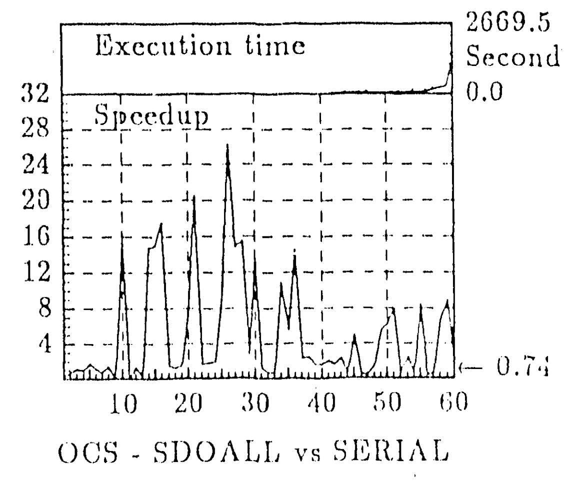


Figure 3: Speedup of Conjugate Gradient Algorithm

number of memory references and the number of arithmetic operations. The beneficial effects of cache and prefetching, and the detrimental effects of network contention affect each loop differently.

The second diagram in Figure 3 reports the speedup (parallel times vector) due to manual application of the data distribution techniques. The bandwidth of the global network and memory, which limits the shared methods, does not restrict this scheme. The overall speedup is now 40, and individual kernels are becoming processor-bound.

Other Translators

Two other projects have been undertaken in the compiler area. We plan to continue both of them. However, in this section we only describe them briefly. Later in the proposal, some specific parts of these projects will be discussed in detail.

The first project is Parafrase 2. The main objective of this project is to develop a successor to the Parafrase Fortran restructurer. The Parafrase-2 project involves the construction of a multilingual parallelizing compiler for C and Fortran, which will incorporate the latest technology on parallelization, as well as recent research results of our group. Parafrase-2 is being developed not only as a source-to-source parallelizer but also as a fully develo[illegible] a compiler which generates code for an abstract parallel architecture.

Although this is an ongoing project involving a faculty member and four graduate students, the compiler has passed its first phase of development and is currently in use at CSRD as well as several other universities and research centers in the U.S. At present, Parafrase-2 is capable of performing loop parallelization inter- and intra-procedurally. Powerful data dependence analysis with new techniques for symbolic dependence analysis have been implemented along with inter-procedural analysis. Several transformations and optimizations have also been completed, including loop parallelization, scalar expansion, loop blocking, induction variable recognition, loop interchange, loop distribution and vectorization.

In addition, an interactive graphics interface has been added to Parafrase-2, which provides a convenient means for visualizing and manipulating internal program representations such as the control flow graph, the procedure call graph, and the data dependence graph. Parafrase-2's graphics interface, which runs under X windows, has provided a powerful tool in interacting with the compiler and guiding parallelization wherever needed. Parafrase-2, which has been developed in C and runs under Unix, comprises at present 80K lines of C code. The use of data abstraction in the development of the system has proven instrumental in its use as the basis for building simulators or other compilers on top of it, at several centers where it is being used.

The second project, MIPRAC, is an attempt to fuse the techniques of automatic parallelization of numerical codes (as employed in Parafrase and its descendants) with those of symbolic codes (as employed in PARCEL, a LISP parallelizer [HaPa88]). The goal is to use intersections and unions of these techniques to arrive at a compiler that can parallelize, for example, a C program that manipulates pointer variables, dynamically allocated objects, as well as arrays of numerical data. The compiler itself will operate upon an intermediate form into which programs in a variety of languages may be rewritten. We have implemented translators from Scheme and Fortran 77 into MIPRAC's intermediate form and are implementing one for Pascal. We intend that programs that are written in several source languages may be sensibly analyzed, compiled, and linked using this multilingual translator.

To ensure that MIPRAC's intermediate form is as compact as possible, several phases of syntactic and control-flow normalization are used to coerce input programs into a highly normalized form. Following this normalization, an interprocedural analysis is performed which extends the side effect and lifetime analysis used in PARCEL to vector and array data. MIPRAC's parallelizing transformations will include array- and DO-loop-oriented transformations (loop distribution, loop interchange, scalar expansion) as well as recursion- and WHILE-loop-oriented transformations (recursion splitting, exit-loop parallelization).

To date, the phases of syntactic and control-flow normalization and the first phase of interprocedural analysis have been implemented.

MIPRAC will produce a machine-independent, parallel intermediate code that can be executed using PARCEL's run-time system, which is being extended to accommodate the requirements of MIPRAC's intermediate form.

The Ansi C, Scheme, and Common LISP front ends for MIPRAC are implemented, and we have begun debugging them. The front ends rewrite C and Common LISP programs into the intermediate language used by MIPRAC (called MIL). The control structures in these languages (tail recursion in Scheme, `foreach` and `mapcar` in Common LISP, `for`, `do / while`, `break` in C) are too many and too diverse to be represented directly in the intermediate form. Our solution is to give all of these structures a simple translation in terms of `gotos` and labels, in MIL. Afterward, a phase of control flow normalization eliminates all `gotos` and labels, leaving only properly nested `begins`, `ifs`, `whiles`, and procedure calls. A final phase of normalization rewrites the `while` loops as tail-recursive procedures, so that in the end we have only `begin`, `if` and procedure call as control structures in a MIL program. The normalization phases are complete and have been debugged.

The interprocedural analysis of MIPRAC has been implemented and is being tested and debugged. The analysis is in C; its interface to MIPRAC is a simple, ASCII one: an encoding of the program arrives on standard input as an ascii text, and the output is returned in ascii format. The analysis is therefore available to others, as a module separable from MIPRAC. Three kinds of information are gathered by the analysis: the interprocedurally visible side effects of procedures; the lifetimes of dynamically allocated objects; and structure sharing within the data structures created by the program. The analysis uses a new algorithm for finding the fixpoints of functionals efficiently. This algorithm employs techniques from data flow analysis to direct the solution of fixpoint equations. We intend to parallelize the analysis and to run it remotely on the Alliants or Cedar, to speed the analysis of large programs.

We are also developing a new theory of automatic parallelization based entirely upon recurrence relations. The idea is that in every iterative or recursive structure there are simple recurrence relations (e.g., induction variables) from which more complex recurrence relations are computed, and finally from which the sequence of memory addresses that are accessed are computed. Automatic parallelization requires two things: a view of the relationship between objects in memory, and a view of the recurrence relations computed by iterative and recursive structures. These two components allow us to reason about whether the addresses issued in one iteration overlap with those issued in another iteration. In general, we find that if the recurrences that carry us from locations to new locations are "linear" in that they do not return to previously accessed locations, then the iterations of a recursive or iterative structure are independent. These observations may be made the basis of an algorithm for automatic parallelization that treats both subscripted and pointer accesses in a single framework. We are trying to do exactly this in MIPRAC. The two components of the framework are the above-mentioned interprocedural analysis (which gives us a view of the objects in memory and their interrelationship) and an interprocedural recurrence recognizer, which gives us a view of the recurrences described by iterative and recursive structures. (In fact, by the time these analyses occur in MIPRAC, all iterative structures from Fortran, C, and Common LISP are rewritten as recursive or tail-recursive procedures in the intermediate language of MIPRAC.)

Finally, we are designing a code generator and run-time support for MIL (the intermediate language of MIPRAC) for the Cedar machine. The intermediate language has been designed to make this particularly simple: there are only a handful of control forms, and low-level operations that correspond mostly to sequences of a few instructions. Input/output operations are equally simple: block reads and writes, creation and deletion of files, and positioning of the cursor (point of reading/writing) within a file. The run-time support for a source language like Common LISP, which has many built-in procedures, is written entirely in MIL itself, and compiled using MIPRAC, so that this portion of the run-time system will not require porting. We are concentrating upon the efficient translation of the intermediate form into the Alliant/Cedar instruction set. One especially interesting aspect of the run-time system and code generator is its automatic storage reclamation. The run-time system will not use a conventional garbage collector, but rather will automatically deallocate objects according to their lifetimes, as determined by the interprocedural analysis. This means, for one thing, that we will support automatic storage reclamation for C programs. For another, it means that the system will have the desirable aspects of concurrent garbage collection (no lengthy pauses in processing to reclaim storage). The reclamation algorithm is particularly simple to implement, but its success will ultimately depend entirely upon the accuracy of the interprocedural analysis.

Multiprocessor Performance on Algorithms and Applications

In the past year, the Applications Group in the Center has been involved in two main activities:

1. The design and implementation of various hierarchical numerical algorithms that take advantage of the three levels of parallelism of the four-cluster Cedar in which each cluster contains four vector processors (e.g., the 4x4 Cedar).
2. The parallelization of several large-scale application codes in ocean modeling, crash worthiness, and thermal hydraulics, and their implementation on Cedar.

Algorithm development on Cedar

Several parallel algorithms have been developed for implementation on the 4x4 Cedar in which each cluster memory consists of 2MW and a global memory of 2MW with a modified intermediate global network. These algorithms deal with crucial parts in important applications:

SPIKE: A Hierarchical Parallel Block-Tridiagonal System Solver

The solution of large banded diagonally dominant or symmetric positive definite linear systems constitutes one of the most common computational tasks associated with implementations of the finite element method in applications such as fluid dynamics and structural analysis. The design of multiprocessor algorithms for solving large block-tridiagonal systems becomes of paramount importance for efficient implementation of these applications on vector and parallel machines. In [Berr90], we discuss the design and implementation of a hierarchical-based method, SPIKE, for solving these systems on the 4-cluster (16-processor) Cedar machine. We compare the performance of SPIKE with that of an efficient block Gaussian elimination scheme, BGE, on an 8-processor Alliant FX/80. Results for the SPIKE algorithm on CEDAR block-tridiagonal system of order 16384 with block size 16 indicate speedups of greater than 6.5 vs. the best segmental scheme and a speed improvement of 11.5 vs. SPIKE on one processor.

A Symmetric and Antisymmetric Domain Decomposition in Structural Mechanics

Domain decomposition has recently become a topic under intensive research (see [GGMP88] and [CGPW89] for references). The symmetric and antisymmetric domain decomposition introduced in [Chen88] and [ChSa89a&b] is ideally suited for problems in structural mechanics. It is ideally suited for multiprocessors such as Cray Y-MP, Cray-2, Alliant FX/80, and Cedar. Unlike other methods that decompose the symmetric and antisymmetric response of a structure (e.g., see [NoPe87a&b], [BrDM88], [DoSm88]), our scheme exploits special properties of reflexive matrices to decompose stiffness matrices into several independent subproblems. Speedups realized for 2-D and 3-D elasticity problems on an Alliant FX/80 vs. the best sequential solvers range from 5.5 to 7.0 (out of a maximum of 8) depending on the problem. For the three levels of parallelism of Cedar, speedup of 9 (out of a maximum of 16) is realizable (see [Chen90]), even on unsymmetric structures that can be preconditioned using the symmetric and antisymmetric domain decomposition on the symmetric structure which is considered as a small perturbation of the original problem.

The Block Cimmino Method

The block Cimmino algorithm is a row projection scheme for nonsymmetric indefinite linear systems with arbitrary spectral distributions [Bram89], [BrSa90a]. The method can be more robust than other commonly used iterative solvers such as conjugate gradients (CG) on the normal equations or Krylov subspace based methods such as GMRES*(k)*. The most time-consuming part of the algorithm involves forming matrix-vector products, with a matrix that is the sum of orthogonal projectors. For structured problems such as those arising from discretizations of partial differential equations, each projection can be computed as a set of parallel tasks, each such task having some vectorization possible. Block Cimmino is thus especially suitable for the Cedar architecture, because the algorithm allows the necessary three levels of parallelism. A particular implementation is described and tested on Cedar [Bram90], showing speedups of 11 (out of a maximum of 16) that are realizable even on modest size problems.

Parallel Elliptic and Parabolic Problem Solvers

A rapid elliptic solver based on a parallel block cyclic reduction scheme [GaSa89] has been implemented on 2x4 Cedar with favorable results [Fran90]. Further, algorithms based on the conjugate gradient schemes, with and without preconditioning, have been implemented on Cedar [MeEi90] for self-adjoint elliptic problems on regular domains, and on T-shaped domains with the Schwarz alternating procedure [GaFM90].

Parallel algorithms for solving linear parabolic problems via implicit methods, based on Pade and Chebyshev rational approximation to matrix exponential, are implemented on the 4x4 Cedar. The algorithms are ideally suited for the hierarchical memory organization of Cedar and realize high speedup vs. the classical sequential Crank-Nicolson scheme. Moreover, these algorithms are those of choice if the goal is to compute the solution at a given time point as quickly as possible.

Direct Sparse System Solvers

In this activity we have developed two solvers:

1. McSparse [GaMW90] - This algorithm is ideally suited for large grain parallelism that takes advantage of the multicluster organization of Cedar. This algorithm consists of two stages. The first uses a hybrid ordering [Wijs89b] which produces (on most general sparse matrices) a block sparse upper triangular matrix. A sparse block Gaussian elimination procedure (with pivoting) produces an LU-factorization using the four clusters of Cedar with reasonable load balancing.
2. DSPACK [Yang90] - This algorithm was designed to test Cedar as a "flat" multiprocessor which utilizes mainly the global memory and capitalizes on the fine-grain parallelism of the 16-CE Cedar. In spite of lack of explicit utilization of the individual cluster memories, speedups of around 6 (out of a maximum of 16) vs. the best sequential scheme are realizable.

Multiprocessing of Large Scale Codes

The purpose of this project is to apply techniques of paral'el computing to mature computational fluid dynamics (CFD) and structural dynamics (SD) computer codes. Specifically, although the power and utility of parallel computers and the corresponding software development techniques have been demonstrated on small scale prototype problems, full-scale production use has lagged behind because parallelizing large engineering codes requires coordination and interaction among engineers (who understand the physical problem being modeled), numerical algorithm designers (who specialize in designing efficient algorithms to solve basic computational problems), and performance evaluation experts (who build software tools and machine models that enable the design of efficient programs for a specific computer). For this effort, we are collaborating with another DOE site, Argonne National Laboratory, combining our expertise to bear on industrially important problems. We are also continuing our work on a problem of concern to environmental activities, namely the multiprocessing of a state-of-the-art ocean circulation model (OCM).

Upon completion, we expect two major results. First, the very process described above will expose the strengths and weaknesses of existing techniques for parallelizing programs and will identify those problems that need to be solved in order to enable widespread production use of parallel computers. Second, the increased efficiency of the CFD and SD codes will allow the simulation of larger, more accurate engineering models. In particular, we hope that the work could serve as an exemplary model for similar future activities.

Our main parallel machines are the 4 CPU Cray X-MP and Y-MP, Alliant FX/80 and FX/2800 and Cedar.

Parallel Ocean Circulation

Global ocean circulation modeling is an important component of climate prediction studies. In an effort to conduct faithful simulations, one needs considerable computational power and storage capabilities. Interest in such efforts is evidenced in DOE's recent CHAMMP initiative. Much pioneering work in the parallelization of ocean global circulation modeling has been conducted by Semtner and Chervin for the Cray. Our interest is to build models that profit from computational and memory hierarchies for good performance. We hope that such models will maintain good performance as machine and problem size scale.

