

**Intel Corporation**  
5200 N.E. Elam Young Parkway  
Hillsboro, OR 97124-6497

(503) 696-8080



May 1995

**Dear Paragon™ System Customer:**

The Paragon™ XP/S supercomputer can accommodate thousands of advanced microprocessors connected in a two-dimensional rectangular mesh. The Paragon XP/E supercomputer is a distributed-memory multicomputer that can accommodate up to thirty advanced compute nodes connected in a two-dimensional rectangular mesh.

The interconnect network offers high-bandwidth, low-latency communication and frees programmers from having to concern themselves with interconnect topology. The operating system brings the Open Software Foundation's industry standard version of the UNIX operating system to the performance-driven environment of scalable, distributed-memory computing.

Thank you for joining the fast-growing community of scientists and programmers now taking advantage of Paragon systems to tackle problems not before possible at an affordable price.

This package contains the system software for the Paragon system. Please read through the documentation and distribute the materials to those intending to use the system.

**Before using your system:**

- **Read this letter completely.**
- **Verify the contents of this package.**
- **Read the *Paragon™ System Software Release 1.3 Release Notes*.**



## Package Contents

The operating system software is provided on three 0.25-inch 12000 BPI cartridge tapes in UNIX tar format. These tapes are listed in Table 1.

**Table 1. Installation Media**

<b>Description</b>	<b>Order Number</b>
Paragon™ System Software Release 1.3	634485-001 ✓
Paragon™ SAT Software Release 1.3	634486-001 ✓
Paragon™ ParAide Software Release 1.3 Native, Sun4/SunOS-4 and Silicon Graphics Hosted Development Environment Tools	634487-001 ✓

Diagnostic and compiler software are packaged separately. If you are a brand new customer, receiving Paragon system software for the very first time, Table 2 lists the included documentation.

**Table 2. Documentation (1 of 3)**

<b>Description</b>	<b>Order Number</b>
<i>Paragon™ System Software Release 1.3 Release Notes</i>	313167-001
<i>Paragon™ System Software Installation Guide</i>	313168-001
<i>OSF/1 Documentation Errata</i>	312857-003
<i>Paragon™ System Technical Documentation Guide</i>	312820-003
<i>User Group Membership Pack</i>	638535-001
<i>Paragon™ System Acceptance Test User's Guide</i>	312648-004
<i>Paragon™ System Administrator's Guide</i>	312544-004
<i>Paragon™ System Application Tools User's Guide</i>	312545-004
<i>Paragon™ XP/S System Cabling Guide (only Paragon XP/S system customers)</i>	312823-001
<i>Paragon™ System Commands Reference Manual</i>	312486-004
<i>Paragon™ System C Compiler Release 5.0 Release Notes</i>	633948-001
<i>Paragon™ System C Compiler User's Guide</i>	312490-003
<i>Paragon™ System C System Calls Reference Manual</i>	312487-004
<i>Paragon™ System Fiber Distributed Data Interface Installation and Configuration Guide</i>	312814-003



**Table 2. Documentation (2 of 3)**

<b>Description</b>	<b>Order Number</b>
<i>Paragon™ System Fortran Compiler Release 5.0 Release Notes (only Paragon XP/S MP system customers)</i>	633953-001
<i>Paragon™ System Fortran Compiler User's Guide (only Paragon XP/S MP system customers)</i>	312491-003
<i>Paragon™ System Fortran Language Reference Manual (only Paragon XP/S MP system customers)</i>	312644-002
<i>Paragon™ System Fortran System Calls Reference Manual (only Paragon XP/S MP system customers)</i>	312488-003
<i>Paragon™ System Graphics Libraries User's Guide</i>	312887-001
<i>Paragon™ XP/S System Hardware Installation Manual (only Paragon XP/S system customers)</i>	312543-004
<i>Paragon™ XP/E System Hardware Installation Manual (only Paragon XP/E system customers)</i>	312843-003
<i>Paragon™ XP/S System Hardware Maintenance Manual</i>	312822-002
<i>Paragon™ System High-Performance Parallel Interface Manual</i>	312824-003
<i>Paragon™ System Interactive Parallel Debugger Reference Manual</i>	312547-004
<i>Paragon™ XP/S i860™ 64-Bit Microprocessor Assembler Reference Manual</i>	312546-001
<i>Paragon™ System Multi-User Accounting and Control System Manual</i>	312891-003
<i>Paragon™ System Network Queueing System Manual</i>	312645-004
<i>Paragon™ System Performance Visualization Tool User's Guide</i>	312889-003
<i>i860™ Microprocessor Family Programmer's Reference Manual</i>	240875-002
<i>Paragon™ XP/S System RAID Utilities Manual</i>	312646-003
<i>Paragon™ XP/S System Site Preparation Guide (only Paragon XP/S system customers)</i>	312485-004
<i>Paragon™ XP/E System Site Preparation Guide (only Paragon XP/E system customers)</i>	312842-003
<i>Paragon™ System User's Guide</i>	312489-004



