

U

DS

USE

U

NIX

USENIX

## SYMPOSIUM PROCEEDINGS

Symposium on Experiences with  
Distributed and Multiprocessor Systems  
(SEDMS II)

March 21 - 22, 1991

Atlanta, Georgia





# SEDMS II

## Symposium on Experiences with Distributed and Multiprocessor Systems

*Sponsored by*

The USENIX Association

and

The Software Engineering Research Center

江苏工业学院图书馆  
藏书章

*In cooperation with:*

ACM Special Interest Group on Data Communication (SIGCOMM)


ACM Special Interest Group on Operating Systems (SIGOPS)

IEEE-CS Technical Committee on Distributed Processing (TCDP)

IEEE-CS Technical Committee on Operating Systems (TCOS)

IEEE-CS Technical Committee on Software Engineering (TCSE)

March 21-22, 1991  
Atlanta, GA



For additional copies of these proceedings write

USENIX Association  
2560 Ninth Street, Suite 215  
Berkeley, CA 94710 USA

The price is \$30 for members and \$36 for non-members.

Outside the U.S.A and Canada, please add  
\$20 per copy for postage (via air printed matter).

Past USENIX Distributed & Multiprocessor Systems Proceedings

Distributed & Multiprocessor Systems Workshop 1989 Florida \$30

Outside the U.S.A. and Canada, please add  
\$20 per copy for postage (via air printed matter).

Copyright © 1991 by The USENIX Association  
All rights reserved.

This volume is published as a collective work.  
Rights to individual papers remain  
with the author or the author's employer.

UNIX is a registered trademark of UNIX System Laboratories, Inc.  
Other trademarks are noted in the text.

## Introduction

In 1988, George Leach and members of the Florida West Coast Unix community got the idea of having a workshop or conference on distributed systems in the area. George contacted Peter Salus (then Executive Director of Usenix) and Gene Spafford about the idea, and plans were made. The decision was to have an event that would stress experiences with distributed and multiprocessor systems, rather than the theory and algorithms that seem to predominate at so many other conferences in the area. A formal proposal was made to the Usenix board, and they approved the workshop. Gene got the directors of the SERC (Software Engineering Research Center) to cover some of the expenses and clerical efforts, and to provide other assistance in publicity.

Thus, in the autumn of 1989, Usenix and the SERC cosponsored the first Experiences with Distributed and Multiprocessor Systems event. That was intended to be a workshop, but because of the quantity and quality of submissions and participation, it became a miniconference. It was known by the acronym WEBDMS, and included 25 presented papers, some of which were later developed into a special issue of the journal *Computing Systems*. The workshop was attended by over 125 people in Ft. Lauderdale, Florida, and was judged a great success.

We decided to see if the theme for this event could support a second event, this time as a more formal symposium requiring somewhat more developed papers. Thus, with Usenix and SERC as cosponsors again, and again with cooperation of the ACM and the IEEE Computer Society, we issued calls for submissions. The call was not widely publicized in academic journals, but we covered some large mailing lists and Usenet newsgroups.

The response was both gratifying and surprising. With only three months lead and limited publicity, we received 60 papers from researchers on 5 continents. The submissions dealt with everything from cache design to optical computing to multimedia workstations to performance tuning and debugging of multiprocessor systems. The program committee reviewed the papers, made some tough choices, and the 21 papers in this proceedings are the result.

We have already started to plan a third event, another symposium. It has already been approved by Usenix. It will be held in the spring of 1992, probably in the Western United States somewhere. Again, we expect Usenix and SERC cosponsorship, and ACM and IEEE-CS participation. We also hope you will consider contributing to that event, and in future SEDMS that others may put together in the years to come.



In the meantime, our thanks to the hardworking members of our program committee, and to all the reviewers who aided them in their reading and decision-making. We greatly appreciate the hard work, advice, and efforts on our behalf by Ellie Young, Carolyn Carr, and Judy DesHarnais of Usenix. Georgia Conarroe of Purdue proved herself invaluable (again) in helping keep Spaf on track with the program committee tasks. And thanks especially to all the people who took the time and effort to contribute papers to the symposium, and to come to Atlanta in March to be with us. Thank you — we hope you enjoy it!