In cooperation with A. Navarra from IMGA-CNR, Modena, Italy we are developing a parallel version of a state-of-the-art model of circulation in the Mediterranean basin. The model is based on the ocean circulation codes from GFDL, Princeton University, and simulates the basic aspects of large-scale baroclinic ocean circulation. A preliminary multicluster Cedar implementation of the code has been completed. In this phase of the work we have concentrated on the aspects of the code not dealing with the two-dimensional relaxation procedure, used to compute transport streamfunction The two models of interest consist of 167 x 57 x 8 and 334 x 118 x 16 gridpoints respectively. The smaller model is used primarily to allow some performance evaluation of a single cluster code, since the large model is slowed down considerably on one cluster due to paging. For example, Figure 4 shows the effect of using four clusters of Cedar relative to the performance of the eight advanced computational elements of the Alliant FX/80. The times for one and two Cedar clusters are very large due to the small memory available compared to that of the FX/80 and the paging behavior. Figure 5 corresponds to the smaller model. It shows that the two clusters of Cedar (8 CEs) perform almost as well as the eight faster Advanced CEs of the FX/80.

Tables 1 and 2 show the times, speedups, and efficiencies for each of the phases of the calculation. The three-dimensional computations have been distributed across the clusters. The relaxation was left on one cluster as we are currently implementing a multicluster version.

All the multicluster Cedar speedups we have shown are constrained by our not parallelizing the two-dimensional relaxation. This, together with the evaluation of different data-partitioning techniques constitutes our next task in the project.

COMMIX and WHAMS

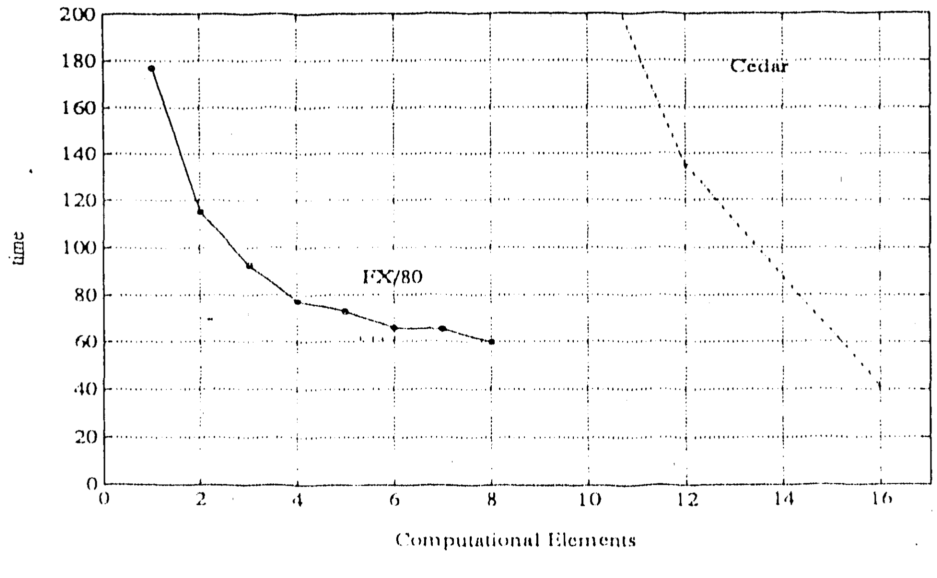


Figure 4. Preliminary performance of OCM on Alliant FX/80 and Cedar for 3 time steps of 334 x 118 x 16 grid.

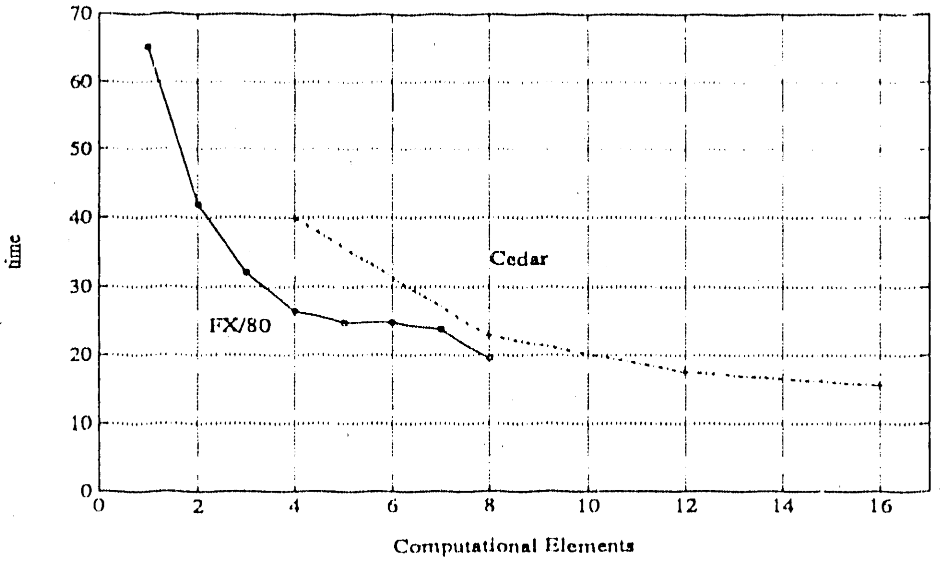


Figure 5. Preliminary performance of OCM on Alliant FX/80 and Cedar for 10 time steps of 167 x 57 x 8 grid.

proc.	number of clusters			
	1	2	3	4
startup	0.03	0.03	0.03	0.03
3D phase	36.62	18.88	13.26	10.98
relaxation	3.55	3.79	3.58	3.62
total	40.20	22.70	16.87	15.03

Table 1. Small ocean model running 10 time steps (preliminary).

Clusters (CE)	Speedup	Efficiency	Speedup for 3D ph.	Eff. 3D ph.
1 (4)	1.00	1.00	1.00	1.00
2 (8)	1.77	0.89	1.94	0.97
3 (12)	2.38	0.79	2.76	0.92
4 (16)	2.67	0.67	3.33	0.83

Table 2. Small ocean model running 10 time steps (preliminary).

COMMIX and WHAMS-3D are two production codes that have been developed at ANL and are widely used by industry and government laboratories. The first is a computational fluid dynamics code that is used for both nuclear reactor design and safety and as a design tool for the casting industry. COMMIX 1-ARP consists of approximately 30,000 lines of code across 150 subroutines [BIGG89]. WHAMS-3D is a three-dimensional structural dynamics code used in nuclear reactor safety as well as crashworthiness studies [BeTs82]. Both codes are available for both sequential and vector computers only. Our main goal is to optimize these two codes on shared memory multiprocessors. The first phase of this project has been completed and fully documented in [ABCG90]. It consisted of an intense profile study of both codes using data sets typically used by ANL for testing the validity of the codes.

Argonne's development and refinement of COMMIX, which has continued for more than ten years, was originally supported by the U.S. Nuclear Regulatory Commission for application to a wide variety of reactor safety problems. COMMIX has been developed using a unique porous media approach to the solution of the Navier-Stokes equations in an arbitrary three-dimensional region. In its various versions, COMMIX can model separate single-phase fluids, multiphase flows, and free surface flows. The code uses differenced momentum/mass conservation equations which are combined to form a pressure equation. Once the pressures are known, the fluid velocities are updated to provide input to the energy simulation and the next iteration or time step. The hydraulic driving force may be flow or pressure boundary conditions at inlets and outlets, one of several pump models, or a fluid temperature/density distribution. The energy equations are differenced using the updated velocities, and the source terms are accumulated from the treatments of convection boundaries, conduction boundaries, thermal structures, or heat generation in the fluid itself. One-dimensional shell structures superimposed on the fluid geometry model various fluid system thermal components such as vessels, pipes, baffles, tube-shell heat exchangers, and reactor fuel. Once the fluid temperature distribution is updated, the submerged thermal structures' internal temperature distributions are recomputed, assuming one-dimensional conduction through each thermal structure segment.

The momentum and fluid energy equation time differencing is implicit, which requires that the difference equation coefficients be constructed from end-of-timestep temperatures and velocities. The pressure equations of the mass-momentum loop are solved using conjugate gradient (CG) with incomplete Cholesky factorization preconditioning.

A large amount of the computing time is spent constructing matrix equations, a process that would require massive recoding to vectorize because of its large, logic-loaded loops. Such loops, however, are expected to lend themselves very well to the parallelization efforts which we plan to pursue in the context of this work.

We next list the function of the most important routines in the set:

energi	Construct coefficients in the energy equation.
lowfcv	Solve the upper and lower triangular system as part of the conjugate gradient solution of the pressure equation.
peqn	Construct the coefficients in the pressure equation.
qstrds	Calculates finite differences of solid/fluid heat transfer rate over the thermal structures.
qstruc	Set solid-to-fluid source term for the fluid energy equation.
solvev	Solver of linear system for the energy equation using Gauss-Seidel relaxation on red-black ordering.
xmomi, ymomi, zmomi	Sweep over all fluid cells to set-up the x, y, and z direction momentum equations.

Table 3 summarizes the runtimes for each of the machines and compilation options for two typical data sets, P1r0 (steady-state calculation) and P1r2 (transient calculation). The compilation options S,V,C stand for scalar, vector, and concurrent optimization. The notation SV(Zv) stands for enhanced vectorization.

Routine	P1r0	P1r2
Cray SV	93.96	920.94
Cray SV(Zv)	81.48	872.90
Cray SVC	225.25	2025.48
1 CPU	22.35	297.28
2 CPU	31.76	405.18
3 CPU	47.19	461.88
4 CPU	123.95	861.20
Alliant FX/80		
SV (1 CE)	1,322.4	12,331.2
SC (8 CE)	1,128.7	10,015.0
SVC (8 CE)	1,136.7	10,048.6

Table 3. Execution times for SV, SV(Zv) and SVC COMMIX-1AR/P on the Cray X-MP/48 and Alliant FX/80.

Data set	P1r0		P1r2	
Routine	SV	SV(Zv)	SV	SV(Zv)
QSTRDS	6.20	6.68	6.19	6.69
ZMOMI	6.42	7.19	6.42	7.15
ENERGI	6.06	6.61	6.06	6.64
SOLVEV	10.29	36.28	10.37	33.51
YMOMI	6.97	7.78	6.97	7.81
QSTRUC	6.67	7.28	6.66	7.29
PEQN	3.05	3.49	3.06	3.50
XMOMI	6.93	7.82	6.93	7.84
LOWFCV	30.98	34.12	30.99	34.57
Overall	9.12	11.76	7.68	8.76

Table 4. MFLOPS profile of the most time consuming subroutines in baseline scalar-vector, and enhanced vectorization -Zv compilation modes for the COMMIX-1AR/P code running on Cray X-MP using data decks P1r0 and P1r2 (from PERFTRACE).

Table 4 shows that the best MFLOPS rate is achieved for the **solvev** and **lowfcv** subroutines, for a performance far superior to all other listed subroutines. These have been written to take advantage of vectorization and thus perform at more than 30 MFLOPS on one CPU of the Cray X-MP. This is in contrast to a meager average of 6.5 MFLOPS for those sections of the code working on the task of matrix construction. This of course causes the overall performance to drop to almost 11 MFLOPS, even for the full vector optimization. It is thus clear that a first step in improving the performance of the code is going to be the restructuring of the matrix construction phase.

Table 4 indicates a great deal about the nature of the work to be done during the next phase of this project:

1. The difference in performance for **solvev** under the different compilation options tells one about the ability of restructuring compilers to take advantage of the machine capabilities, if the code is written properly. It also shows that the use of more sophisticated restructurers can be very beneficial. We should keep in mind however that **solvev** was coded for vector processing.
2. The high performance of **lowfcv** for both SV and SV(Zv) shows that even a less sophisticated restructurer can do well if it has some help from the user (in the form of inline directives).
3. The low performance for both SV and SV(Zv) options for all other routines shows that there is work to be done until these commercial compilers can handle satisfactorily dusty-decks. ("Dusty-decks" as the matrix assembly routines were not coded to take advantage of the architecture.) We will thus investigate the use of novel restructurers as part of Phase 2 of our work.
4. The overall low performance for both compilation options, and the small difference among the two, shows that it is dangerous to concentrate only on those stages of the code that are related to well-defined algebraic computations in need of new algorithms (e.g., linear system solvers), at least until the automatic transformation tools become more powerful.

An important goal of this work is to demonstrate the effectiveness of multiprocessing for these codes applied to industrial-strength data sets. We have secured such sets from the Thermal Hydraulics Section of Commonwealth Edison, including a 12-hour simulation when performed on one vector processor of an IBM 3090.

The WHAMS-3D computer program employs explicit time integration to do nonlinear, transient analysis of frames, shells, plates, and continua in three dimensions. Both material nonlinearities due to elasto-plastic behavior and geometric nonlinearities due to large displacements can be treated. This program has been developed jointly at Northwestern University and Argonne National Laboratory and is internationally recognized as a state-of-the-art program for performing nonlinear transient analysis. The program employs a finite element format, so that it possesses considerable versatility in modeling complex shapes and boundary conditions. The element library consists of the following: quadrilateral and triangular plate-shell elements, a beam element, a spring element, and a hexahedral continuum element. In addition, a rigid linkage is included which permits the efficient modeling of very stiff portions of a structure, such as the bottom ring of a core barrel. In a rigid linkage, the motion of a master node defines the motion of all slave nodes linked to the master node. This option is also useful for eccentrically connected elements where the midlines of the connected elements do not coincide, as, for example, in stiffeners. All of the elements in the program are three dimensional. Table time steps can be automatically computed or input by the user or a driving program. Mixed time integration, a procedure that allows for different time steps in different parts of the mesh, may be employed.

The performance results of the original code WHAMS3D on the Cray X–MP/48 for several input data decks are presented in Tables 5 to 7. Table 5 shows the execution

Version	Execution Time			
	buckle	cylpanel	frame	spcap
S	59.19	391.19	4048.66	88.02
SV	9.73	64.11	650.07	14.72
SVC	9.66	61.53	633.92	15.30

Table 5. Execution times for WHAMS–3D on the Cray X–MP/48 (from HPM).

No. of Concurrent CPUs (i)	Connect time (T_i) in each CPU			
	buckle	cylpanel	frame	spcap
1	3.15	19.66	162.35	5.99
2	0.59	9.92	131.54	1.88
3	2.45	12.30	147.89	4.34
4	3.47	19.74	211.34	3.08
Total exec. time	9.66	61.63	653.12	15.30
Total CPU time	25.54	155.37	1714.46	35.10

Table 6. Execution times for SVC version of WHAMS–3D on the Cray X–MP/48 (from HPM).

Version	MFLOPS			
	buckle	cylpanel	frame	spcap
S	11.36	11.50	11.37	11.25
SV	71.00	71.65	72.65	69.05
SVC	71.38	74.49	76.97	66.56

Table 7. MFLOPS for WHAMS–3D on the Cray X–MP/48 (from HPM).

time for all four data sets. As can be seen from this table, the vector speedup of the SV version over the S version ranges from 5.98 for the data set *spcap* to 6.23 for *frame*. It is also observed that the SVC version does not yield good performance although four CPUs are available.

We have started using tools developed at CSRD to capture the detailed histories of routine invocation together with machine performance statistics. The goal is to study performance behavior at a more refined level using trace data of routine entry and exit actions. We used tracing tools developed for the Cray X–MP and Cray 2 which are described in [MaLR90]. In summary, these tools can capture detailed histories of routine invocation together with machine performance statistics. The produced trace graphs visually depict where time is being spent in routines during the execution and the routine calling dynamics as the application proceeds.