**Table 2. Documentation (3 of 3)**

<b>Description</b>	<b>Order Number</b>
<i>C: A Reference Manual</i>	313152-001
<i>The C Programming Language</i>	122008-002
<i>Effective Fortran 77 (only Paragon XP/S MP system customers)</i>	312201-001
<i>X Protocol Reference Manual</i>	312654-001
<i>Xlib Programming Manual</i>	312655-001
<i>Xlib Reference Manual</i>	312656-001
<i>X Toolkit Intrinsic Reference Manual</i>	312658-001
<i>X Toolkit Intrinsic Programming Manual</i>	312657-001
<i>Open Desktop User's Guide</i>	312954-001
<i>Open Desktop Administrator's Guide</i>	312955-001
<i>OSF/1 Network Application Programmer's Guide</i>	PSCOSF1M <sup>1</sup>
<i>OSF/1 Command Reference</i>	PSCOSF1M <sup>1</sup>
<i>OSF/1 Programmer's Reference</i>	PSCOSF1M <sup>1</sup>
<i>OSF/1 System and Network Administrator's Reference</i>	PSCOSF1M <sup>1</sup>
<i>OSF/1 User's Guide</i>	PSCOSF1M <sup>1</sup>

<sup>1</sup>This product code designates all five volumes of the OSF/1 documentation set.

If you are an existing customer, receiving an update, Table 3 lists the included documentation. This list includes all manuals and release notes that have changed since your last shipment.

**Table 3. Updated Documentation (1 of 2)**

<b>Description</b>	<b>Order Number</b>
<i>Paragon™ System Software Release 1.3 Release Notes</i>	313167-001 ✓
<i>Paragon™ System Software Installation Guide</i>	313168-001 ✓
<i>OSF/1 Documentation Errata</i>	312857-003 ✓
<i>Paragon™ System Technical Documentation Guide</i>	312820-003 ✓
<i>User Group Membership Pack</i> ✓	312805-001 ✓
<i>Paragon™ System Acceptance Test User's Guide</i>	312648-004 ✓