George Leach  
General Chair

Gene Spafford  
Program Chair

## The Program Committee

John R. Barr, Ph.D.  
Software Systems Research Laboratory  
Motorola, Inc.

Professor Bharat Bhargava  
Department of Computer Sciences  
Purdue University

Professor David L. Cohn  
Computer Science and Engineering  
University of Notre Dame

George W. Leach  
AT&T Paradyne

Michael D. O'Dell  
Bellcore

Professor Karsten Schwan  
College of Computing  
Georgia Institute of Technology

Professor Michael L. Scott  
Computer Science Department  
University of Rochester

Roger K. Shultz  
Collins Commercial Avionics  
Rockwell International

Professor Eugene H. Spafford, Chair  
Software Engineering Research Center  
Department of Computer Sciences  
Purdue University

Professor Satish K. Tripathi  
Department of Computer Science  
University of Maryland at College Park

## Other Reviewers (thanks!)

Ricardo G. Bianchini, William J. Bolosky, Michael R. Casey, Troy Cauble, Steve Chapin, Catherine F. Chronaki, Dennis J. Ciplickas, Gib Copeland, Prakash Ch. Das, William E. Garrett, Frank Greco, Paul M. Greenawalt, James Griffioen, Yenjo Han, Abdelsalam Helal, Karl F. Heubaum, Leonidas I. Kontothanassis, Edward W. Krauser, Dinesh C. Kulkarni, Evangelos Markatos, John O. Moody, Hsin Pan, Cesar A. Quiroz, James R. Robbins, Vincent F. Russo, John E. Saldanha, Lih-Chyun Shu, Neil G. Smithline, Matthew P. Stevenson, Colleen L. Templin, John M. Tracey, Karen M. Tracey, Jack E. Veenstra, Robert W. Wisniewski



# Program and Table of Contents

**Wednesday, March 20**

*Registration and Reception*

**7:00-9:00pm**

**Thursday, March 21**

*Opening Remarks*

**8:30**

George Leach, General Chair (ATT Paradyne, Largo, FL)  
Eugene Spafford, Program Chair (Software Engineering  
Research Center/Department of Computer Sciences, Purdue  
University)

*Keynote Presentation*

**8:45**

How to Move Parallel Processing into the Mainstream?  
Professor H. T. Kung (School of Computer Science,  
Carnegie-Mellon University)

*Break*

**9:45**

*Session I System-Building & Experience*

**10:15**

Experience Developing the RP3 Operating System..... 1  
Ray Bryant, Hung-Yang Chang and Bryan Rosenberg (IBM  
Research Division, Thomas J. Watson Research Center)

Experiences with Distributed Data Management in Real-time  
C3 Systems ..... 19  
Paul J. Fortier (Naval Underwater Systems Center), David  
V. Pitts, John C. Sieg, Jr. and C. Thomas Wilkes (Department  
of Computer Science, University of Lowell)

The *ION* Data Engine ..... 35  
Marc F. Pucci (Bellcore)

Building a Semi-Loosely Coupled Multiprocessor System  
Based on Network Process Extension ..... 51  
Helen S. Raizen (Prime Computer Inc.) and Stephen C.  
Schwarm (Digital Equipment Corporation)

*Lunch* 12:15 pm

*Session II RPC and Communications* 1:30 pm

Implementation and Performance of a Communication Facility for the RAID Distributed Transaction Processing System .... 69  
Enrique Mafla and Bharat Bhargava (Department of Computer Sciences, Purdue University)

Experience with Threads and RPC in Mach ..... 87  
Dan Duchamp (Computer Science Department, Columbia University)

Kernel-Kernel Communication in a Shared-Memory Multiprocessor..... 105  
Eliseu M. Chaves, Jr. (Universidade Federal do Rio de Janeiro, Brazil), Thomas J. LeBlanc, Brian D. Marsh and Michael L. Scott (Computer Science Department, University of Rochester)