Part of our work has concentrated on the effectiveness of using software tools to detect opportunities for parallelism. It is clear that the greatest potential for improvements, and consequently the biggest challenge, lies in the substitution of the key algorithms in the application code with redesigned algorithms which exploit the new architectures and use better numerical techniques. This will constitute a major activity of this project. We have already started work in evaluating the effect of numerical algorithms more amenable to parallelization and their effect on the codes. For example, we are studying the behavior of and alternatives to the use of CG with incomplete Cholesky preconditioner in COMMIX.

Overall the first phase has shown that significant improvements in the codes' performance result from vectorization. This is partly because of the effectiveness of vectorizing compilers, and partly because the principles of vectorization have been available to programmers for more than 15 years. The results of the first phase also show that applying existing automatic restructurers for multiprocessing gives little improvement or even degradation in performance, but provides clues on how to achieve better results.

Perfect Benchmarks™

Success in the field of parallel computing must be calibrated by measuring the performance gains made on real applications programs. A number of examples in this progress report use applications codes drawn from the Perfect Benchmarks ™, which are a collection of 13 applications level Fortran programs totaling over 60,000 lines of code. The effort has been collective with early participation from academia (University of Illinois, Princeton, Caltech, Florida State), industry (Cray, IBM) and various research centers (Houston Area Research Consortium, Institute for Supercomputer Research - Tokyo, NASA-Ames).

The programs were chosen to satisfy various constraints, including portability, breadth of application areas, reasonable running times and perhaps most importantly maturity and popularity of the codes themselves (see [Berr89, CKPK90] for more discussions about the effort and references on each of the codes). It is important to note that each Perfect code represents more than a Fortran program in that it solves a well defined scientific or engineering problem. Initially, the codes were ported to and timed on 8 machines [Berr89] and subsequently the process has been repeated on more than 20 additional machines. Insight into system software and architecture is sought, not just benchmarking numbers. The Perfect Benchmark ™ program is a collaborative effort, and contributions such as new codes or new analysis techniques are welcome and encouraged at all times.

The Perfect benchmarking effort strives to collect baseline and optimized execution statistics. A baseline measurement involves running the code as is, with only those program changes necessary for porting allowed. Note that all the Perfect codes were originally written for and used on some high-performance system; thus certain performance biases were built-in to the data, as would be the case with any such real-world collection of codes. After a baseline measurement is made, the user is encouraged to undertake any kind of program rewriting to improve performance. Diaries describing program transformations, which include loop reordering, subroutine in-lining, library routine substitutions or algorithm replacement for example, are to be kept. Loosely speaking, in a baseline measurement the original Fortran program is an invariant while in an optimized execution the scientific problem is invariant.

Table 8 contains some basic information about the Perfect codes.

Perfect Performance Data

Each of the 13 Perfect Benchmark™ codes is a stand alone program that uses no external library calls and upon successful execution creates an output file containing execution times, a MFLOPS rate and the results of a validation check. Both wall clock and CPU time are recorded. The derived MFLOPS rate on all machines is based on a reference floating point operation count obtained by the Cray XMP hpm. The validation check is performed by comparing known correct output values against the values computed by the current execution. The comparison is not always exact but uses a meaningful threshold that was determined by the code's author or maintainer at the time the code was included in the benchmark suite. The validation check has turned out to be quite useful since there are numerous reported incidents involving precision problems and incorrect code generation by buggy compilers, even though a code may compile, execute and terminate.

The Center for Supercomputing Research and Development at the University of Illinois publishes periodic reports containing tabulations of results. The last report contained over 40 tables of raw data on about 30 machines' performance [Poin90]. Efforts have been made to port codes to massively parallel machines but such endeavors are labor intensive and only two codes have been run successfully on hypercube-type machines at Caltech. The Perfect Benchmark™ database contains no results on SIMD machines for the same reason.

Observations

The Perfect Benchmark™ data demonstrates the complexity of computer systems performance. Put another way, no single number captures performance in any meaningful way and, conversely, efforts to quantify performance by a single number are intrinsically oversimplified. However, some general observations based on the totality of data are possible.

1. *Delivered performance is significantly lower than peak performance and is growing at a slower rate.* Figure 6 is a scatterplot of Perfect code performances on various supercomputers of the 1980's. It clearly illustrates that the correlation between peak performance and delivered performance is weak. It should be noted that the Perfect Benchmarks™ are by and large complete applications, in some cases with scaled down problem sizes that may lead to poorer performance. Machine performance on more homogeneous, simpler algorithms could be significantly higher. This gap between peak and delivered performance reaches two orders of magnitude in Figure 6 and will become even more dramatic as machines with more processors and/or functional units become available.

2. *Increased performance leads to increased performance variations.* Variations in performance, quantified as the ratio between maximum and minimum delivered megaflops, are correlated with peak performance as can be seen from Figure 6. Calling this ratio *instability*, higher performance machines appear to be more unstable because of advanced hardware features that are difficult to take advantage of uniformly. Note that instabilities also approach two orders of magnitude in Figure 6. Further development of the notion of instability is currently underway at CSRD.

3. *Porting applications level codes to hypercubes and SIMD machines is difficult so that comprehensive applications-based benchmarking of these classes has not been possible to date.* After three years, only two Perfect codes have been successfully ported to hypercube machines and none to SIMD machines. This is a particularly troubling fact since much of the current enthusiasm over existing massively parallel machines is based on the performance of simple algorithms.

Conclusion

As stated throughout this report, much progress has been made not only on the Cedar hardware, operating system, applications and algorithms, but also in advancing the state of the art. Further emphasis must be placed on much that is left to do. We are on the verge of more breakthroughs in parallel processing which has been acknowledged to be the means to achieving truly high performance computing.

Code Name	Lines of Code	Application	Contributor	Algorithms Used 1	2	3	4	5	6	7	8	9
ADM	6105	Air pollution, Fluid dynamics	IBM	.	.	x	.	.	.	.	.	.
ARC2D	3964	Supersonic reentry, 2-D fluid dynamics	NASA Ames	x	.	.	x	.	.	[illegible]	.	.
BDNA	3977	Nucleic acid simulation, Molecular dynamics	IBM	.	.	.	.	.	x	.	.	.
DYFESM	7608	Structural dynamics, Engineering design	CSRD	x	x	.	.	.	x	.	.	.
FLO52	1986	Transonic flow, 2-D Fluid dynamics	Princeton	.	.	.	.	x	x	.	.	.
MDG	1238	Liquid water simulation, Molecular dynamics	IBM	.	.	.	.	.	x	.	.	.
MG3D	2812	Seismic migration, Signal processing	Cray	.	.	x	.	.	x	.	.	.
OCEAN	4343	Ocean simulation, 2-D fluid dynamics	Princeton	.	.	x	.	.	.	.	.	.
QCD	2327	Lattice gauge, Quantum chromodynamics	Caltech	.	.	.	.	.	.	x	.	.
SPEC77	3885	Weather simulation, Fluid dynamics	CSRD	.	.	x	x	.	.	.	.	.
SPICE	18521	Circuit simulation, Engineering design	CSRD	x	.	.	.	.	x	.	.	.
TRACK	3784	Missile tracking, Signal Processing	Caltech	.	.	.	.	.	.	.	.	x
TRFD	485	2-electron transform integrals, Molecular dynamics	IBM	.	.	.	.	.	.	.	x	.

Algorithms:

1. Sparse Linear Systems Solvers
2. Nonlinear Algebraic System Solvers
3. Fast Fourier Transforms
4. Rapid Elliptic Problem Solvers
5. Multigrid Schemes
6. Ordinary Differential Equation Solvers
7. Monte Carlo Schemes
8. *Integral Transforms*
9. Convolution

Table 8. The Perfect Benchmarks ™.

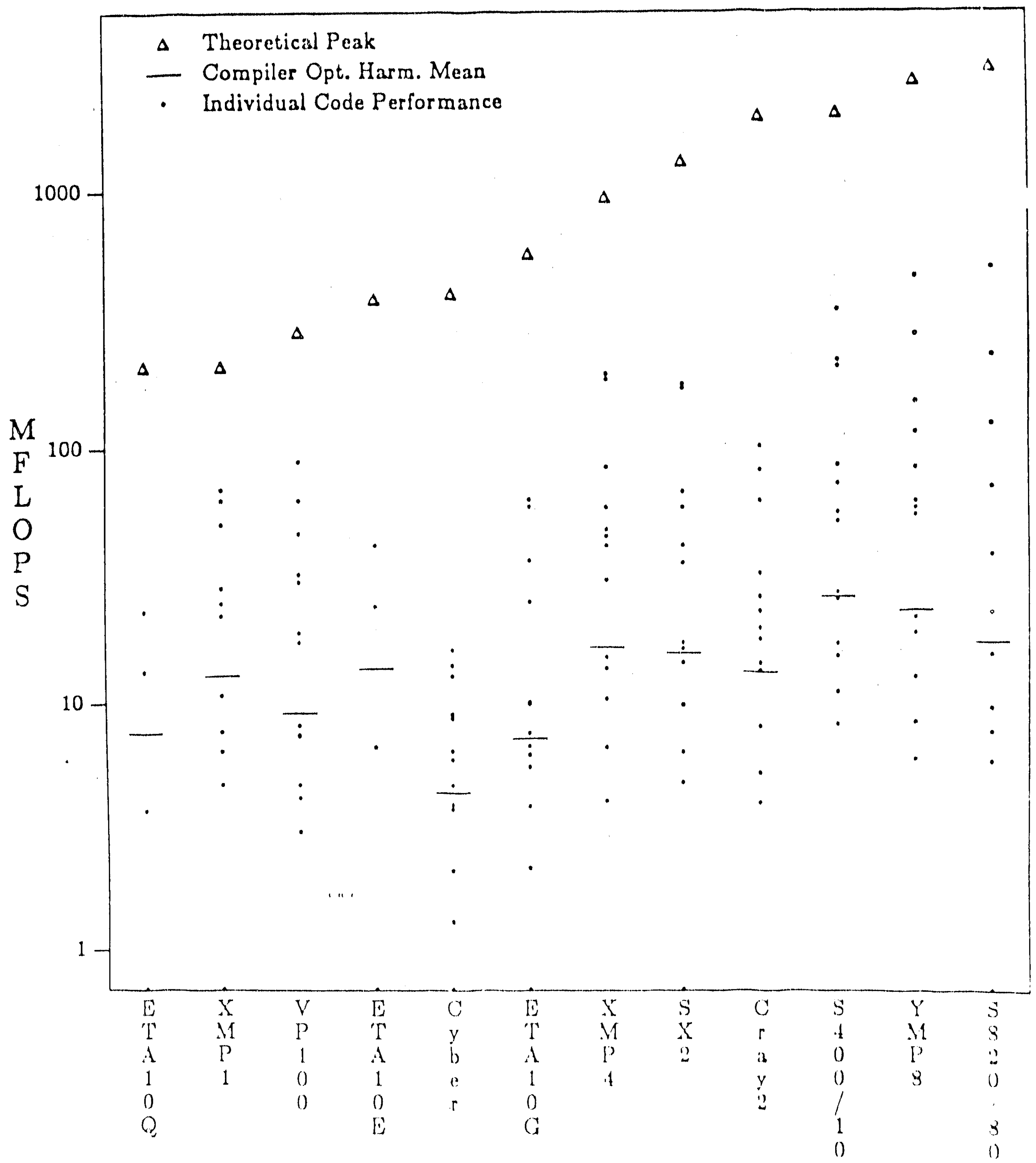


Figure 6. Instability Scatter Plot.

Center for Supercomputing Research and Development
REFERENCES

[ABCG90] S. Aslam, R. Bramley, H.-C. Chen, G. Cybenko, et.al. "The Advanced Software Development and Commercialization Project", CSRD Rept. No. 1047, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., Sept. 1990.

[Berr90] Michael Waitsel Berry. "Multiprocessor Sparse SVD Algorithms and Applications", CSRD Report No. 1049 UILU-ENG-90-8031, Univ. of Illinois at Urbana-Champaign, Cntr. for Supercomputing Res. & Dev., November 1990.

[Berr89] M. Berry et al., *The Perfect Club Benchmarks: Effective performance evaluation of supercomputers*, The International Journal of Supercomputer Applications, 3 (1989), pp. 5-40.

[BeTs82] T. Belytschko and C.S. Tsay. "WHAMSE: A Program for Three-Dimensional Nonlinear Structural Dynamics". Research Project 1065-3, NP-2250, Dept. of Civil Engineering, Northwestern University, Evanstan, IL, Feb. 1982.

[BlGG89] R.N. Blomquist, P.L. Garner, E.M. Gelbard. "Code Abstract for COMMIX-1AR/P". Computing and Telecommunications Div., Argonne National Laboratory, July 1989.

[Bram89] Randall B. Bramley. "Row Projection Methods for Linear Systems", CSRD Report No. 881, UILU-ENG-89-8008, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1989.

[Bram90] R. Bramley. "RP-Pack: A Tool for Examining Row Projection Methods", CSRD Report No. 1008, Univ. of Illinois at Urbana-Champaign, Ctr. for Supercomputing Res. and Dev., May 1990.

[BrDM88] F. Brezzi, C. Douglas, L. Marini. "Parallel Domain Reduction Method", IBM Research Report RC 13778.

[BrSa90a] Randall Bramley and Ahmed Sameh. *Row Projection Methods for Large Nonsymmetric Linear Systems.* To appear SIAM Journal Scientific Statistical Computing, January 1990.

[CGPW89] T. Chan, R. Glowinski, J. Periaux, O. Widlund, et.al. (editors). *Domain Decomposition Methods*, SIAM, Philadelphia, 1989. Proceedings of the Second International Symposium on Domain

Decomposition Methods, Los Angeles, CA, January 1988.

[Chen90] Hsin–Chu Chen. *Two Special Classes of Matrices.* **Submitted for publication**, March 1990.

[Chen88] Hsin–Chu Chen. "The SAS Domain Decomposition Method for Structural Analysis", CSRD Report No. 754, UILU–ENG–88–8003, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., March 1988.

[ChSa89a] Hsin–Chu Chen and Ahmed Sameh. *A Matrix Decomposition Method for Orthotropic Elasticity Problems.* **SIAM Jour. on Matrix Analysis and Applications**, Vol. 10, No. 1, pp. 39–64, 1989.

[ChSa89b] Hsin–Chu Chen and Ahmed Sameh. *A Domain Decomposition Method for 3D Elasticity Problems.* **Applications of Supercomputers in Engineering: Fluid Flow and Stress Analysis Applications**, pp. 171–188, September 1989.

[CKPK90] G. Cybenko, L. Kipp, L. Pointer, and D. Kuck, *Supercomputer performance evaluation and the Perfect Benchmarks$"" sup TM$*, Tech. Rep. 965, Center for Supercomputing Research and Development, University of Illinois at Urbana, Technical Report, 1990.

[DoSm88] C. Douglas and B. Smith. *Using Symmetries and Antisymmetries to Analyze a Parallel Multigrid Algorithm: the Elliptic Boundary Value Problem Case.* **To appear in the special issue of SIAM J. Numer. Anal.** in honor of Jim Douglas, Jr.'s 60th birthday.