**Table 3. Updated Documentation (2 of 2)**

Description	Order Number
<i>Paragon™ System Administrator's Guide</i>	312544-004 ✓
<i>Paragon™ System Application Tools User's Guide</i>	312545-004 ✓
<i>Paragon™ System Commands Reference Manual</i>	312486-004 ✓
<i>Paragon™ System C Compiler Release 5.0 Release Notes</i>	633948-001 ✓
<i>Paragon™ System C Compiler User's Guide</i>	312490-003 ✓
<i>Paragon™ System C System Calls Reference Manual</i>	312487-004 ✓
<i>Paragon™ System Fiber Distributed Data Interface Installation and Configuration Guide</i>	312814-003
<i>Paragon™ System Fortran Compiler Release 5.0 Release Notes (only Paragon XP/S MP system customers)</i>	633953-001 ✓
<i>Paragon™ System Fortran Compiler User's Guide (only Paragon XP/S MP system customers)</i>	312491-003 ✓
<i>Paragon™ System Fortran Language Reference Manual (only Paragon XP/S MP system customers)</i>	312644-002 ✓
<i>Paragon™ System Fortran System Calls Reference Manual (only Paragon XP/S MP system customers)</i>	312488-003 004 ✓
<i>Paragon™ XP/S System Hardware Installation Manual</i>	312543-004 ✓
<i>Paragon™ XP/E System Hardware Installation Manual (only Paragon XP/E system customers)</i>	312843-003
<i>Paragon™ XP/S System Hardware Maintenance Manual</i>	312822-002 ✓
<i>Paragon™ System High-Performance Parallel Interface Manual</i>	312824-003 ✓
<i>Paragon™ System Interactive Parallel Debugger Reference Manual</i>	312547-004 ✓
<i>Paragon™ System Multi-User Accounting and Control System Manual</i>	312891-003 ✓
<i>Paragon™ System Network Queueing System Manual</i>	312645-004 ✓
<i>Paragon™ System Performance Visualization Tool User's Guide</i>	312889-003 ✓
<i>Paragon™ XP/S System Site Preparation Guide</i>	312485-004 ✓
<i>Paragon™ XP/E System Site Preparation Guide (only Paragon XP/E system customers)</i>	312842-003
<i>Paragon™ System User's Guide</i>	312489-004 ✓



Additional materials are listed in the letters describing the diagnostics, compiler, and other software that accompany Release 1.3.

If items are missing, or if you have any questions, please contact Intel Scalable Systems Division immediately. Please refer to the section "Comments and Assistance" in this letter for information about how to contact Intel Supercomputer Systems Division.

### **What is in Release 1.3?**

The Paragon System Release 1.3 software is the latest release of the operating system. It includes the X Window System, online manual pages, and manuals in both hardcopy and PostScript formats. For a list of features in this release, refer to the *Paragon™ System Software Release 1.3 Release Notes*.

### **Installation**

For directions on how to install the Paragon System Release 1.3 software, refer to the *Paragon™ System Software Release 1.3 Release Notes*.

### **NOTE**

Adding or removing any boards or components from your Paragon system can damage the system and may invalidate your warranty. Please contact Intel Scalable Systems Division Customer Support for assistance in answering your questions.

### **Restrictions and Limitations of Release 1.3**

Every effort has been taken to ensure the quality of this release, but at shipment we are aware of some limitations. Please refer to the *Paragon™ System Software Release 1.3 Release Notes* for known limitations and available workarounds.

### **Comments and Assistance**

We are eager to hear of your experiences with the Paragon system. Please call us if you need assistance, have questions, or otherwise want to comment on your Paragon system.



**U.S.A./Canada Intel Corporation**  
**Phone: 800-421-2823**  
**Internet: support@ssd.intel.com**

---

**Intel Corporation Italia s.p.a.**  
Milanofiori Palazzo  
20090 Assago  
Milano  
Italy  
1678 77203 (toll free)

**France Intel Corporation**  
1 Rue Edison-BP303  
78054 St. Quentin-en-Yvelines Cedex  
France  
0590 8602 (toll free)

**Intel Japan K.K.**  
**Scalable Systems Division**  
5-6 Tokodai, Tsukuba City  
Ibaraki-Ken 300-26  
Japan  
0298-47-8904

**United Kingdom Intel Corporation (UK) Ltd.**  
**Scalable Systems Division**  
Pipers Way  
Swindon SN3 IRJ  
England  
0800 212665 (toll free)  
(44) 793 491056  
(44) 793 431062  
(44) 793 480874  
(44) 793 495108

**Germany Intel Semiconductor GmbH**  
Dornacher Strasse 1  
85622 Feldkirchen bei Muenchen  
Germany  
0130 813741 (toll free)

---

**World Headquarters**  
**Intel Corporation**  
**Scalable Systems Division**  
15201 N.W. Greenbrier Parkway  
Beaverton, Oregon 97006  
U.S.A.  
(503) 677-7600 (Monday through Friday, 8 AM to 5 PM Pacific Time)  
Fax: (503) 677-9147

If you have comments about our manuals, please fill out and mail the enclosed Comment Card. You can also send your comments electronically to the following address:

**techpubs@ssd.intel.com** (Internet)




## Intel Supercomputer Users' Group

The Intel Supercomputer Users Group promotes the exchange of information among users. Intel strongly supports the Users Group and encourages participation in its activities, which include: Special Interest Groups (SIGs), an annual international users conference, an electronic mail task force, and a "freeware" library of user-contributed software, available electronically to all members of the Intel Supercomputer Users' Group. For membership information contact:

**JoAnne Wold (503-677-5322)**  
**joanne@ssd.intel.com (Internet)**

Again, thank you for acquiring a Paragon Supercomputer.

Sincerely,



Peter Wolochow

Product Marketing Manager  
Intel Scalable Systems Division

---

Paragon is a trademark of Intel Corporation.  
UNIX is a trademark of UNIX System Laboratories.  
SCO and OPEN DESKTOP are trademarks of The Santa Cruz Operation, Inc.  
Silicon Graphics is a registered trademark of Silicon Graphics, Inc.  
Sun Microsystems is a trademark of Sun Microsystems.  
The X Window System is a trademark of the Massachusetts Institute of Technology.

Copyright © 1995 Intel Corporation





May 1995

Order Number: 313167-001

---

**Paragon<sup>™</sup> System Software**  
**Release 1.3**  
**Release Notes**

---

**Intel<sup>®</sup> Corporation**

Copyright ©1995 by Intel Scalable Systems Division, Beaverton, Oregon. All rights reserved. No part of this work may be reproduced or copied in any form or by any means...graphic, electronic, or mechanical including photocopying, taping, or information storage and retrieval systems...without the express written consent of Intel Corporation. The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update or to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication, or disclosure is subject to restrictions stated in Intel's software license agreement. Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraphs (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 252.227-7013. Intel Corporation, 2200 Mission College Boulevard, Santa Clara, CA 95052-8119. For all Federal use or contracts other than DoD, Restricted Rights under FAR 52.227-14, ALT. III shall apply.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

286	i386	Intel	iPSC
287	i387	Intel386	Paragon
i	i486	Intel387	
	i487	Intel486	
	i860	Intel487	

APSO is a service mark of Verdex Corporation

DGL is a trademark of Silicon Graphics, Inc.

Ethernet is a registered trademark of XEROX Corporation

EXABYTE is a registered trademark of EXABYTE Corporation

Excelan is a trademark of Excelan Corporation

EXOS is a trademark or equipment designator of Excelan Corporation

FORGE is a trademark of Applied Parallel Research, Inc.

Green Hills Software, C-386, and FORTRAN-386 are trademarks of Green Hills Software, Inc.

GVAS is a trademark of Verdex Corporation

IBM and IBM/VS are registered trademarks of International Business Machines

Lucid and Lucid Common Lisp are trademarks of Lucid, Inc.

NFS is a trademark of Sun Microsystems

OpenGL is a trademark of Silicon Graphics, Inc.

OSF, OSF/1, OSF/Motif, and Motif are trademarks of Open Software Foundation, Inc.

PGI and PGF77 are trademarks of The Portland Group, Inc.

PostScript is a trademark of Adobe Systems Incorporated

ParaSoft is a trademark of ParaSoft Corporation

SCO and OPEN DESKTOP are registered trademarks of The Santa Cruz Operation, Inc.

Seagate, Seagate Technology, and the Seagate logo are registered trademarks of Seagate Technology, Inc.

SGI and SiliconGraphics are registered trademarks of Silicon Graphics, Inc.

Sun Microsystems and the combination of Sun and a numeric suffix are trademarks of Sun Microsystems

The X Window System is a trademark of Massachusetts Institute of Technology

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd.

VADS and Verdex are registered trademarks of Verdex Corporation

VAST2 is a registered trademark of Pacific-Sierra Research Corporation

VMS and VAX are trademarks of Digital Equipment Corporation

VP/ix is a trademark of INTERACTIVE Systems Corporation and Phoenix Technologies, Ltd.