*Break* 3:00

*Session III Scheduling / Sincronization* 3:45

Process Scheduling and Synchronization in the Renaissance Object-Oriented Multiprocessor Operating System..... 117  
Vincent F. Russo (Department of Computer Sciences, Purdue University)

A Hybrid Approach to Load Balancing in Distributed Systems ..... 133  
Prabha Gopinath (Philips Laboratories), and Rajiv Gupta (Department of Computer Science, University of Pittsburgh)

FALCON: A Distributed Scheduler for MIMD Architectures .... 149  
Andrew S. Grimshaw (Department of Computer Science, University of Virginia) and Virgilio E. Vivas (Department of Informatics, LAGOVEN, S.A., Venezuela)

*Short break* 5:15

*Work-in-Progress Session* 5:30  
Moderator: Mike O'Dell (Bellcore)



<i>Reception &amp; light buffet</i>	6:30-8:30	
<i>Optional discussion/panel session</i>	8:30	
Orthogonal Optimization Languages in Distributed Systems plus other discussion, debates, etc.		
<b>Friday, March 22</b>		
<i>Planning for SEDMS III</i> Leach/Spafford	8:30am	
<i>Session IV Debugging and Analysis</i>	8:45	
Performance Evaluation of the Sylvan Multiprocessor Architecture .....		165
Forbes J. Burkowski, Charles L. A. Clarke, Gordon J. Vreugdenhil (Department of Computer Science, University of Waterloo) and Crispin Cowan (Computer Sciences Department, University of Western Ontario)		
Debugging Multiprocessor Operating System Kernels .....		185
Noemi Paciorek, Susan LoVerso and Alan Langerman (Encore Computer Corporation)		
Debugging the Time Warp Operating System and Its Application Programs .....		203
Peter L. Reiher (Jet Propulsion Laboratory), Steven Bellenot (The Florida State University), and David Jefferson (UCLA)		
<i>Break</i>	10:15	
<i>Session V Performance Tuning</i>	11:00	
Lock Granularity Tuning Mechanisms in SVR4/MP .....		221
Mark D. Campbell, Russ Holt and John Slice (NCR Corporation, E&M Columbia)		
Measured Performance of Caching in the Sprite Network File System .....		229
Brent Welch (Xerox-PARC CSL)		
<i>Lunch</i>	12:00	
<i>Session VI Distributing Memory and Data</i>	1:30pm	

Using Kernel-Level Support for Distributed Shared Data ..... 247  
 David L. Cohn, Paul M. Greenawalt, Michael R. Casey and  
 Matthew P. Stevenson (Department of Computer Science and  
 Engineering, University of Notre Dame)

Virtual Memory Xinu ..... 261  
 Douglas Comer and James Griffioen (Department of Computer  
 Sciences, Purdue University)

Early Experience with Building and Using the Gothic Distri-  
 buted Operating System ..... 271  
 Isabelle Puaut, Michel Banatre and Jean-Paul Routeau  
 (IRISA—INRIA, France)

*Break* ..... 3:00

*Session VII Distributed Systems* ..... 3:30

Supporting an Object-Oriented Distributed System: Experience  
 with UNIX, Mach and Chorus ..... 283  
 F. Boyer, J. Cayuela, P. Y. Chevalier, A. Freyssinet, and  
 Daniel Hagimont (Unité Mixte Bull-IMAG/Systèmes, Gieres,  
 France)

Can We Study Design Issues of Distributed Operating Systems  
 in a Generalized Way? ..... 301  
 G. W. Gerrity, A. Goscinski, J. Indulska, W. Toomey and W.  
 Zhu (Department of Computer Science, University College,  
 University of New South Wales)

Language and Operating System Support for Distributed Pro-  
 gramming in Clouds ..... 321  
 Partha Dasgupta (Department of Computer Science and  
 Engineering, Arizona State University), R. Ananthanarayanan,  
 Sathis Menon, Ajay Mohindra, Mark Pearson, Raymond Chen  
 and Christopher Wilkenloh (Distributed Systems Laboratory,  
 College of Computing, Georgia Institute of Technology)