[Fran90] George N. Frank. "Experiments on the Cedar Multicluster with Parallel Block Cyclic Reduction and an Application to Domain Decomposition Methods", CSRD Report No. 1056 UILU–ENG–90–8032, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., November 1990.

[GaFM89] E. Gallopoulos, G. Frank, U. Meier. *Experiments with Elliptic Problem Solvers on the Cedar Multicluster.* **To appear in Proc. of Fourth SIAM Conf. Par. Proc. Sci. Comput.**, Chicago, IL, December 1989

[GaSa89c] E. Gallopoulos and Ahmed Sameh. *Solving Elliptic Equations on the Cedar Multiprocessor.* In: **Aspects of Computation on Asynchronous Parallel Processors**, M.H. Wright, ed. Elsevier Science Pub. B.V. (North–Holland), pp. 1–12, 1989.

[GaMW90] K. Gallivan, B. Marsolf, H. Wijshoff. *A Large Grain Parallel Sparse System Solver.* In: **Proceedings of the Fourth Siam Conf. on Parallel Processing for Scientific Computing**, J. Dongarra, P. Messina, D.C. Sorensen, R.G. Voigt, eds. SIAM, pp. 23–28, 1990.

[GGMP88] R. Glowinski, G. Golub, G. Meurant, and J. Periaux (editors). *Domain Decomposition Methods for Partial Differential Equations.* SIAM, Philadelphia, 1988. **Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations**, Paris, France, January 1987.

[GPHL90] Mark D. Guzzi, David A. Padua, Jay P. Hoeflinger and Duncan H. Lawrie. *Cedar Fortran and Other Vector and Parallel Fortran Dialects.* **Jour. of Supercomputing**, Vol. 4, No. 1, pp. 37–62, March 1990.

[Guzz87] Mark Guzzi. "Cedar Fortran Programmer's Handbook", CSRD Report No. 601, Univ. of Illinois at Urbana-Champaign, Ctr. for Supercomputing Res. & Dev., June 1987.

[HaPa88] W. Ludwell Harrison III and David Padua. *PARCEL: Project for the Automatic Restructuring and Concurrent Evaluation of Lisp.* **Proc. of 1988 Int'l. Conf. on Supercomputing, St. Malo, France**, pp. 527–538, July 1988.

[LiYe88a] Zhiyuan Li and Pen-Chung Yew. *Interprocedural Analysis for Parallel Programs.* **Proc. of 1988 Int'l. Conf. on Parallel Processing: Vol. II Software, St. Charles, IL**, pp. 221–228, August 1988.

[LiYe88b] Zhiyuan Li and Pen-Chung Yew. *Program Parallelization with Interprocedural Analysis.* **Jour. of Supercomputing**, Vol. 2, pp. 225–244, 1988.

[LiYe88c] Zhiyuan Li and Pen-Chung Yew. *Efficient Interprocedural Analysis for Program Parallelization and Restructuring.* **Proc. of ACM/SIGPLAN PPEALS, New Haven, CT**, pp. 85–99, July 1988.

[MaLR90] A. Malony, J. Larson, D. Reed. "Tracing Application Program Execution on the Cray X-MP and Cray 2". CSRD Rept. No. 985, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., Nov. 1990.

[McEi90] U. Meier and R. Eigenmann. "Parallelization and Performance of Conjugate Gradient Algorithms on the Cedar Hierarchical Memory Multiprocessor", CSRD Report No. 1035, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., 1990.

[MiPa90] Sam Midkiff and David Padua. *Issues in the Compile-Time Optimization of Parallel Programs.*. **Proc. of Int'l Conf. on Parallel Processing 1990,** Vol. II, pp. 105–113, August 1990.

[NoPe87a] A. Noor and J. Peters. *Preconditioned Conjugate Gradient Techniques for the Analysis of Symmetric Anisotropic Structures.* **IJNME**, Vol. 24, pp. 2057-2070, 1987.

[NoPe87b] A. Noor and J. Peters. *Model-size Reduction for the Nonlinear Dynamic Analysis of Quasi-symmetric Structures.* **Engineering Computations**, Vol. 4, No. 4, pp. 178–189, September 1987.

[Poin90] L. Pointer, *Perfect: Performance Evaluation for Cost-effective Transformations Report 2*, Tech. Rep. 964, Center for Supercomputing Research and Development, University of Illinois at Urbana, Technical Report, 1990.

[Veid85] Alexander Veidenbaum. "Compiler Optimizations and Architecture Design Issues for Multiprocessors", CSRD Rpt. No. 85-520, UILU-ENG-85-8012, Univ. of Illinois at Urbana–Champaign, Dept. of Comput. Sci., May, 1985.

[Wijs89b] Harry A. G. Wijshoff. "Symmetric Orderings for Unsymmetric Sparse Matrices", CSRD Report No. 901, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1989.

[Yang90] G.-C. Yang. "PARASPICE: A Parallel Direct Circuit Simulator for Shared-Memory Multiprocessors", Univ. of Illinois at Urbana–Champaign, Ctr. for Supercomputing Res. and Dev., January 1990.

[CSRD494*] CSRD internal report.

[CSRD857*] CSRD internal report.

[CSRD858*] CSRD internal report.

Center for Supercomputing Research and Development Bibliography DOE DE–FG02–85ER25001

[AbPa90] Seth Abraham and Krishnan Padmanabhan. *Constraint Based Evaluation of Multicomputer Networks.* **Proc. 1990 Int'l Conference on Parallel Processing**, Vol. 1, pp. 521–525, August 13–17 1990.

[Abra90] Seth Abraham. "Issues in the Architecture of Direct Interconnection Schemes for Multiprocessors", CSRD Report No. 977 UILU-ENG-90-8019, Univ. of Illinois at Urbana-Champaign Center for Supercomputing Res. & Dev., March 1990.

[AmHa90] Zahira Ammarguellat and Luddy Harrison. *Automatic Recognition of Induction & Recurrence Relations by Abstract Interpretation..* **Proc. of Sigplan 1990, Yorktown Heights**, Vol. 25, No. 6, pp. 283–295, June 1990.

[Amma90] Zahira Ammarguellat. *A Control-Flow Normalization Algorithm and its Complexity.* **Submitted for publication**, June 1990.

[BCVY90] John Bruner, Hoichi Cheong, Alexander Veidenbaum and Pen-Chung Yew. *Chief: A Parallel Simulation Environment for Parallel Systems.* **Submitted for publication**, November 1990.

[BePo90] Carl Beckmann and Constantine Polychronopoulos. *Fast Barrier Synchronization Hardware.* **Proc. Supercomputing '90, New York, NY**, pp. 180–189, November 12–16, 1990.

[Berr90] Michael Waitsel Berry. "Multiprocessor Sparse SVD Algorithms and Applications", CSRD Report No. 1049 UILU-ENG-90-8031, Univ. of Illinois at Urbana-Champaign, Cntr. for Supercomputing Res. & Dev., November 1990.

[BlBG90] B. Bliss, M. C. Brunet and E. Gallopoulos. *Automatic Parallel Program Instrumentation with Applications in Performance and Error Analysis.* **To appear in Expert Systems for Numerical Computering**, June 1990.

[BrSa90] Randall Bramley and Ahmed Sameh. *Row Projection Methods for Large Nonsymmetric Linear Systems.* **To appear SIAM Journal Scientific Statistical Computing**, January 1990.

[BrSa90] Randall Bramley and Ahmed Sameh. *Domain Decomposition for Parallel Row Projection Algorithms.* **To appear in Applied Numerical Mathematics**, January 1990.

[Brun90] John Bruner. "PARSIM User Interface Tutorial", CSRD Report No. 1040, Univ. of Illinois at Urbana-Champaign, Cntr. for Supercomputing Res. & Dev., September 1990.

[Chen90] Hsin-Chu Chen. *Two Special Classes of Matrices.* **Submitted for publication**, March 1990.

[Chen90] Mei-Qin Chen. *A Parallel Quasi-Newton Method For Partially Separable Large-Scale Minimization.* **Presented at ORSA/TIMS Conference**, October 1990.

[ChHa90] Jyh-Herng Chow and Luddy Harrison. *Switch-Stacks: A Scheme for Microtasking Nested Parallel Loops.* **Proc. of Supercomputing '90, New York, NY**, pp. 190-199, November 12-16, 1990.

[Chow90] Jyh-Herng Chow. "Parallel Execution of Lisp Programs in the Parcel Run Time Environment", CSRD Report No. 991, UILU-ENG-90-8021, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1990.

[ChSa90] Mei-Qin Chen and Ahmed Sameh. *Conjugate Subspaces Decomposition and Its Application in Solving Linear Systems with Many Right-Hand Sides.* **The Fourth Auburn Linear Algebra Conference, March 20-23, 1990**, March 1990.

[ChVe90] Hoichi Cheong and Alexander Veidenbaum. *Compiler-directed Cache Management in Multiprocessors.* **IEEE Computer Journal**, Vol. 23, No. 6, pp. 39-47, June 1990.

[CKPK90] George Cybenko, Lyle Kipp, Lynn Pointer and David Kuck. "Supercomputer Performance Evaluation and the Perfect Benchmarks^{TM}", CSRD Report No. 965, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., March 1990.

[DaYe90] Timothy Davis and Pen-Chung Yew. *A Stable Non-deterministic Parallel Algorithm for General Unsymmetric Sparse LU Factorization.* **SIAM J. on Matrix Analysis and Applications**, Vol. 2, No. 3, pp. 383-403, July 1990.

[EHJP90] R. Eigenmann, J. Hoeflinger, G. Jaxon and D. Padua. *Cedar Fortran and Its Compiler.* **Lecture Notes in Computer Science: Proc. of the Joint Conf. on Vector and Parallel Processing, Zurich, Switzerland**, Vol. 457, pp. 288-200, January 1990.

[Eijk90] Victor Eijkhout. *Analysis of Parallel Incomplete Point Factorizations.* **Submitted for publication**, October 1990.

[Eijk90] Victor Eijkhout. "Beware of Modified Incomplete Point Factorizations", CSRD Report No. 1048, Univ. of Illinois at Urbana–Champaign, Cntr. for Supercomputing Res. & Dev., October 1990.

[EJJP90] R. Eigenmann, J. Hoeflinger, G. Jaxon and D. Padua. *Cedar Fortran and its Restructuring Compiler.* In: **Languages and Compilers for Parallel Computing II**, T. Bross D. Gekruter A. Nicolau and D. Padua, ed. MIT Press, 1990.

[Fran90] George N. Frank. "Experiments on the Cedar Multicluster with Parallel Block Cyclic Reduction and an Application to Domain Decomposition Methods", CSRD Report No. 1056 UILU–ENG–90–8032, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., November 1990.

[FuCh90] Chuigang Fu. "Evaluating the Effectiveness of Fortran Vectorizers by Measuring Total Parallelism", CSRD Report No. 1033, UILU–ENG–90–8029, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1990.

[GaBe90] Hui Gao and Michael Berry. "Performance Studies of LAPACK on Alliant FX/80 and 1 Cedar Cluster", CSRD Report No. 1001, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., May 1990.

[GaPS90] K. A. Gallivan, R. J. Plemmons and A. H. Sameh. *Parallel Algorithms for Dense Linear Algebra Computations.* **SIAM Review**, Vol. 32, No. 1, pp. 54–135, March 1990.

[GaSa90] E. Gallopoulos and Y. Saad. *Efficient Solution of Parabolic Equations by Polynomial Approximation Methods.* **Submitted for Publication**, February 1990.

[GaSZ90] Kyle Gallivan, Ahmed Sameh and Zahari Zlatev. *Solving General Sparse Linear Systems Using Conjugate Gradient-type Methods.* **Proc. 1990 Int'l. Conf. on Supercomputing, Amsterdam, Netherlands, ACM Press**, pp. 132–139, June 1990.

[GoGV90] Edward H. Gornish, Elana D. Granston and Alexander V. Veidenbaum. *Compiler-directed Data Prefetching in Multiprocessors with Memory Hierarchies.* **Proc. of ICS'90, Amsterdam, The Netherlands**, Vol. 1, pp. 354–368, May 1990.

[GPHL90] Mark D. Guzzi, David A. Padua, Jay P. Hoeflinger and Duncan H. Lawrie. *Cedar Fortran and Other Vector and Parallel Fortran Dialects.* **Jour. of Supercomputing**, Vol. 4, No. 1, pp. 37–62, March 1990.

[GrVe90] Elana D. Granston and Alexander V. Veidenbaum. *Detecting Opportunities for Data Reuse.* **Submitted for Publication**, November 1990.

[HaAm90] Luddy Harrison and Zahira Ammarguellat. *A Comparison of Automatic versus Manual Parallelization of the Boyer–Moore Theorem Prover.* **To appear in Monographs in Parallel & Distributed Computing, Chris Jesshope & David Klappholz, eds., Pitman Publishing**, 1990.

[HsYe90] William Tsun–yuk Hsu and Pen–Chung Yew. *An Effective Synchronization Network for Large Multiprocessor Systems.* **Subm. the 5th Int'l Parallel Processing Symp., Disneyland Hotel Convention Center, April 30 – May 2, 1991**, September 1990.

[HWGS90] G. G. Hung, Y. C. Wen, K. Gallivan and R. Saleh. *A Parallel Circuit Simulation using Hierarchical Relaxation.* **Proc. 27th Design Automation Conf.**, pp. 394–399, June 1990.

[KoCG90] C. K. Koc, P. R. Cappelo and E. Gallopoulos. *Decomposing Polynomial Interpolation for Systolic Arrays.* **To appear in Int'l Journal of Computer Mathematics**, July 1990.

[KrCV90] David W. Krumme, George Cybenko and K. N. Venkataraman. *Gossiping in Minimal Time.* **To appear in SIAM Journal on Computing**, August 1990.

[Leun90] Bruce Paul Leung. "Issues on the Design of Parallelizing Compilers", CSRD Report No. 1012, UILU-ENG-90-8027, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., June 1990.

[LiYe90] David J. Lilja and Pen–Chung Yew. "The Interaction of Cache Block Size and Parallel Loop Scheduling Strategy", CSRD Report No. 989, Univ. of Illinois at Urbana–Champaign, Cntr. for Supercomputing Res. & Dev., July 1990.

[LiYe90] David Lilja and Pen–Chung Yew. *The Performance Potential of Fine-Grain and Coarse-Grain Parallel Architecture.* **Submitted for publication**, June 1990.

[LiYe90] David J. Lilja and Pen–Chung Yew. "A Compiler–Assisted Directory–Based Cache Coherence Scheme", Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., November 1990.

[MiPa90] Sam Midkiff and David Padua. *Issues in the Compile-Time Optimization of Parallel Programs.* **Proc. of Int'l Conf. on Parallel Processing 1990**, Vol. II, pp. 105–113, August 1990.

[Padu89] David Padua. *Problem Solving Environments for Parallel Computing.* **Proc. of an International Conf. organized by the IPSJ to Commemorate the 30 Anniversary**, pp. 323–330, November 1990.

[PoJa90] David Pointer and Greg Jaxon. "Cedar Synchronization Processor Instruction Set Reference", CSRD Report No. 1017, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res & Dev., July 1990.

[Saad90] Youcef Saad. "SPARSKIT: A Basic Tool Kit for Sparse Matrix Computation", CSRD Report No. 1029, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1990.

[Scho90] Dale Allan Schouten. "An Overview of Interprocedural Analysis Techniques for High Performance Parallelizing Compilers", CSRD Report No. 1005, UILU-ENG-90-8025, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1990.