XENIX is a trademark of Microsoft Corporation

## **WARNING**

Some of the circuitry inside this system operates at hazardous energy and electric shock voltage levels. To avoid the risk of personal injury due to contact with an energy hazard, or risk of electric shock, do not enter any portion of this system unless it is intended to be accessible without the use of a tool. The areas that are considered accessible are the outer enclosure and the area just inside the front door when all of the front panels are installed, and the front of the diagnostic station. There are no user serviceable areas inside the system. Refer any need for such access only to technical personnel that have been qualified by Intel Corporation.

## **CAUTION**

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

## **LIMITED RIGHTS**

The information contained in this document is copyrighted by and shall remain the property of Intel Corporation. Use, duplication or disclosure by the U.S. Government is subject to Limited Rights as set forth in subparagraphs (a)(15) of the Rights in Technical Data and Computer Software clause at 252.227-7013. Intel Corporation, 2200 Mission College Boulevard, Santa Clara, CA 95052. For all Federal use or contracts other than DoD Limited Rights under FAR 52.2272-14, ALT. III shall apply. Unpublished—rights reserved under the copyright laws of the United States.



# Preface

---

These release notes contain guidelines and limitations to keep in mind when using the operating system software. The release notes also contain a list of open and fixed bugs.

## Organization

- |           |  |
|-----------|--|
| Chapter 1 | Describes guidelines and limitations that you should know before using the operating system software.  |
| Chapter 2 | Contains an open bug list and a fixed bug list. The open bug list lists open bugs against the operating system software. The fixed bug list lists bugs fixed since the last release. |

## Notational Conventions

This manual uses the following notational conventions:

- |                                     |  |
|-------------------------------------|--|
| <b>Bold</b>                         | Identifies command names and switches, system call names, reserved words, and other items that must be used exactly as shown.  |
| <i>Italic</i>                       | Identifies variables, filenames, directories, processes, user names, and writer annotations in examples. Italic type style is also occasionally used to emphasize a word or phrase.  |
| Plain-Monospace                     | Identifies computer output (prompts and messages), examples, and values of variables. Some examples contain annotations that describe specific parts of the example. These annotations (which are not part of the example code or session) appear in <i>italic</i> type style and flush with the right margin. |
| <b><i>Bold-Italic-Monospace</i></b> | Identifies user input (what you enter in response to some prompt).   |
-

**Bold-Monospace**

Identifies the names of keyboard keys (which are also enclosed in angle brackets). A dash indicates that the key preceding the dash is to be held down *while* the key following the dash is pressed. For example:

`<Break>`      `<s>`      `<Ctrl-Alt-Del>`

- [ ]      (Brackets) Surround optional items.
- ...      (Ellipses) Indicate that the preceding item may be repeated.
- |      (Bar) Separates two or more items of which you may select only one.
- { }      (Braces) Surround two or more items of which you must select one.

## Applicable Documentation

For information about the manuals shipped with the Paragon system, see the *Paragon™ System Technical Documentation Guide*.

## Comments and Assistance

If you are part of the Release 1.3 Beta program, contact the Parallel Systems Engineer or Field Applications Engineer associated with your site.

Intel Scalable Systems Division is eager to hear of your experiences with our products. Please call us if you need assistance, have questions, or otherwise want to comment on your Paragon system.

**U.S.A./Canada Intel Corporation**  
**Phone: 800-421-2823**  
**Internet: [support@ssd.intel.com](mailto:support@ssd.intel.com)**

---

**Intel Corporation Italia s.p.a.**

Milano Fiori Palazzo  
 20090 Assago  
 Milano  
 Italy  
 1678 77203 (toll free)

**France Intel Corporation**

1 Rue Edison-BP303  
 78054 St. Quentin-en-Yvelines Cedex  
 France  
 0590 8602 (toll free)

**Intel Japan K.K.**

**Scalable Systems Division**

5-6 Tokodai, Tsukuba City  
 Ibaraki-Ken 300-26  
 Japan  
 0298-47-8904

**United Kingdom Intel Corporation (UK) Ltd.**

**Scalable Systems Division**

Pipers Way  
 Swindon SN3 IRJ  
 England  
 0800 212665 (toll free)  
 (44) 793 491056 (*answered in French*)  
 (44) 793 431062 (*answered in Italian*)  
 (44) 793 480874 (*answered in German*)  
 (44) 793 495108 (*answered in English*)