**Adjourn**

**5:00**

## **Alternate Paper**

***Not scheduled for presentation:***

**Experiences with the Liason Network Multimedia Workstation**

**Howard P. Katseff, Robert D. Gaglianella, Thomas B. London, Bethany S. Robinson and Donald B. Swicker (AT&T Bell Laboratories)**

# Experience Developing the RP3 Operating System<sup>1</sup>

Ray Bryant  
Hung-Yang Chang  
Bryan Rosenberg

IBM Research Division  
Thomas J. Watson Research Center  
P. O. Box 704  
Yorktown Heights, NY 10598  
{raybry,hychang,rosnbrg} @ ibm.com

## Abstract

RP3, the Research Parallel Processing Prototype, is a research vehicle for exploring the hardware and software aspects of highly parallel computation. RP3 is a shared-memory machine that was designed to be scalable to 512 processors; a 64 processor machine has been in operation since October 1988. The operating system for RP3 is a version of the Mach system from Carnegie Mellon University. This paper discusses what we have learned about developing operating systems for shared-memory parallel machines such as RP3 and includes recommendations on how we feel such systems should and should not be structured. We also evaluate the architectural features of RP3 from the viewpoint of our use of the machine. Finally, we include some recommendations for others who endeavor to build similar prototype or product machines.

## Introduction

The RP3 project of the IBM Research Division had as its goal the development of a research vehicle for exploring all aspects of highly parallel computation. RP3 is a shared-memory machine designed to be scalable to 512 processors; a 64-way machine was built and has been in operation since October of 1988.

For the past few years, the authors of this paper have been responsible for creating the operating system environment used to run programs on RP3. (The operating system for RP3 is a version of Mach [1] which is a restructured version of BSD 4.3 Unix.<sup>2</sup>) The extensions we made to Mach to support RP3 are described in [6] and will not be discussed in detail here. Instead, this paper summarizes our experience developing Mach/RP3 and presents our views on how operating systems for highly-parallel shared-memory machines such as RP3 should be constructed, as well as our experience in supporting and using this system for parallel processing research.

In the following sections of this paper, we provide an overview of the RP3 architecture and a brief history of the RP3 project. We then discuss the lessons we feel we learned during the course of this project and we state some recommendations we would make to developers of similar machines, whether such machines are designed as research prototypes or commercial products.

## RP3 Hardware Overview

Figure 1 illustrates the RP3 architecture. An RP3 machine can consist of up to 512 *processor memory elements* or PME's. The prototype hardware that was actually built, which we will refer to as RP3x, consists of 64 PME's. Each PME includes the following components:

- <sup>1</sup> Supported in part by the Defense Advanced Research Projects Agency under Contract Number N00039-87-C-0122 (Multi-processor System Architecture).
- <sup>2</sup> Unix is a registered trademark of AT&T in the United States and other countries.