[ShLY90] Zhiyu Shen, Zhiyuan Li and Pen-Chung Yew. *An Empirical Study of Fortran Programs for Parallelizing Compilers.* **IEEE Trans. on Parallel and Distributed Systems**, pp. 350–364, July 1990.

[SMBS90] Sanjay Sharma, Allen Malony, Michael Berry and Priyanvada Sinvhal-Sharma. *Run-Time Monitoring and Performance Visualization of Concurrent Programs.* **Proc. of Supercomputing 90, November 1990, New York, NY**, pp. 784–793, November 12–16, 1990.

[TuBe90] Allan Tuchman and Michael Berry. *Matrix Visualization in the Design of Numerical Algorithms.* **ORSA Journal of Computing**, Vol. 2, No. 1, 1990.

[YeBr90] Pen-Chung Yew and John Bruner. *SEE: A System Evaluation Environment for Studying Parallel Systems.* **To be presented at the First Workshop on Parallel Processing, National Tsing Hua, Univ., Taiwan, Dec. 20–21, 1990**, December 1990.

[YiKw90] Kwang Keun Yi. "On-the-Fly Methods to Measure the Locality of Programs", CSRD Report No. 988, UILU-ENG-90-8020, Univ. of Illinois at Urbana-Champaign Center for Supercomputing Res. & Dev., May 1990.

[AbPa89] Seth Abraham and Krishnan Padmanabhan. *An Analysis of the Twisted Cube Topology.* **Proc. of 1989 Int'l. Conf. on Parallel Processing, St. Charles, IL**, Vol. I, pp. 116–120, August 1989.

[AbPa89] Seth Abraham and Krishnan Padmanabhan. *An Analysis of the Star Graph Topology for Multicomputer Systems.* **Subm. for publ. to 10th Int'l. Conf. on Dist. Computing Systems, May 1990, Paris, France,** October 1989.

[Amma89] Zahira Ammarguellat. "Normalization of Program Control Flow", CSRD Report No. 885, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., May 1989.

[AnSa89] Edward Anderson and Youcef Saad. *Solving Sparse Triangular Linear Systems on Parallel Computers.* **Int'l. Journal of High Speed Computing,** Vol. 1, No. 1, pp. 73–95, 1989.

[BCGM89] Michael Berry, Hsin–Chu Chen, E. Gallopoulos, Ulrike Meier, Allan Tuchman, Harry Wijshoff and Gung–Chung Yang. *Algorithmic Design on the Cedar Multiprocessor.* **Proc. of 4th Int'l. Conf. on Supercomputing & 3rd World Computer Exhibition, Santa Clara, CA**, Vol. 27, pp. 78–88, April 1989.

[BCKK89] M. Berry, D. Chen, P. Koss, D. Kuck, L. Pointer, S. Lo, Y. Pang, R. Roloff, A. Sameh, E. Clementi, S. Chin, D. Schneider, G. Fox, P. Messina, D. Walker, C. Hsiung, J. Schwarzmeier, K. Lue, S. Orszag and F. Seidl. *The Perfect Club Benchmarks: Effective Performance Evalution of Supercomputers.* **Int'l. Jour. of Supercomputer Applications, Fall 1989**, Vol. 3, No. 3, pp. 5–40, Fall 1989.

[BCMS89] Randall Bramley, Hsin–Chu Chen, Ulrike Meier and Ahmed Sameh. *On Some Parallel Preconditioned CG Schemes.* **To appear in Proc. of Conf. on Preconditioned Conjugate Gradient Methods**, 1989.

[BePo89] Carl J. Beckmann and Constantine Polychronopoulos. *The Effect of Barrier Synchronization and Scheduling Overhead on Parallel Loops.* **Proc. of 1989 Int'l. Conf. on Parallel Processing, St. Charles, IL**, Vol. II, pp. 200–204, August 1989.

[BeSa89] Michael Berry and Ahmed Sameh. *Parallel Algorithms for the Singluar Value and Dense Symmetric Eigenvalue Problems.* **Jour. of Computational and Applied Mathematics**, Vol. 27, pp. 191–213, 1989.

[BiEm89] Eric J. Bina and Perry A. Emrath. *A Faster fsck for BSD Unix.* **Proc. of**

USENIX Technical Conference, Winter 1989, San Diego, CA, pp. 173–185, Winter 1989.

[BlBG89] Brian Bliss, Marie–Christine Brunet and E. Gallopoulos. *Automatic Parallel Program Instrumentation: Applications in Performance and Error Analysis.* **To appear in Proc. of 2nd ACM Int'l. Conf. on Expert Systems for Numerical Computing, W. Lafayette, IN, April 1990**, December 1989.

[Blis89] Brian E. Bliss. "Instrumentation of Fortran Programs for Automatic Roundoff Error Analysis and Performance Evaluation", CSRD Report No. 936, UILU–ENG–89–8015, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., October 1989.

[Bram89] Randall B. Bramley. "Row Projection Methods for Linear Systems", CSRD Report No. 881, UILU–ENG–89–8008, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1989.

[Chen89] Ding–Kai Chen. "MaxPar: An Execution Driven Simulator for Studying Parallel Systems", CSRD Report No. 917, UILU–ENG–89–8013, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., October 1989.

[Cheo89] Hoichi Cheong. "Compiler–Directed Cache Coherence Strategies for Large–Scale Shared–Memory Multiprocessor Systems", CSRD Report No. 953, UILU–ENG–89–8018, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., December 1989.

[ChSa89] Hsin–Chu Chen and Ahmed Sameh. *A Matrix Decomposition Method for Orthotropic Elasticity Problems.* **SIAM Jour. on Matrix Analysis and Applications**, Vol. 10, No. 1, pp. 39–64, 1989.

[ChSa89] Hsin–Chu Chen and Ahmed Sameh. *A Domain Decomposition Method for 3D Elasticity Problems.* **Applications of Supercomputers in Engineering: Fluid Flow and Stress Analysis Applications**, pp. 171–188, September 1989.

[ChSY89] Ding–Kai Chen, Hong–Men Su and Pen–Chung Yew. *The Impact of Synchronization and Granularity on Parallel Systems.* **To appear in the Proc. of the 17th Int'l. Symp. on Computer Architecture, Seattle, WA**, December 1989.

[ChVe89] Hoichi Cheong and Alexander Veidenbaum. *A Version Control Approach to Cache Coherence.* **Proc. of 1989 Int'l. Conf. on Supercomputing**, pp.

322–330, June 1989.

[Cybe89] George Cybenko. *Approximations by Superpositions of a Sigmoidal Function.* **Mathematics of Control, Signals & Systems (1989)**, Vol. 2, pp. 303–314, August 1989.

[CyBe89] George Cybenko and Michael Berry. *Hyperbolic Householder Algorithms for Factoring Structured Matrices.* **To appear in SIAM Jour. on Matrix Analysis and Applications**, May 1989.

[Cybe89] George Cybenko. *Mathematical Problems in Neural Computing.* **Proc. of Int'l. Symp. on Mathematical Theory of Networks and Systems**, June 1989.

[Cybe89] George Cybenko. *Designing Neural Networks.* **Proc. First Annual IEEE Int'l. Conf. on Artificial Neural Networks**, pp. 1–3, October 1989.

[Davi89] Timothy Alden Davis. "A Parallel Algorithm for Sparse Unsymmetric Factorization", CSRD Report No. 907, UILU-ENG-89-8012, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., September 1989.

[DeGG89] Luis DeRose, Kyle Gallivan and E. Gallopoulos. "Trace Analysis of the GFDL Ocean Circulation Model: A Preliminary Study", CSRD Report No. 863, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., April 1989.

[EmGP89] Perry Emrath, Sanjoy Ghosh and David Padua. *Event Synchronization Analysis for Debugging Parallel Programs.* **To appear in Proc. of Supercomputing '89, Reno, Nevada**, November 1989.

[EmPY89] Perry Emrath, David Padua and Pen-Chung Yew. *Cedar Architecture and Its Software.* **Proc. of Hawaii Int'l. Conf. on System Sciences (January 1989)**, pp. 306–315, 1989.

[Emra89] Perry Emrath. *Program Laborious.* **Unix Review**, pp. 51–60, April 1989.

[GaFM89] E. Gallopoulos, G. Frank and U. Meier. *Experiments with Elliptic Problem Solvers on the Cedar Multicluster.* **Proc. of Fourth SIAM Conf. Par. Proc. Sci. Comput., Chicago, IL**, pp. 245–250, December 1989.

[GaGJ89] Yogesh Gaur, Vincent A. Guarna Jr. and David Jablonowski. *An Environment for Performance Experimentation on Multiprocessors.* **Proc. of Supercomputing '89, Reno, NV**, pp. 589–596, November 1989.

[GaGL89] Dennis Gannon, Vincent A. Guarna Jr. and Jenq Kuen Lee. *Static Analysis and Runtime Support for Parallel Execution.* **To appear in Monographs in Parallel and Distributed Computing, Chris Jesshope and David Klappholz, eds., Pitman Publishing, 1989.**, 1989.

[GaLe89] E. Gallopoulos and Daeshik Lee. *Fast Laplace Solver by Boundary Integral-Based Domain Decomposition Methods.* In: **Parallel Processing for Scientific Computing**, G. Rodrigue, ed. SIAM, pp. 141–145, 1989.

[GaSa89] E. Gallopoulos and Y. Saad. *Parallel Rapid Elliptic Solvers.* **Parallel Processing for Scientific Computing**, pp. 51–55, 1989.

[GaSa89] E. Gallopoulos and Youcef Saad. *Parallel Block Cyclic Reduction Algorithm for the Fast Solution of Elliptic Equations.* **Parallel Computing**, Vol. 10, No. 2, pp. 143–160, 1989.

[GaSa89] E. Gallopoulos and Ahmed Sameh. *Solving Elliptic Equations on the Cedar Multiprocessor.* In: **Aspects of Computation on Asynchronous Parallel Processors**, M.H. Wright, ed. Elsevier Science Pub. B.V. (North-Holland), pp. 1–12, 1989.

[GGJM88] Vincent Guarna Jr., Dennis Gannon, David Jablonowski, Allen Malony and Yogesh Gaur. *Faust: An Integrated Environment for the Development of Parallel Programs.* **IEEE Software**, July 1989.

[GJMW89] Kyle Gallivan, William Jalby, Allen Malony and Harry Wijshoff. *Performance Prediction of Loop Constructs on Multiprocessor Hierarchical.* **Proc. Third Int'l. Conf. on Supercomputing, Crete, Greece**, pp. 433–442, June 1989.

[Gorn89] Edward H. Gornish. "Compile Time Analysis for Data Prefetching", CSRD Report No. 949, UILU-ENG-89-8016, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., December 1989.

[Harr89] Luddy Harrison. "The Interprocedural Analysis and Automatic Parallelization of Scheme Programs", CSRD Report No. 860, UILU-ENG-89-8003, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1989.

[JaGu89] David Jablonowski and Vince Guarna Jr. *GMB: A Tool for Manipulating and Animating Graph Data Structures.* **Software–Practice and Experience**, Vol. 19, No. 3, pp. 283–301, March 1989.

[KaSa89] Chandrika Kamath and Ahmed Sameh. *A Projection Method for Solving*

Nonsymmetric Linear Systems on Multiprocessors. **Parallel Computing,** Vol. 9, pp. 291–312, 1989.

[KaSa89] E. Kamgnia and A. Sameh. *A Fast Elliptic Solver for Simply Connected Domains.* **Computer Physics Communications, North Holland, Amsterdam,** Vol. 55, pp. 43–69, 1989.

[KGAR89] T. Kerkhoven, A. Galick, J. H. Arends, U. Ravaioli and Y. Saad. *Efficient Numerical Simulation of Electron States in Quantum Wires.* **To appear in Journal of the American Physical Society,** October 1989.

[Koss89] Peter Koss. "Application Performance on Supercomputers", CSRD Report No. 847, UILU–ENG–89–8001, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., January 1989.

[Krau89] James John Krause. "A Graphical User Interface for the XDBX Debugger", CSRD Report No. 867, UILU–ENG–89–8004, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., May 1989.

[Kuck89] David J. Kuck. *Keynote Address 15th Annual Int'l. Symp. on Computer Architecture, May 30–June 3, 1988, Honolulu, HI.* **Computer Architecture News,** Vol. 17, No. 1, pp. 5–26, March 1989.

[LaMe89] J. Laminie and U. Meier. *Solving Navier–Stokes Equations on the Cedar Multi–Cluster System.* **Proc. 4th SIAM Conference Parallel Processing for Scientific Computing, Chicago, IL.,** pp. 174–179, December 1989.

[Lave89] Daniel M. Lavery. "The Design of a Hardware Performance Monitor for the Cedar Supercomputer", CSRD Report No. 866, UILU–ENG–89–8006, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., May 1989.

[Lilj89] David Lilja. *Efficient Generation of Poisson Distributed Random Numbers.* **Trans. of Soc. for Comp. Simulation,** Vol. 6, No. 1, pp. 31–41, 1989.

[LiMY89] David Lilja, David Marcovitz and Pen–Chung Yew. "Memory Referencing Behavior and a Cache Performance Metric in a Shared Memory Multiprocessor", CSRD Report No. 836, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., February 1989.

[LiRe89] Zhiyuan Li and Edward M. Reingold. *Solution of a Divide–and–Conquer Maximin Recurrence.* **SIAM Journal of Computing,** 1989.

[LiYZ89] Zhiyuan Li, Pen–Chung Yew and Chuan–Qi Zhu. *Data Dependence Analysis on Multi–Dimensional Array References.* **Proc. of 1989 Int'l. Conf. on Supercomputing**, pp. 215–224, June 1989.

[LiZh89] Zhiyuan Li. "Intraprocedural and Interprocedural Data Dependence Analysis for Parallel Computing", CSRD Report No. 910, UILU–ENG–89–8011, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1989.

[Malo89] Allen D. Malony. *JED: Just an Event Display.* **Proc. of Workshop on Performance Instrumentation for Parallel Systems, Santa Fe, NM**, June 1989.

[MiPC89] Samuel P. Midkiff, David Padua and Ronald G. Cytron. *Compiling Programs with User Parallelism.* **Research Monographs in Parallel & Distributed Computing, C. Jesshope, D. Klappholz** (eds.), **Pitman Publishing, 1989.**, 1989.

[NeTu89] Henry Neeman and Allan Tuchman. "Simulation Time Animation System", CSRD Report No. 859, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., February 1989.

[Perf89] Lynn Pointer Editor. "Perfect Report: 1", CSRD Report No. 896, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., July 1989.

[Pete89] Paul Marx Petersen. "PDE Tutor: A System for the Automatic Solution of Partial Differential Equations", CSRD Report No. 891, UILU–ENG–89–8009, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1989.

[PGHL89] Constantine Polychronopoulos, Milind Girkar, Mohammad Reza Haghighat, Chia–Ling Lee, Bruce Leung and Dale Schouten. *Parafrase–2: A New Generation Parallelizing Compiler.* **Proc. of 1989 Int'l. Conf. on Parallel Processing, St. Charles, IL**, Vol. II, pp. 39–48, August 1989.