**Germany Intel Semiconductor GmbH**

Dornacher Strasse 1  
 85622 Feldkirchen bei Muenchen  
 Germany  
 0130 813741 (toll free)

---

**World Headquarters**

**Intel Corporation**

**Scalable Systems Division**

15201 N.W. Greenbrier Parkway  
 Beaverton, Oregon 97006  
 U.S.A.

(503) 677-7600 (Monday through Friday, 8 AM to 5 PM Pacific Time)

Fax: (503) 677-9147

If you have comments about our manuals, please fill out and mail the enclosed Comment Card. You can also send your comments electronically to the following address:

**[techpubs@ssd.intel.com](mailto:techpubs@ssd.intel.com)**  
 (Internet)





# Table of Contents

---

## Chapter 1 Features, Guidelines, and Limitations

<b>Introduction</b> .....	1-1
<b>Operating System Memory Usage</b> .....	1-1
<b>New Features</b> .....	1-2
Interactive Parallel Debugger (IPD) .....	1-2
ParAide .....	1-2
SPV .....	1-3
XIPD .....	1-3
<b>Guidelines and Limitations</b> .....	1-4
Booting the Paragon™ System .....	1-4
Recompile and Relink Application Code .....	1-5
Gang Scheduling .....	1-5
General Programming Guidelines .....	1-5
Do Not Use Mach System Calls .....	1-6
Avoid Using Large Statically-Allocated Data Structures .....	1-6
SMP Programming .....	1-6
Namelist Groups .....	1-6
-Msave and -Mreentrant .....	1-7
Pthread-Specific Data .....	1-7
setjmp() and longjmp() Functions .....	1-7

---

Global Operations .....	1-7
SIGUSR2 Signal .....	1-7
Message Passing .....	1-7
Global Sends .....	1-8
Synchronous Sends on Large Paragon™ Systems .....	1-8
Setting the Process Type of a Controlling Process .....	1-8
Message Handlers .....	1-8
Forced Message Types .....	1-9
Using the Allocator .....	1-9
Verifying MACS Accounts with the Allocator .....	1-9
I/O System .....	1-10
Maximum Compute Nodes Per I/O node .....	1-10
Logical Volume Manager .....	1-10
Verifying fsck Results .....	1-11
HIPPI Limitations .....	1-11
IPI-3 Limitations .....	1-11
The ParAide Toolset .....	1-12
Using IPD on Applications with NX Handler Routines .....	1-12
The msgqueue Command and Global Sends .....	1-13
Performance Monitoring .....	1-13
Scaling IPD .....	1-14
Graphical Tools .....	1-14
Paragraph and NX Handler-Invoking Routines .....	1-14
XIPD .....	1-14
SPV X Resources for Pre-X11R5 Servers .....	1-14
SPV and Memory Usage .....	1-15
SPV Not Started If Configuration Includes SUNMOS .....	1-15
System Administration .....	1-15
No Disk Quota Support .....	1-15
Shutting Down the Paragon™ System .....	1-15
Backing Up a File System to the DAT Tape Drive .....	1-16
Disabling Exception Debug Code .....	1-16

Shared Virtual Memory .....1-16  
NQS Enhancements .....1-17  
NQS For Workstations .....1-18  
NQS Limitation .....1-18  
MACS Database File .....1-18  
NFS For Workstations .....1-19

## **Chapter 2**

### **Bug Lists for System Software**

**Introduction** .....2-1  
**Open Bug List** .....2-2  
**Fixed Bug List** .....2-24

## List of Tables

Table 1-1. Operating System Memory Usage .....1-2

# Features, Guidelines, and Limitations

---

1

## Introduction

The chapter contains the following information:

- Memory usage information for the operating system.
- New features with this release.
- Guidelines for using the operating system software.