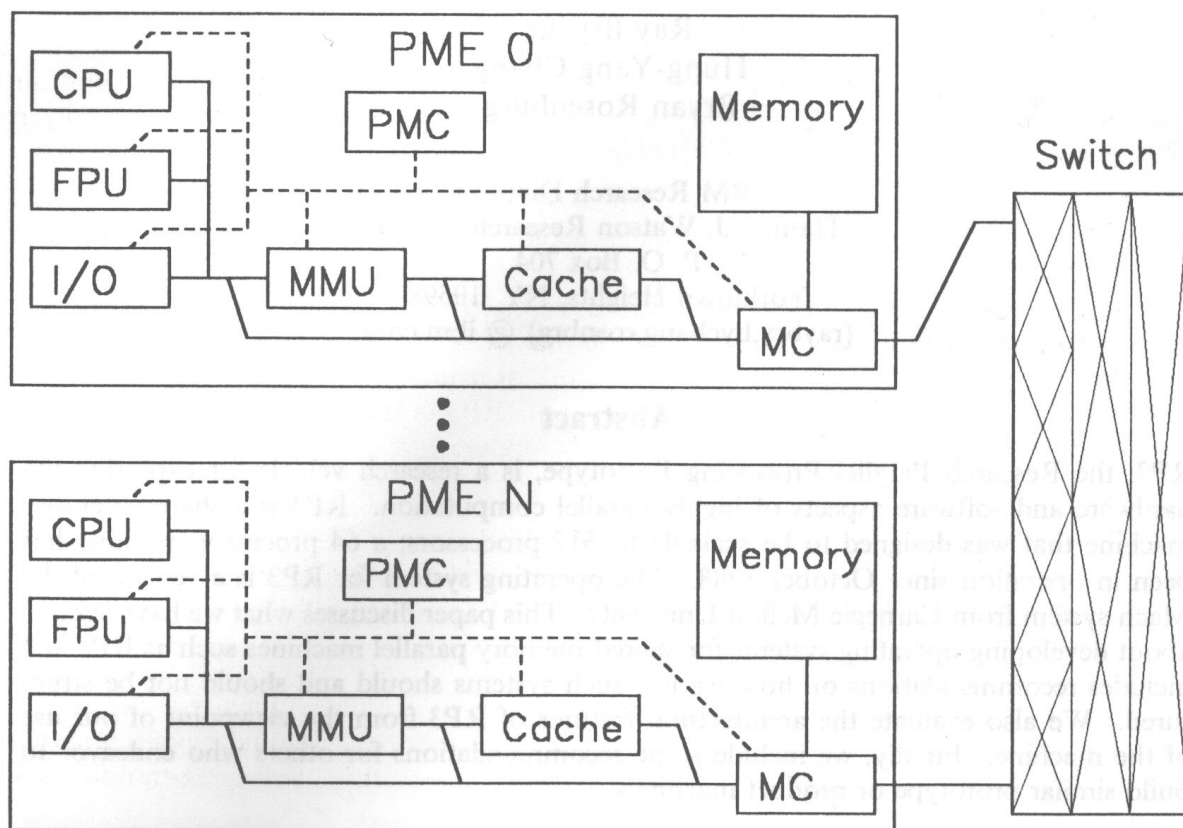


Figure 1. The RP3 Architecture

- CPU** The central processing unit, a 32-bit RISC processor known as the ROMP. The same processor is used in the IBM RT workstation.
- FPU** The floating point unit, similar to the floating point unit found on the IBM RT System APC card. It uses the Motorola MC68881 floating point chip which implements the IEEE floating point standard.
- I/O** The I/O interface, which provides a connection to an IBM PC/AT that serves as an I/O and Support Processor, or ISP. Each ISP is connected to 8 PME's and to an IBM System/370 mainframe.
- MMU** The memory management unit. The MMU supports a typical segment and page table address translation mechanism and includes a 64-entry, two-way set associative translation lookaside buffer (TLB).
- Cache** A 32-kilobyte, two-way set associative, real-address cache. To allow cache lookup to proceed simultaneously with virtual address translation, the RP3 page size is made equal to the cache set size of 16 kilobytes.
- MC** The memory controller. The memory controller examines each memory request to determine whether it is for this PME (in which case it is passed to the memory module) or a remote PME (in which case it is passed to the switch). The top 9 bits of the address specify the target PME.

**Memory** A 1- to 8-megabyte memory module. (The 64-way RP3x is fully populated with 8-megabyte memory modules.) Note that all memory in RP3 is packaged with the processors.

**PMC** The performance measurement chip. This device includes registers that count such things as instruction completions, cache hits and misses, local and remote memory references, and TLB misses. It can also periodically sample the switch response time.

All the PME's of an RP3 machine are connected by a multistage interconnection network or switch. The switch, which is constructed of water-cooled bipolar technology, has 64-bit data paths and a bandwidth of roughly 14 megabytes/second per PME. All memory on RP3 is local to individual PME's but is accessible from any processor in the machine. However, a performance penalty is incurred when accessing remote memory. RP3x has an access time ratio of 1:12:20 between cache, local, and remote memory, assuming no network or memory contention. The fact that not all memory in the system has the same access times puts RP3 in the class of *non-uniform memory access* or *NUMA* machines. Support of this NUMA architecture required operating system extensions that are discussed in [6].