[PoKP89] Constantine Polychronopoulos, David Kuck and David Padua. *Utilizing Multidimensional Loop Parallelism on Large–Scale Parallel Processor Systems.* **IEEE Trans. on Comp.**, Vol. 38, No. 9, pp. 1285–1296, September 1989.

[Poly89] Constantine Polychronopoulos. *Multiprocessing versus Multiprogramming.* **Proc. of 1989 Int'l. Conf. on Parallel Processing, St. Charles, IL**, Vol. II, pp. 223–230, August 1989.

[SGCH89] R. Saleh, K. Gallivan, M. Chang, I. Hajj, D. Smart and T. Trick. *Parallel Circuit Simulation on Supercomputers.* **Proc. of IEEE**, Vol. 77, No. 12, pp. 1915–1931, December 1989.

[ShLY89] Zhiyu Shen, Zhiyuan Li and Pen-Chung Yew. *An Empirical Study on Array Subscripts and Data Dependences.* **Proc. of 1989 Int'l. Conf. on Parallel Processing, St. Charles, IL**, Vol. II, pp. 145–152, August 1989.

[ShMa89] Sanjay Sharma and Allen Malony. "XCOUNT: A Tool to Interpret Counting Related Events", CSRD Report No. 897, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., June 1989.

[ShNe89] Peter Shirley and Henry Neeman. *Volume Visualization at the Center for Supercomputing Research & Development.* **Proc. of Chapel Hill Workshop on Volume Visualization**, pp. 15–19, January 1989.

[Skee89] Robert D. Skeel. "Macromolecular Dynamics on a Shared-Memory Multiprocessor", CSRD Report No. 929, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., October 1989.

[SuYe89] Hong-Men Su and Pen-Chung Yew. *On Data Synchronizations for Multiprocessor Systems.* **Proc. of 16th Annual Int'l. Symp. on Computer Architecture, Jerusalem, Israel**, pp. 416–423, May 1989.

[Tang89] Ju-Ho Tang. "Performance Evaluation of Vector Machine Architectures", CSRD Report No. 850, UILU-ENG-89-8002, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., January 1989.

[TaYZ90] Peiyi Tang, Pen-Chung Yew and Chuan-Qi Zhu. *A Parallel Linked List for Shared-Memory Multiprocessors.* **Proc. of the 13th Annual Int'l Computer Software & Application Conf.**, pp. 130–135, September 1989.

[Turn89] Stephen Wilson Turner. "Shared Memory and Interconnection Network Performance for Vector Multiprocessors", CSRD Report No. 876, UILU-ENG-89-8007, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1989.

[TuVe89] Stephen Turner and Alexander Veidenbaum. "Burst Traffic in MIN-Based Shared Memory Systems", CSRD Report No. 855, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., February 1989.

[Wart89] Nancy Jeanne Warter. "An Environment Monitor for the Cedar Supercomputer", CSRD Report No. 871, UILU-ENG-89-8005, Univ. of Illinois at

Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1989.

[Wijs89] Harry A. G. Wijshoff. "Implementing Sparse BLAS Primitives on Concurrent/Vector Processors: A Case Study", CSRD Report No. 843, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., January 1989.

[Wijs89] Harry A. G. Wijshoff. "Symmetric Orderings for Unsymmetric Sparse Matrices", CSRD Report No. 901, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1989.

[ZhFL89] Chuan-Qi Zhu, Zhixi Fang and Xiaobo Li. *A New Parallel Sorting Approach with Sorting Memory Module.* **Jour. of Parallel and Distributed Computing**, Vol. 7, pp. 482-502, 1989.

[AbDa88] Santosh Abraham and Timothy Davis. *Blocking for Parallel Sparse Linear System Solvers.* **Proc. of 1988 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 166-173, August 1988.

[Abra88] Santosh Abraham. "Reducing Interprocessor Communication in Parallel Architectures: System Configuration and Task Assignment", CSRD Report No. 726, UILU-ENG-88-8001, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., January 1988.

[Ande88] Edward Charles Anderson. "Parallel Implementation of Preconditioned Conjugate Gradient Methods for Solving Sparse Systems of Linear Equations", CSRD Report No. 805, UILU-ENG-88-8007, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1988.

[Bina88] Eric J. Bina. "Modifications to the UNIX File System Check Program FSCK for Quicker Crash Recovery", CSRD Report No. 811, UILU-ENG-88-8010, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1988.

[BrSa88] Randall Bramley and Ahmed Sameh. *A Robust Parallel Solver for Block Tridiagonal Systems.* **Proc. of 1988 Int'l. Conf. on Supercomputing, St. Malo, France, ACM Press**, pp. 39 54, July 1988.

[BrSa88] Randall Bramley and Ahmed Sameh. "A Robust Parallel Solver for Block Tridiagonal Systems", CSRD Report No. 806, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1988.

[Chen88] Hsin-Chu Chen. "The SAS Domain Decomposition Method for Structural Analysis", CSRD Report No. 754, UILU-ENG-88-8003, Univ. of Illinois at

Urbana-Champaign, Center for Supercomputing Res. & Dev., March 1988.

[Chen88] Hsin-Chu Chen. *The SAS Domain Decomposition Method.* **Presented at SIAM Conf. on Applied Linear Algebra, Madison, WI**, May 1988.

[ChVe88] Hoichi Cheong and Alex Veidenbaum. *Cache Coherence Enforcement Scheme with Fast Selective Invalidation.* **15th Int'l. Symp. on Comp. Architecture**, p. 299, June 1988.

[ChVe88] Hoichi Cheong and Alex Veidenbaum. *Detection of Stale Copy Accesses and Fast Selective Cache Coherence Enforcement Using a Flow Analysis Approach.* **Proc. of 1988 Int'l. Conf. on Parallel Processing, Vol. I: Architecture**, pp. 138-145, August 1988.

[Cont88] Thomas Martin Conte. "The Simulation and Tuning of the Global Memory Subsystem of a Multiprocessor", CSRD Report No. 821, UILU-ENG-88-8013, M.S. Thesis, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., September 1988.

[CyKM88] R. Cytron, S. Karlovsky and K. McAuliffe. *Automatic Management of Programmable Caches.* **Proc. of 1988 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 229-238, August 1988.

[DaDa88] Timothy Davis and Edward Davidson. *Pairwise Reduction for the Direct, Parallel Solution of Sparse, Unsymmetric sets of Linear Equations.* **IEEE Trans. on Computers**, Vol. 37, No. 12, pp. 1648-1654, December 1988.

[EmPa88] Perry Emrath and David Padua. *Automatic Detection of Nondeterminacy in Parallel Programs.* **Proc. of ACM SIGPLAN/SIGOPS Workshop on Parallel & Distributed Debugging, Madison, WI**, pp. 89-99, May 1988.

[GaJG88] Kyle Gallivan, William Jalby and Dennis Gannon. *On the Problem of Optimizing Data Transfers for Complex Memory Systems.* **Proc. of 1988 Int'l. Conf. on Supercomputing, St. Malo, France**, pp. 238-253, July 1988.

[GaLe88] E. Gallopoulos and Daeshik Lee. *Boundary Integral Domain Decomposition on Hierarchical Memory Multiprocessors.* **Proc. of 1988 Int'l. Conf. on Supercomputing, St. Malo, France**, pp. 488-499, July 1988.

[GGGJ88] Vincent Guarna Jr., Dennis Gannon, Yogesh Gaur and David Jablonowski. *Faust: An Environment for Programming Scientific Applications.* **Proc. of Supercomputing '88**, pp. 3-10, November 1988.

[GlPo88] Milind Girkar and Constantine Polychronopoulos. *Partitioning Programs for Parallel Execution.* **Proc. of 1988 ACM Int'l. Conf. on Supercomputing, St. Malo, France**, pp. 216–229, July 1988.

[GlPo88] Milind Girkar and Constantine Polychronopoulos. *Compiler Issues for Supercomputers.* **Proc. of Supercomputing '88**, pp. 164–173, November 1988.

[GJMS88] Kyle Gallivan, William Jalby, Ulrike Meier and Ahmed Sameh. *The Impact of Hierarchical Memory Systems on Linear Algebra Algorithm Design.* **Int'l. Jour. of Supercomputer Applications**, Vol. 2, No. 1, pp. 12–48, Spring 1988.

[GKLS88] Kyle Gallivan, Peter Koss, S. Lo and R. A. Saleh. *A Comparison of Parallel Relaxation–Based Circuit Simulation Techniques.* **Professional Program Session Record 43: Simulation of Analog and Mixed–Mode Circuits, Electro '88**, pp. 1–6, May 1988.

[GKPR88] M. M. Gooley, L. V. Kale, D. A. Padua, B. Ramkumar, U. S. Reddy, D. C. Sehr, W. W. Shu and B. W. Wah. *Prolog at the University of Illinois.* **Proc. of COMPCON '88**, pp. 68–73, 1988.

[Guar88] Vincent Guarna Jr. *A Technique for Analyzing Pointer and Structure References in Parallel Restructuring Compilers.* **Proc. of 1988 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 212–220, August 1988.

[GuGa88] Vince Guarna Jr. and Yogesh Gaur. *A Portable User Interface for a Scientific Programming Environment.* **Proc. of Siggraph Symp. on User Interface Software, Banff, Alta., Canada**, pp. 211–220, October 1988.

[HaPa88] W. Ludwell Harrison III and David Padua. *PARCEL: Project for the Automatic Restructuring and Concurrent Evaluation of Lisp.* **Proc. of 1988 Int'l. Conf. on Supercomputing, St. Malo, France**, pp. 527–538, July 1988.

[HuKP88] Harlan Husmann, David Kuck and David Padua. *Automatic Compound Function Definition for Multiprocessors.* **Proc. of 1988 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 34–41, August 1988.

[Jaya88] D. N. Jayasimha. *Distributed Synchronizers.* **Proc. of 1988 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 23–27, August 1988.

[Jaya88] Doddaballapur Narasimha–Murthy Jayasimha. "Communication and Synchronization in Parallel Computation", CSRD Report No. 819, UILU–ENG–88–8012, Ph.D. Thesis, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., September 1988.

[KaPS88] Laxmikant Kale, David Padua and David Sehr. *OR Parallel Execution of Prolog Programs with Side Effects.* **Journal of Supercomputing**, Vol. 2, pp. 209–223, 1988.

[LeeD88] Daeshik Lee. "Performance Enhancement of the Extra-Stage Multistage Interconnection Networks", CSRD Report No. 645, Univ. of Illinois at Urbana-Champaign, Cntr. for Supercomputing Res. & Dev., January 1988.

[LeLe88] Kyungsook Y. Lee and Daeshik Lee. *On the Augmented Data Manipulator Network in SIMD Environments.* **IEEE Trans. on Computers**, Vol. 37, pp. 574–584, May 1988.

[Lilj88] David Lilja. *Methods of Reducing the Branch Penalty in Pipelined Processors.* **Computer**, Vol. 21, No. 7, pp. 47–55, July 1988.

[LiYe88] Zhiyuan Li and Pen-Chung Yew. *Interprocedural Analysis for Parallel Programs.* **Proc. of 1988 Int'l. Conf. on Parallel Processing: Vol. II Software, St. Charles, IL**, pp. 221–228, August 1988.

[LiYe88] Zhiyuan Li and Pen-Chung Yew. *Program Parallelization with Interprocedural Analysis.* **Jour. of Supercomputing**, Vol. 2, pp. 225–244, 1988.

[LiYe88] Zhiyuan Li and Pen-Chung Yew. *Efficient Interprocedural Analysis for Program Parallelization and Restructuring.* **Proc. of ACM/SIGPLAN PPEALS, New Haven, CT**, pp. 85–99, July 1988.

[Malo88] Allen Malony. *Regular Processor Arrays.* **Presented at 2nd Symposium on the Frontiers of Massively Parallel Computation, Fairfax, VA**, October 1988.

[Malo88] Allen D. Malony. *Multiprocessor Instrumentation: Approaches for Cedar.* **Proc. of Workshop on Instrumentation for Future Parallel Systems**, May 1988.

[MaPi88] Allen D. Malony and Joseph R. Pickert. "An Environment Architecture and Its Use in Performance Data Analysis", CSRD Report No. 829, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., October 1988.

[Marc88] David Michael Marcovitz. "A Multiprocessor Cache Performance Metric", CSRD Report No. 813, UILU-ENG-88-8011, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1988.

[MaRe88] Allen D. Malony and Daniel A. Reed. *Visualizing Parallel Computer System Performance.* **Proc. of Workshop on Instrumentation for Future Parallel Systems, Santa Fe, NM**, May 1988.

[MeSa88] Ulrike Meier and Ahmed Sameh. *The Behavior of Conjugate Gradient Algorithms on a Multivector Processor with a Hierarchical Memory.* **Journal of Computational and Applied Mathematics**, Vol. 24, pp. 13–32, 1988.

[MiIy88] Samir G. Mitra and Ravishankar Iyer. "Measurement–Based Analysis of Multiple Errors and Near–Coincident Fault Discovery in a Shared Memory Multiprocessor", CSRD Report No. 744, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., February 1988.

[Mitr88] Samir G. Mitra. "Measurement–Based Analysis of Multiple Errors and Near-Coincident Fault Discovery on a Shared Memory Multiprocessor", CSRD Report No. 743, UILU–ENG–88–8002, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., February 1988.

[Padu88] David Padua. *The Cedar Parallel Processor: Machine Organization and Software.* **Speedup**, Vol. 2, No. 1, January 1988.

[PaHK88] D. Padua, W. L. Harrison III and D. J. Kuck. *Languages and Compilers for Parallel Symbolic Computing.* In: **Biological and Artificial Intelligence Systems**, E. Clementi and S. Chin, ed. ESCOM, Leiden, The Netherlands, pp. 453–461, 1988.

[Poly88] Constantine Polychronopoulos. *The Impact of Run–Time Overhead on Usable Parallelism.* **Proc. of 1988 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 108–112, August 1988.

[Poly88] Constantine Polychronopoulos. *Compiler Optimizations for Enhancing Parallelism and Their Impact on Architecture Design.* **IEEE Trans. on Computers**, Vol. C–37, No. 8, pp. 991–1004, August 1988.

[Poly88] Constantine Polychronopoulos. *Toward Auto–Scheduling Compilers.* **Journal of Supercomputing**, May 1988.

[ReMa88] Daniel A. Reed and Allen Malony. *Parallel Discrete Event Simulation: The Chandy-Misra Approach.* **Proc. of SCS Multiconference on Distributed Simulation, San Diego, CA**, pp. 8–13, February 1988.

[Saad88] Youcef Saad. *Preconditioning and Indefinite Linear Systems.* **Special Issue of Jour. of Computational & Applied Mathematics (CAM)**, May 1988.

[SaWi88] Youcef Saad and Harry Wijshoff. *Benchmark Package for Sparse Matrix Computations.* **Proc. of 1988 Int'l. Conf. on Supercomputing, St. Malo, France**, pp. 500–509, July 1988.

[Scro88] Jeffrey S. Scroggs. "Solution of a Parabolic Partial Differential Equation via Domain Decomposition: The Synthesis of Asymptotic and Numerical Analysis", CSRD Report No. 791, UILU-ENG-88-8005, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1988.

[Sehr88] David Christopher Sehr. "OR-Parallel Execution of Prolog Programs with Side Effects", CSRD Report No. 822, UILU-ENG-88-8014, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., October 1988.