## Operating System Memory Usage

This section provides memory usage information for Release 1.3 of the operating system. Table 1-1 shows the amount of memory after a fresh boot. The numbers are scaled up to the next Megabyte. The memory shown includes that taken up by the microkernel and the compute server, including SPV (which takes about 0.5M byte). This number subtracted from the total amount of physical memory on the board represents the amount of memory available for your application before paging commences.

Note that the memory used by the operating system increases with the physical memory on the board. This is primarily because larger page tables are required. Please note that these numbers do not include memory taken up by default message passing buffers, which is about 1M byte. Please refer to the *application* manual page for information about how to lower message buffer requirements for applications that do minimal message passing.

Table 1-1. Operating System Memory Usage

Physical Memory on Compute Node Board (Megabytes)	Memory Used by the Operating System (Megabytes)
16	9
32	10
64	11
128	12

## CAUTION

Never shut down the diagnostic station before shutting down the Paragon system. Doing so would cause system errors and incorrectly halt the system.

## New Features

This section highlights features that are new to Release 1.3.

### Interactive Parallel Debugger (IPD)

IPD now allows you to analyze the core files that may be created when programs fault. Evidence in the core files may provide valuable insight into why the application faulted, and may help you create a more reliable system. A new **coreload** command has been added to allow you to load one or more core files. Any of the IPD commands that allow static examination can then be used to analyze the core files. For more information, refer to the *Paragon™ Interactive Parallel Debugger Reference Manual*, particularly the description of the **coreload** command.

### ParAide

The shell area of the ParAide main window did not previously provide support for interactive jobs and job control. ParAide now provides a terminal area that runs like an embedded xterm, providing support for interactive jobs and job control.

In addition, a status line has been added to the bottom of the ParAide main window. A status message is displayed when any of the following occur.

- A graphical tool is started
- A load command is sent to the terminal
- An editor window is brought up
- A command that does not bring up the Command Viewer is started from the Command menu.

Finally, the ParAide main window now provides a Command menu that can be customized by the user. The items in the Command menu are defined by an application resource, so you can modify the menu to contain any non-interactive commands you wish. The commands can execute directly or prompt the user. The output of the commands can be discarded or loaded into a dialog for viewing. For a complete description of the Command menu and how to customize it, refer to the ParAide online help for the Command menu.

## SPV

The Node display has been enhanced to show the new features of the MP node.

In addition, you can now select a set of nodes in a display by holding down the mouse select button, dragging the cursor to form a rectangle over the desired nodes, and releasing the select button.

## XIPD

XIPD has been enhanced to support the core file analysis features of IPD. A Core Analysis dialog has been added so you can load core directories and analyze core files. For a complete description of the Core Analysis dialog, refer to the XIPD online help for the Core Analysis dialog.

## NOTE

XIPD, unlike IPD, does not support debugging single, non-parallel host-node processes. Because of this, XIPD only supports the analysis of core directories, not single host-node core files. IPD should be used to analyze single host-node core files.

The following X resources have been added to XIPD:

### **XIpd.doCoreAnalysis**

A boolean value that controls whether the Core Analysis dialog comes up first instead of the Load Application dialog. The default is **False**.

**XIpd.coreDirectoryName**

A string that defines the default name for the core directory to be loaded into the Core Analysis dialog.

**XIpd.coreSelect**

A string that defines the default selection of files from the core directory. The string can be one of the following:

<b>Fault</b>	All files from faulting processes are selected. This is the default.
<b>Non-Fault</b>	All files from non-faulting processes are selected.
<b>All</b>	All files are selected.

The following command-line options have been added to XIPD:

**-core** Directs XIPD to go directly to the Core Analysis dialog instead of bringing up the Load Application dialog.

**-coreDir** *directory*

Directs XIPD to go directly to the Core Analysis dialog instead of bringing up the Load Application dialog, and to load the specified core directory.

The Instrumentation dialog has been simplified so that seldom-used areas reside in collapsible regions that can be expanded by setting a toggle button. For a description of the changes to the Instrumentation dialog, refer to the XIPD online help for the Instrumentation dialog.