To avoid potential memory bottlenecks, the RP3 architecture includes the concept of *interleaved* memory. Addresses for interleaved memory undergo an additional transformation after virtual to real address translation. The interleaving transformation exchanges bits in the low- and high-order portions of the real address (see figure 2). Since the high-order bits of the address specify the PME number, the effect of the interleaving transformation is to spread interleaved pages across memory modules in the system, with adjacent double-words being stored in different memory modules. The number of bits interchanged (and hence the log base 2 of the number of modules used) is specified by the interleave amount in the page table. Figure 2 shows how the interleaving transformation can be used to spread virtual pages across multiple PME's.

Normally, all data used by more than one processor is stored in interleaved memory. For this reason, interleaved memory is also referred to as *global* memory. Local, or non-interleaved, memory is referred to as *sequential* memory.

If enabled in the hardware, a one-to-one hashing transformation is applied before the interleaving transformation. The hashing transformation randomizes sequential memory references as an additional technique to minimize memory conflicts.

The RP3 hardware does not provide any mechanism for keeping caches coherent between PME's; cache coherency must be maintained in software. The cache is visible to application code in the sense that instructions to invalidate the cache are provided in user mode. In addition, the segment and page tables include cacheability information, so that ranges of virtual addresses (on page boundaries) can be specified as cacheable or non-cacheable memory. Since there is no page table associated with real-mode memory access, all real-mode memory accesses on RP3 are non-cacheable references. Cacheable memory can be further identified as *marked data*. A single cache operation can be used to invalidate all data in the cache that has been loaded from virtual memory identified as marked data.

RP3 supports the *fetch-and-add* [9] operation (as well as *fetch-and-or*, *fetch-and-and*, etc.) as the basic synchronization primitive. *Fetch-and-add(location,value)* is an atomic operation that returns the contents of 'location' and then increments the contents of the location by 'value'.

Further details of the design of the RP3 PME and system organization can be found in [15] and [5]. RP3x, our working 64-way prototype, differs from the published design in the following respects:

- The I/O and Support Processors, or ISP's, in RP3x are simple IBM PC/AT's rather than the elaborate custom-built machines described in the published RP3 design. Each PC/AT is connected to 8 PME's and can access RP3 memory through its PME's. Memory requests from an ISP are identical to requests from a PME's own processor, so an ISP can address real or virtual memory that is local, remote, or even interleaved. The ISP-PME bandwidth is