[SRSK88] Wennie Shu, Balkrishma Ramkumar, David Sehr and Laxmicant Kale. "The REDUCE-OR Process Model for Logic Programs on Parallel Machines", CSRD Report No. 737, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., January 1988.

[Staf88] CSRD Staff. "Papers Presented at the Third Siam Conference on Parallel Processing for Scientific Computing", CSRD Report No. 747, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., February 1988.

[TaDa88] Ju-Ho Tang and Edward S. Davidson. *An Evaluation of Cray-1 and Cray X-MP Performance on Vectorizable Livermore Fortran Kernels.* **Proc. of 1988 Int'l. Conf. on Supercomputing, St. Malo, France**, pp. 510–518, July 1988.

[Tang88] Peiyi Tang. "Self-Scheduling, Data Synchronization and Program Transformation for Multiprocessor Systems", CSRD Report No. 809, UILU-ENG-88-8009, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., September 1988.

[TaYZ88] Peiyi Tang, Pen-Chung Yew and Chuan-Qi Zhu. *Impact of Self-Scheduling Orders on Performance.* **Proc. of 1988 Int'l. Conf. on Supercomputing, St. Malo, France**, pp. 593–603, July 1988.

[TzYZ88] Nian-Feng Tzeng, Pen-Chung Yew and Chuan-Qi Zhu. *Realizing Fault-Tolerant Interconnection Networks via Chaining.* **Special Issue on Fault-Tolerant Computing, IEEE Trans. on Computers**, Vol. 37, No. 4, pp. 458–462, April 1988.

[XiaE88] Eugene Zhu Xia. "Parallel Waveform-Relaxation-Newton for Circuit

Simulation", CSRD Report No. 772, UILU-ENG-88-8004, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1988.

[YewP88] Pen-Chung Yew. *Architecture of the Cedar Parallel Supercomputer.* In: **Parallel Systems and Computation**, G.S. Almasi G. Paul, ed. North-Holland, Amsterdam, pp. 137-148, 1988.

[Zhon88] Jialin Zhong. "Computer Visualization of Electron Movement in Gallium Arsenide Simulation", CSRD Report No. 797, UILU-ENG-88-8006, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1988.

[AbPa87] Santosh Abraham and Janak Patel. *Parallel Garbage Collection on a Virtual Memory System.* **Proc. of 1987 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 243-246, August 19-22, 1987.

[AlPa87] Todd R. Allen and David Padua. *Debugging Parallel Fortran on a Shared Memory Machine.* **Proc. of 1987 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 721-727, August 19-22, 1987.

[AnSa87] Edward Anderson and Youcef Saad. *Preconditioned Conjugate Gradient Methods for General Sparse Matrices on Shared Memory Machines.* **Presented at 3rd SIAM Conf. on Parallel Processing for Scientific Computing, Los Angeles, CA**, December 1-4, 1987.

[BePl87] Michael Berry and Robert Plemmons. *Algorithms and Experiments for Structural Mechanics on High Performance Architecture.* **Computer Methods in Applied Mechanics & Engrg**, Vol. 64, pp. 487-507, 1987.

[BeSa87] Michael Berry and Ahmed Sameh. *A Multiprocessor Scheme for the Singular Value Decomposition.* In: **Parallel Processing for Scientific Computing**, G. Rodrigue, ed. SIAM, pp. 67-71, 1987.

[Bren87] Glen Alan Brent. "Using Program Structure to Achieve Prefetching for Cache Memories", CSRD Rpt. No. 647, Univ. of Illinois at Urbana-Champaign, Ctr. for Supercomputing Res. & Dev., January 1987.

[BrSa87] P. N. Brown and Y. Saad. "Hybrid Krylov Methods for Nonlinear Systems of Equations", CSRD Report No. 699, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., November 1987.

[ChHa87] M. Q. Chen and S. P. Han. "A Parallel Quasi-Newton Method for Partially Separable Large Scale Minimization", CSRD Report No. 680, Univ. of

Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., April 1987.

[ChSa87] Hsin–Chu Chen and Ahmed Sameh. *Numerical Linear Algebra Algorithms on the Cedar System.* In: **Parallel Computations and Their Impact on Mechanics**, A. Noor, ed. The American Society of Mechanical Engineering, pp. 101–125, 1987.

[CHSS87] R. C. Y. Chin, G. W. Hedstrom, J. S. Scroggs and D. C. Sorensen. *Parallel Computation of a Domain Decomposition Method.* **Proc. of Sixth IMACS Int'l. Symp. on Computer Methods for Partial Differential Equations**, March 1987.

[ChVe87] Hoichi Cheong and Alex Veidenbaum. *The Performance of a Software-Managed Multiprocessor Caches on Parallel Numerical Programs.* In: **Lecture Notes in Computer Science No. 297: Proc. of First Int'l. Conf. on Supercomputing, Athens, Greece**, T.S. Papatheodorou E.N. Houstis C.D. Polychronopoulos, ed. Springer–Verlag, New York, NY, pp. 316–337, 1987.

[Defe87] Daniel Defend. "Parallel Ray Tracing in a Vector–Multiprocessing Environment", CSRD Report No. 677, UILU–ENG–87–8005, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1987.

[DoSo87] J. J. Dongarra and D. C. Sorensen. *A Fully Parallel Algorithm for the Symmetric Eigenvalue Problem.* **SIAM Journ. on Scientific & Statistical Computing, Special Issue on Parallel Processing, Norfolk Mtg.**, Vol. 8, No. 2, pp. s139–s154, March, 1987.

[FTYZ87] Zhixi Fang, Peiyi Tang, Pen–Chung Yew and Chuan–Qi Zhu. *Dynamic Processor Self-Scheduling for General Parallel Nested Loops.* **Proc. of 1987 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 1–10, August 19–22, 1987.

[GaEK87] E. Gallopoulos, O. Egecioglu and C. Koc. "Fast and Practical Parallel Polynomial Interpolation", CSRD Report No. 646, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., January 1987.

[GaJa87] Dennis Gannon and William Jalby. *The Influence of Memory Hierarchy on Algorithm Organization: Programming FFTs on a Vector Multiprocessor.* In: **Characteristics of Parallel Algorithms**, D. Gannon, L. Jamison and B. Douglas, eds. MIT Press, Cambridge, MA, 1987.

[GaJM87] K. Gallivan, W. Jalby and U. Meier. *The Use of BLAS3 in Linear Algebra on a Parallel Processor with a Hierarchical Memory System.* **SIAM J. Sci. Stat. Comput.**, Vol. 8, No. 6, pp. 1079–1084, November 1987.

[GaSa87] E. Gallopoulos and Youcef Saad. *A Parallel Block Cyclic Reduction Algorithm for the Fast Solution of Elliptic Equations.* In: **Lecture Notes in Computer Science No. 297: Proc. of First Int'l. Conf. on Supercomputing, Athens, Greece**, T.S. Papatheodorou E.N. Houstis C.D. Polychronopoulos, ed. Springer–Verlag, New York, NY, pp. 563–575, 1987.

[GiSo87] Milind Girkar and Milind Sohoni. "On Finding the Vertex Connectivity of Graphs", CSRD Report No. 669, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., May 1987.

[Guar87] Vincent A. Guarna Jr. "Analysis of C Programs for Parallelization in the Presence of Pointers", CSRD Report No. 695, UILU–ENG–87–8007, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., December 1987.

[Guar87] Vincent A. Guarna Jr. "VPC – A Proposal for a Vector Parallel C Programming Language", CSRD Report No. 666, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., June 1987.

[GuMa87] Vincent Guarna and Allen Malony. *Instrumentation for the Development of Parallel Programs.* **Presented at IEEE/ACM Sigarch Workshop on Instrumentation for Distributed Computing Systems, Sanibel Island, FL**, January 1987.

[Guzz87] Mark Guzzi. "Cedar Fortran Programmer's Handbook", CSRD Report No. 601, Univ. of Illinois at Urbana–Champaign, Ctr. for Supercomputing Res. & Dev., June 1987.

[HsYZ87] William Hsu, Pen–Chung Yew and Chuan–Qi Zhu. *An Enhancement Scheme for Hypercube Interconnection Networks.* **Proc. of to 1987 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 820–823, August 19–22, 1987.

[Jaya87] D. N. Jayasimha. *Parallel Access to Synchronization Variables.* **Proc. of 1987 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 97–100, August 19–22, 1987.

[Kamg87] Emmanuel R. Kamgnia. "The von–Neumann–Ulam Monte Carlo Method for Solving Systems of Linear Algebraic Equations", Subm. for publ. to Jour. Parallel and Dist. Comp., July 1987.

[KDLS87] David Kuck, Edward Davidson, Duncan Lawrie and Ahmed Sameh. *Parallel Supercomputing Today and the Cedar Approach.* In: **Experimental Parallel Computing Architectures**, J. J. Dongarra, ed. Elsevier Science Publishers B.V. (North-Holland), New York, NY, pp. 1–20, 1987.

[KeSa87] T. Kerkhoven and Youcef Saad. "Acceleration Techniques for Decoupling Algorithms in Semiconductor Simulation", CSRD Report No. 686, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1987.

[Koni87] Kris Gibson Konigsfeld. "An Ethernet Local Area Network Controller with Supporting Operating System Software", CSRD Report No. 668, UILU-ENG-87-8004, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1987.

[Kwok87] Alex Yuen-Wai Kwok. "A Performance Analysis of Architectural Scalability", CSRD Report No. 679, UILU-ENG-87-8006, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1987.

[LeeD87] Daeshik Lee. *The Lambda Network: A New Permutation Network.* **Proc. of 1987 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 407–410, August 19–22, 1987.

[LeeR87] Roland Lun Lee. "The Effectiveness of Caches and Data Prefetch Buffers in Large-Scale Shared Memory Multiprocessors", CSRD Report No. 670, UILU-ENG-87-8005, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1987.

[LeYL87] Roland Lee, Pen-Chung Yew and Duncan H. Lawrie. *Data Prefetching in Shared Memory Multiprocessors.* **Proc. of 1987 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 28–31, August 19–22, 1987.

[LeYL87] Roland Lee, Pen-Chung Yew and Duncan H. Lawrie. *Multiprocessor Cache Design Considerations.* **Proc. of 14th Int'l. Symp. on Computer Architecture, Pittsburgh, PA**, pp. 253–262, June 1987.

[LiAb87] Zhiyuan Li and Walid Abu-Sufah. *On Reducing Data Synchronization in Multiprocessed Loops.* **IEEE Trans. on Computers**, Vol. C-36, No. 1, pp. 105–109, Jan., 1987.

[LoPS87] Sy-Shin Lo, Bernard Philippe and Ahmed H. Sameh. *A Multiprocessor Algorithm for the Symmetric Tridiagonal Eigenvalue Problem.* **SIAM Journ. on Scientific and Statistical Computing**, Vol. 8, No. 2, pp. s155–s165, March, 1987.

[MaRM87] Allen D. Malony, Daniel A. Reed and Patrick J. McGuire. *MPF: A Portable Message Passing Facility for Shared Memory Multiprocessors.* **Proc. of 1987 Int'l. Conf. on Parallel Processing**, pp. 739–741, August 1987.

[McEm87] Robert McGrath and Perry Emrath. *Using Memory in the Cedar System.* In: **Lecture Notes in Computer Science No. 297: Proc. of First Int'l. Conf. on Supercomputing, Athens, Greece**, T.S. Papatheodorou E.N. Houstis C.D. Polychronopoulos, ed. Springer–Verlag, New York, NY, pp. 43–67, 1987.

[McGu87] Patrick John McGuire. "A Measurement–Based Study of Concurrency in a Multiprocessor", CSRD Report No. 674, UILU–ENG–87–2210, CSG–61, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., January 1987.

[Mill87] Alan Ray Miller. "Non–Preemptive Run–Time Scheduling Issues on a Multitasked, Multiprogrammed Multiprocessor with Dependencies, Bidimensional Tasks, Folding, and Dynamic Graphs", CSRD Report No. 656, UILU–ENG–87–8002, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., May 1987.

[MiPa87] Samuel Midkiff and David Padua. *Compiler Algorithms for Synchronization.* **IEEE Trans. on Computers**, Vol. C–36, No. 12, pp. 1485–1495, December 1987.

[PaGL87] David A. Padua, Vincent A. Guarna Jr. and Duncan H. Lawrie. *Supercomputer Programming Environments.* **Proc. of Symp. on Parallel Computations and Their Impact on Mechanics, Boston, MA**, pp. 55–79, December 1987.

[PoBa87] Constantine Polychronopoulos and Utpal Banerjee. *Processor Allocation for Horizontal and Vertical Parallelism and Related Speedup Bounds.* **IEEE Trans. on Computers**, Vol. 36, No. 4, April 1987.

[PoKu87] Constantine Polychronopoulos and David J. Kuck. *Guided Self–Scheduling: A Practical Scheduling Scheme for Parallel Supercomputers.* **IEEE Trans. on Computers**, Vol. C–36, No. 12, pp. 1425–1439, December 1987.

[Poly87] Constantine Polychronopoulos. *Loop Coalescing: A Compiler Transformation for Parallel Machines.* **Proc. of 1987 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 235–242, August 19–22, 1987.

[Poly87] Constantine Polychronopoulos. "Compile-Time Reduction of Interprocessor Communication for Parallel Computers", CSRD Report No. 642, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., January 1987.

[Poly87] Constantine Polychronopoulos. "More on Advanced Loop Optimizations", CSRD Report No. 667, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., September 1987.

[Poly87] Constantine Polychronopoulos. *Advanced Loop Optimizations for Parallel Computers.* In: **Lecture Notes in Computer Science No. 297: Proc. of First Int'l. Conf. on Supercomputing, Athens, Greece**, T.S. Papatheodorou E.N. Houstis C.D. Polychronopoulos, ed. Springer-Verlag, New York, NY, pp. 255-277, 1987.

[Saad87] Youcef Saad. *On the Design of Parallel Numerical Methods in Message Passing and Shared Memory Environments.* **Proc. of Int'l. Seminar on Scientific Supercomputers, Paris, France**, pp. 217-238, February 2-6, 1987.

[Scha87] Mark Johannes Schaefer. "A Polynomial Based Iterative Method for Linear Parabolic Equations", CSRD Report No. 661, UILU-ENG-87-8003, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., May 1987.

[Skee87] Robert Skeel. "Waveform Iteration and the Shifted Picard Splitting", CSRD Report No. 700, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., November 1987.

[Staf87] CSRD Staff. "Center for Supercomputing Research and Development Quarterly Report: Second Quarter 1987", CSRD Report No. 681, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1987.

[Staf87] CSRD Staff. "Center for Supercomputing Research and Development Quarterly Report: First Quarter 1987", CSRD Report No. 684, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., June 1987.

[Szcz87] Joseph Szczypek. "A Fault-Diagnosis Scheme for the Cedar Processor-Memory Interconnection Network", CSRD Report No. 716, UILU-ENG-87-8008, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., December 1987.

[TYFZ87] Peiyi Tang, Pen-Chung Yew, Zhixi Fang and Chuan-Qi Zhu. *Deadlock Prevention in Processor Self-Scheduling for Parallel Nested Loops.* **Proc. of 1987 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 11–18, August 19–22, 1987.

[UhPK87] A. Uht, C. Polychronopoulos and J. F. Kolen. *On the Combination of Hardware and Software Concurrency Extraction Methods.*, December 1–4, 1987.