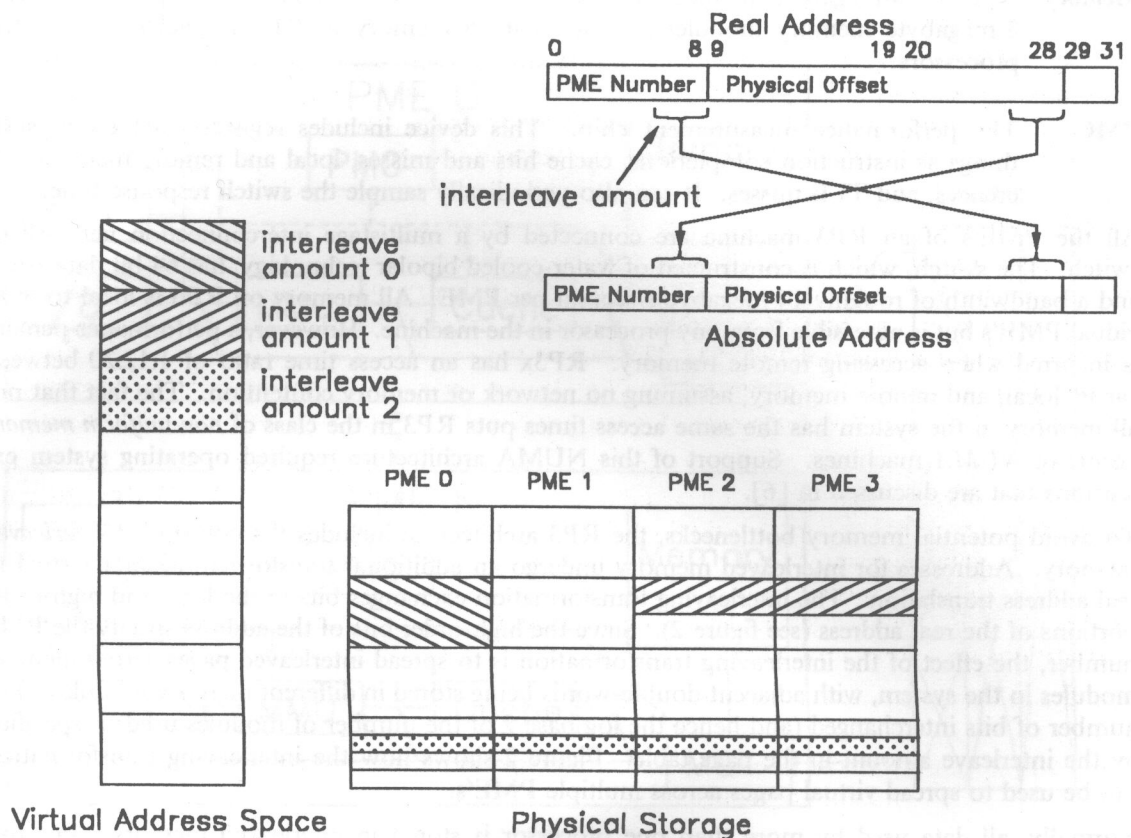


Figure 2. The RP3 interleaving transformation

roughly 500 kilobytes/second and can be dedicated to a single PME or multiplexed among PME's. An ISP can raise an interrupt in any of its PME's, and a PME can signal its ISP. The I/O hardware allows such a signal to interrupt the ISP, but our current ISP software is synchronous and periodically polls each PME's signal line.

In the original RP3 design, each ISP was to be directly connected to devices such as disks and networks. In the implemented design, the ISP's are channel-connected to a System/370 mainframe which in turn can access a large collection of disks and other devices. Bandwidth between an ISP and the System/370 is roughly 3 megabytes/second. RP3 I/O requests are passed from a PME to its ISP to the System/370 and back.

- The original RP3 design called for a *combining switch* that would reduce memory and switch contention by merging fetch-and-op operations when they collide at interior switch elements. The design could not be implemented in the technology available at the time. The current design supports the full range of fetch-and-op's, but the operations are serialized at individual memory controllers and are not combined in the switch.
- The floating point processors in RP3x are based on the standard RT workstation floating point unit and incorporate Motorola MC68881 floating point chips implementing the IEEE floating point standard. The original RP3 design called for vector floating point processors implementing the System/370 floating point specification.
- The RP3x cache system limits the PME clock rate to 7 MHz, about a third of the originally projected rate. Furthermore, the current memory controller is able to support just one out-



standing request to the memory subsystem at a time rather than the eight outstanding requests it was designed to handle.

- The RP3x memory management unit does not support hardware reload of the translation lookaside buffer (TLB). When a processor makes a memory request to a virtual address that is not mapped by the TLB, an exception is raised, and a software exception handler must explicitly load translation information for the faulting address into the TLB.

### **History of RP3**

Our experience with RP3 is closely related to its development history, so it is useful to summarize the major milestones of this project. The original idea that the IBM Research Division should attempt to build a large parallel processor apparently originated with a task force led by George Almasi during the winter of 1982-83. The earliest the name *RP3* was actually used appears to be in the fall of 1983, when a group led by Greg Pfister began to design the RP3 architecture. In October of 1984, the IBM *Corporate Management Committee* agreed to fund the RP3 project.

With funding secured, the existing project team was expanded to include a design automation group, a processor design group, a technology group (responsible for constructing the machine and coordinating production of parts with the IBM development divisions), and a software development group. The software group initially concentrated on the development of parallel applications. Since no parallel hardware was available, the approach selected was to emulate a virtual parallel processor under VM/370. This led to the development of the EPEX [7] system and a significant library of applications were written using the EPEX parallel programming extensions to Fortran.

Other significant technical milestones:

<b>Dec 1984</b>	RP3 architecture frozen. With the exceptions noted previously, this is a description of the machine as it exists today.
<b>Aug 1985</b>	A set of papers on RP3 were published in the <i>Proceedings of the 1985 International Conference on Parallel Processing</i> . [15][5][16]
<b>Dec 1985</b>	Power/mechanical frame completed and installed in lab.
<b>Jun 1986</b>	Uniprocessor version of RP3 instruction level simulator completed.
<b>Aug 1986</b>	First version of Mach on RP3 simulator completed.
<b>Dec 1986</b>	First complete processor chip set assembled and tested.
<b>Apr 1987</b>	Final-pass chip designs released to manufacturing.
<b>Sep 1987</b>	EPEX environment ported to Mach/RT.
<b>Sep 1987</b>	First full PME with final-pass chips completed.
<b>Sep 1987</b>	Multiprocessor version of RP3 instruction level simulator completed.
<b>Oct 1987</b>	Mach/RP3 runs on first PME.
<b>Nov 1987</b>	Mach/RP3 runs under multiprocessor RP3 simulator.
<b>Feb 1988</b>	Mach/RP3 runs on two-processor hardware.
<b>Jun 1988</b>	Mach and three EPEX test applications run on 4-processor hardware.

<b>Aug 1988</b>	Mach and three EPEX test applications run on 8-processor hardware.
<b>Oct 1988</b>	64-processor prototype (RP3x) completed and turned over to software team.
<b>Nov 1988</b>	64-way application speedup experiments completed on three EPEX test programs.
<b>Feb 1989</b>	Mach/RP3 with cacheability control and interleave support completed.
<b>Mar 1989</b>	Mach/RP3 with processor allocation primitives and local memory support completed.
<b>Jun 1989</b>	RP3x upgraded to include cache and PMC.
<b>Jul 1989</b>	RP3x available to outside users via NSF net.
<b>Mar 1990</b>	RP3x upgraded with floating point coprocessors. (Before this, all floating point had been emulated in software.)

A final historical note: all the authors of this paper joined the RP3 project after the initial design for the machine was complete. Thus, we are unable to comment on some of the early RP3 architectural and design decisions.

### *Lessons Learned from RP3 Operating Systems Development*

**Mach was a win.** The original plans for RP3 included a contract with the Ultracomputer project at New York University for the development of a Unix-compatible RP3 operating system based on the Ultracomputer Symunix operating system [8]. For a variety of reasons, our group chose to pursue Mach, first as an alternative, and then as the primary operating system for RP3. The selection of Mach as the basis for the RP3 operating system was a successful strategy for the following reasons:

- It allowed us to use the same operating system on RT workstations, on VM/370, and on RP3. These versions of Mach cooperate in supporting compilation, debugging, and testing of user code on RP3. The same programming environments and compilers execute under Mach/RT as under Mach/RP3, and users are able to accomplish much of their debugging on Mach/RT before moving to the parallel machine.
- It enabled the rapid development of an initial uniprocessor RP3 operating system. Mach was designed to be portable and maintains a fairly clear separation of machine-independent from machine-dependent code. Since RP3 uses the same processor as the RT workstation, porting Mach/RT to a single RP3 PME was straightforward. Uniprocessor Mach/RP3 uses not only the machine-independent code from Mach/RT but much of the machine-dependent code as well. The major exception concerns memory management, because the RP3 and RT memory management units are radically different. Here again the porting effort was aided by Mach's clear encapsulation of machine-dependent memory management code.
- It aided the transformation of the uniprocessor RP3 operating system into a multiprocessor operating system. The machine-independent Mach code was multiprocessor-capable to begin with. We could concentrate on making the RT-based machine-dependent code multiprocessor-capable as well. In this effort we were aided by the examples provided by existing Mach implementations for a variety of commercial multiprocessors.
- It simplified the support of the RP3 memory architecture. Changes for global and local memory support as well as for user-level cacheability control were isolated in the machine-dependent portion of the kernel.

Some disadvantages of using Mach were that