[YeTL87] Pen-Chung Yew, Nian-Feng Tzeng and Duncan H. Lawrie. *Distributing Hot-Spot Addressing in Large Scale Multiprocessors.* **IEEE Trans. on Computers**, Vol. C–36, No. 4, pp. 388–395, April, 1987.

[AbDa86] Santosh G. Abraham and Edward S. Davidson. *A Communication Model for Optimizing Hierarchical Multiprocessor Systems.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 467–474, Aug. 19–22, 1986.

[AbDa86] S. Abraham and E. S. Davidson. "Task Assignment Using Network Flow Methods for Minimizing Communication in n-Processor Systems", CSRD Report No. 598, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., Sept. 3, 1986.

[AbHK86] Walid Abu-Sufah, Harlan Husmann and David Kuck. *On Input/Output Speed-up in Tightly-Coupled Multiprocessors.* **IEEE Trans. on Computers**, Vol. C–35, No. 6, pp. 520–530, June 1986.

[AbMa86] Walid Abu-Sufah and Allen D. Malony. "Experimental Results for Vector Processing on the Alliant FX/8", CSRD Rpt. No. 539, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., Feb. 11, 1986.

[AbMa86] Walid Abu-Sufah and Allen D. Malony. *Vector Processing on the Alliant FX/8 Multiprocessor.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 559–566, Aug. 19–22, 1986.

[AbPa86] Santosh Abraham and Janek Patel. "Parallel Garbage Collection on a Virtual Memory System", CSRD Report No. 620, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., Pittsburgh, PA, December 1986.

[Bane86] Utpal Banerjee. "Direct Parallelization of Call Statements – A Review", CSRD Rpt. No. 576, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., April 1986.

[BeSa86] Michael Berry and Ahmed Sameh. *Multiprocessor Jacobi Algorithms for Dense Symmetric Eigenvalue and Singular Value Decompositions.* **Proc. of the 1986 Int'l Conf. on Parallel Processing, St. Charles, IL**, pp. 433–440, Aug. 19–22, 1986.

[BGHJ86] M. Berry, K. Gallivan, W. Harrod, W. Jalby, S. Lo, U. Meier, B. Philippe and A. H. Sameh. *Parallel Algorithms on the CEDAR System.* In: **CONPAR 86, Lecture Notes in Computer Science**, W. Handler, ed. et. al.Springer–Verlag, pp. 25–39, 1986.

[CyrJ86] J. Cyr. "The Structured Memory Access Architecture: An Implementation and Performance Evaluation"", CSRD Rpt. No. 597, UILU-ENG-86-8008, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1986.

[Davi86] Timothy Davis. "PSolve: A Concurrent Algorithm for Solving Sparse Systems of Linear Equations", CSRD Report No. 612, Univ. of Illinois at Urbana–Champaign, Ctr. for Supercomputing Res. & Dev., December 1986.

[DKLS86] E. Davidson, D. Kuck, D. Lawrie and A. Sameh. *Supercomputing Tradeoffs and the Cedar System.* In: **High–Speed Computing: Scientific Applications and Algorithm Design**, R. Wilhelmson, ed. University of Illinois Press, pp. 3–11, 1986.

[Gann86] Dennis Gannon. "Restructuring Nested Loops on the Alliant Cedar Cluster: A Case Study of Gaussian Elimination of Banded Matrices", CSRD Rpt. No. 543, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., February 1986.

[GoPS86] G. H. Golub, R. J. Plemmons and A. Sameh. *Parallel Block Schemes for Large Scale Least Squares Computations.* In: **High–Speed Computing: Scientific Applications and Algorithm Design**, R. Wilhelmson, ed. University of Illinois Press, pp. 171–179, 1986.

[Guzz86] Mark Guzzi. "Multitasking Runtime Systems for the Cedar Multiprocessor", CSRD Rpt. No. 604, UILU-ENG-86-8009, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., July 1986.

[HaPa86] W. Ludwell Harrison III and David A. Padua. *Representing S–Expressions for the Efficient Evaluation of Lisp on Parallel Processors.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 703 710, Aug. 19–22, 1986.

[Harr86] W. Ludwell Harrison III. "Compiling Lisp for Evaluation on a Tightly

Coupled Multiprocessor", CSRD Rpt. No. 565, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., Mar. 20, 1986.

[Husm86] Harlan Husmann. "Compiler Memory Management and Compound Function Definition for Multiprocessors", CSRD Rpt. No. 575, UILU-ENG-86-8007, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1986.

[JaMe86] William Jalby and Ulrike Meier. *Optimizing Matrix Operations on a Parallel Multiprocessor with a Memory Hierarchy.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 429–432, Aug. 19–22, 1986.

[JaMS86] William Jalby, Ulrike Meier and Ahmed Sameh. "The Behaviour of Conjugate Gradient Based Algorithms on a Multi-Vector Processor with a Memory Hierarchy", CSRD Report No. 607, Univ. of Illinois at Urbana–Champaign, Ctr. for Supercomputing Res. & Dev., November 23, 1986.

[Kama86] C. Kamath. "Solution of Nonsymmetric Systems of Equations on a Multiprocessor", CSRD Rpt. No. 591, UILU-ENG-86-8004, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1986.

[KDLS86] David J. Kuck, Edward S. Davidson, Duncan H. Lawrie and Ahmed H. Sameh. *Parallel Supercomputing Today and the Cedar Approach.* **Science**, Vol. 231, pp. 967–974, Feb. 28, 1986.

[Kolc86] Alexander Kolchinsky. "Emulating Multitasking System Calls for a Multicluster Machine", CSRD Rpt. No. 560, UILU-ENG-86-8001, Univ. of Illinois at Urbana–Champaign, Dept. of Comput. Sci., May, 1986.

[LeeG86] G. Lee. "Some Issues in General-Purpose Shared Memory Multiprocessing: Parallelism Exploitation and Memory Access Combining", CSRD Rpt. No. 589, UILU-ENG-86-6002, Univ. of Illinois at Urbana–Champaign, Ctr. for Supercomputing Res. & Dev., June 1986.

[LeKK86] Gyungho Lee, Clyde P. Kruskal and David J. Kuck. *The Effectiveness of Combining in Shared Memory Parallel Computers in the Presence of 'Hot Spot'.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 35–41, Aug. 19–22, 1986.

[LoPh86] Sy-Shin Lo and Bernard Philippe. *The Symmetric Eigenvalue Problem on a Multiprocessor.* In: **Parallel Algorithms & Architectures**, North

Holland, Amsterdam, The Netherlands, pp. 31–43, 1986.

[Malo86] Allen Malony. "Cedar Performance Measurements", CSRD Rpt. No. 579, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., June 25, 1986.

[Malo86] A. Malony. "Cedar Performance Evaluation Tools: A Status Report", CSRD Rpt. No. 582, Univ. of Illinois at Urbana–Champaign, Ctr. for Supercomputing Res. & Dev., July 21, 1986.

[Midk86] S. Midkiff. "Automatic Generation of Synchronization Instructions for Parallel Processors", CSRD Rpt. No. 588, UILU–ENG–86–8002, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., May 1986.

[MiPa86] Samuel P. Midkiff and David A. Padua. *Compiler Generated Synchronization for DO Loops.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 544–551, Aug. 19–22, 1986.

[Nand86] N. R. Nandakumar. "Polynomial Preconditioning of Symmetric Indefinite Systems", CSRD Rpt. No. 580, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., June 1986.

[PaWo86] D. Padua and M. Wolfe. *Advanced Compiler Optimization for Supercomputers.* **CACM**, Vol. 29, No. 12, pp. 1184–1201, December, 1986.

[PoBa86] Constantine D. Polychronopoulos and Utpal Banerjee. *Speedup Bounds and Processor Allocation for Parallel Programs on Multiprocessors.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 961–968, Aug. 19–22, 1986.

[PoKP86] Constantine D. Polychronopoulos, David J. Kuck and David A. Padua. *Execution of Parallel Loops on Parallel Processor Systems.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 519–527, Aug. 19–22, 1986.

[Poly86] Constantine Polychronopoulos. "On Program Restructuring, Scheduling, and Communication for Parallel Processor Systems", CSRD Rpt. No. 595, UILU–ENG–86–8006, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., August 1986.

[PSKD86] Andrew R. Pleszkun, Gurindar S. Sohi, Bassam Z. Kahhaleh and Edward S. Davidson. *Features of the Structured Memory Access (SMA) Architecture.* **Proc. of Spring COMPCON 86**, pp. 259–263, Mar. 3–6, 1986.

[Swar86] Paul Swarztrauber. "Multiprocessor FFTs", CSRD Rpt. No. 608, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., June 1986.

[TaYe86] Peiyi Tang and Pen-Chung Yew. *Processor Self-Scheduling for Multiple-Nested Parallel Loops.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 528–535, Aug. 19–22, 1986.

[Trio86] Remi Triolet. "Interprocedural Analysis for Program Restructuring with Parafrase", CSRD Rpt. No. 538, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., Aug., 1986.

[Tzen86] N. F. Tzeng. "Fault Tolerant Multiprocessor Interconnection Networks and Their Fault-Diagnoses", CSRD Rpt. No. 594, UILU-ENG-86-8005, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., August 1986.

[TzYZ86] Nian-Feng Tzeng, Pen-Chung Yew and Chuan-Qi Zhu. *Fault-Diagnosis in a Multiple-Path Interconnection Network.* **Proc. of The 16th Int'l. Symp. on Fault-Tolerant Computing**, pp. 98–103, July 1–3, 1986.

[Veid86] Alex Veidenbaum. *A Compiler-Assisted Cache Coherence Solution for Multiprocessors.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 1029–1036, Aug. 19–22, 1986.

[YeTL86] Pen-Chung Yew, Nian-Feng Tzeng and Duncan H. Lawrie. *Distributing Hot Spot Addressing in Large Scale Multiprocessors.* **Proc. of the 1986 Int'l. Conf. on Parallel Processing, St. Charles, IL**, pp. 51–58, Aug. 19–22, 1986.

[AbKw85] Walid Abu-Sufah and Alex Y. Kwok. *Performance Prediction Tools for Cedar: A Multiprocessor Supercomputer.* **Proc. of the 12th Int'l. Symp. on Computer Architecture**, pp. 406–413, June 17–19, 1985.

[AbuS85] Walid Abu-Sufah. "Enhancing the Performance of Small Grain Multiprocessors", To appear in Rpt. on NBS Workshop on Performance Evaluation of High-Speed Computers, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., Gaithersburg, MD, June 5–6, 1985.

[Emra85] Perry Emrath. *Xylem: An Operating System for the Cedar Multiprocessor.* **IEEE Software**, Vol. 2, No. 4, pp. 30–37, July, 1985.

[Jack85] Daniel Thomas Jackson. "Data Movement in Doall Loops", CSRD Rpt. No. 524, UILU-ENG-85-8014, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., Aug., 1985.

[KaSa85] E. Kamgnia and A. Sameh. *A Numerical Conformal Mapping Method for Simply Connected Domains.* **Second SIAM Conf. on Parallel Processing for Scientific Computing**, Sept. 13, 1985.

[Kuck85] David J. Kuck. *Supercomputers and Distributed Computing.* **Proc. of the 1985 ACM Thirteenth Annual Computer Science Conf.** (CSC '85), pp. 34–46, Mar. 12–14, 1985.

[LeKK85] Ghungho Lee, Clyde P. Kruskal and David J. Kuck. *The Effectiveness of Automatic Restructuring on Nonnumerical Programs.* **Proc. of the 1985 Int'l. Conf. on Parallel Processing**, pp. 607–613, August 20–23, 1985.

[LeKK85] Gyungho Lee, Clyde P. Kruskal and David J. Kuck. *An Empirical Study of Automatic Restructuring of Nonnumerical Programs for Parallel Processors.* **Special Issue on Parallel Processing of IEEE Trans. on Computers**, Vol. C–34, No. 10, pp. 927–933, Oct., 1985.

[LiAb85] Zhiyuan Li and Walid Abu–Sufah. *A Technique for Reducing Synchronization Overhead in Large Scale Multiprocessors.* **Proc. of the 12th Int'l. Symp. on Computer Architecture**, pp. 284–291, June 17–19, 1985.

[LiZh85] Zhiyuan Li. "A Technique for Reducing Data Synchronization in Multiprocessed Loops", CSRD Rpt. No. 521, UILU–ENG–85–8013, Univ. of Illinois at Urbana–Champaign, Dept. of Computer Sci., May, 1985.

[LuLa85] Stephen F. Lundstrom and Duncan H. Lawrie. *Experiences with Distributed Systems.* **Special Issue of IEEE Software**, Vol. 2, No. 3, pp. 5–6, May, 1985.

[McDa85] Timothy Alan McDaniel. "Non–Linear Recurrences and EISPACK", CSRD Rpt. No. 511, UILU–ENG–85–8011, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., Oct., 1985.

[PaLa85] Krishnan Padmanabhan and Duncan H. Lawrie. *Performance Analysis of Redundant–Path Networks for Multiprocessor Systems.* **ACM Trans. on Computer Systs.**, Vol. 3, No. 2, pp. 117–144, May, 1985.

[Pawl85] Laura Marie Pawlowski. "Conversion of Parafrase Output to Alliant's Concurrent Fortran", CSRD Rpt. No. 532, UILU–ENG–85–8009, Univ. of Illinois at Urbana–Champaign, Center for Supercomputing Res. & Dev., Dec., 1985.

[Phil85] Bernard Philippe. "Approximating the Square Root of the Inverse of a Matrix", CSRD Rpt. No. 508, UILU-ENG-85-8009, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., July 1985.

[Same85] A. Sameh. *Parallel Linear System Solvers.* **Proc. of the Conf. on Vector and Parallel Processors for Scientific Computation**, May 27-29, 1985.

[Same85] Ahmed H. Sameh. *Algorithms and Experiments for Parallel Linear Systems Solvers.* **Second SIAM Conf. on Parallel Processing for Scientific Computing**, Nov. 18-21, 1985.

[TzYZ85] Nian-Feng Tzeng, Pen-Chung Yew and Chuan-Qi Zhu. *A Fault Tolerant Scheme for Multistage Interconnection Networks.* **Proc. of the 12th Int'l. Symp. on Computer Architecture**, pp. 368-375, June 17-19, 1985.

[TzYZ85] Nian-Feng Tzeng, Pen-Chung Yew and Chuan-Qi Zhu. *The Performance of a Fault-Tolerant Multistage Interconnection Network.* **Proc. of the 1985 Int'l. Conf. on Parallel Processing**, pp. 458-465, Aug. 20-23, 1985.

[Veid85] Alexander Veidenbaum. "Compiler Optimizations and Architecture Design Issues for Multiprocessors", CSRD Rpt. No. 85-520, UILU-ENG-85-8012, Univ. of Illinois at Urbana-Champaign, Dept. of Comput. Sci., May, 1985.

[WoLD85] Michael Wolfe, Bruce Leasure and James Davies. "Understanding Parafrase Output Listings", CSRD Rpt. No. 505, PR-85-6, UILU-ENG-85-8006, Univ. of Illinois at Urbana-Champaign, Center for Supercomputing Res. & Dev., July 9, 1985.

[AbuS84] Walid Abu-Sufah. *Cedar Performance Evaluation Tools.* **Proc. of Int'l. Workshop on Modeling and Performance Evaluation of Parallel Systems**, pp. 472-472, Dec. 13-18, 1984.