## Guidelines and Limitations

This section contains guidelines for using the Paragon system. These guidelines are pertinent for both a system administrator and a programmer. The section also lists some of the most important limitations. A complete list of bugs in a reference format is in Chapter 2, “Bug Lists for System Software.”

## Booting the Paragon™ System

We strongly recommend booting your Paragon system with kernel assertions off.

Please contact SSD Customer Support for information about which operating system kernel is best for the applications running on your system. See the diagnostic station online **reset** reference page for information about using the **reset** command to boot a Paragon system. To boot with assertions, one must specify **reset debug**.



## Recompile and Relink Application Code

When a new release of the system software is installed on your system, recompile and relink your application code using the compilers and system software libraries provided with the new release. You need to do this because executables from different releases are not compatible.

## Gang Scheduling

Using gang scheduling will reduce the stability of a Paragon system at your site depending on the application mix you are running, the number of nodes that applications are running on, and how much paging occurs in an application. In general, small applications, which cause less paging, will be more stable than larger applications.

See the *Paragon™ System Administrator's Guide* for information on configuring partitions for gang scheduling.

Note that higher priority applications do not roll out lower priority applications. The allocator assigns partition layers to each gang-scheduled partition based on whether the application will fit in a layer; priority is not a consideration. If applications of different priorities are scheduled, you may find that not all higher priority applications will run at the same time, even if they can all fit in the same layer. For example, suppose you have a 16 node partition with the following applications to run:

- Application1 has priority of 10 and a size of 8 nodes.
- Application2 has priority of 5 and a size of 8 nodes.
- Application3 has priority of 10 and a size of 8 nodes.

Application1 and Application 2 are started first, then Application3 is started. Application3 is not allowed to run, because Application2 is already running and there is not enough space in the partition for Application3. Even though Application3 has a higher priority than Application2, it does not run before Application2.

## General Programming Guidelines

This software release does not require workarounds for most code that runs on less than 128 nodes. If you are running on a larger number of nodes or if you experience difficulty with your program, the following guidelines may be helpful.

1. Dynamically allocate memory at run time, especially if the application requires more than 10M bytes of memory. Use **malloc()** for C and **ALLOCATE** for Fortran. Typically, dynamic allocation gives a shorter startup execution time.

2. By keeping the runtime size of an application within the available memory as shown in Table 1-1 (under 22M bytes for 32M-byte nodes), you can prevent excessive paging in an application.
3. When setting up a paging tree, maintain a maximum ratio of 32 compute nodes per disk subsystem.
4. Never use an `nx_initve()` or `nx_initve_rect()` system call in a program running in the compute partition.

## Do Not Use Mach System Calls

Do not use the Mach system call interface. Only use calls described in SSD documentation. The Mach interface is not documented in SSD manuals, but you may read about it elsewhere. This interface is not supported. If you use Mach system calls, your application may fail. Mach memory allocation and Paragon memory allocation do not work together.

## Avoid Using Large Statically-Allocated Data Structures

Pages for statically-allocated data structures are touched at load time. This can substantially increase the load time for applications with large statically-allocated data structures. Use dynamically-allocated data structures when possible (for example, use `malloc()` for C or `ALLOCATE` for Fortran).

## SMP Programming

Symmetric Multiprocessing(SMP) programs are multi-pthreaded programs that run on one or more processors. A single image of the operating system runs on these processors. Also, none of the pthreads has any specialized access to the hardware.

## Namelist Groups

Namelist groups are used in Fortran programs. Local variables that appear in a namelist group are not placed on the stack, even if the module is compiled with `-Mreentrant`. They are statically allocated. If you compile with `-Mncall` (which allows loops with procedures to be parallelized), then you should check that the variables in the namelist group do not introduce unwanted thread dependences. Treat the namelist variables as if they are declared in a `SAVE` statement.

## **-Msave and -Mreentrant**

**-Msave** has the opposite effect of **-Mreentrant**. Programs compiled with **-Msave** will have their local variables static, not stack-based. When compiling procedures intended to be thread-safe, do not use **-Msave** with **-Mreentrant**.

## **Pthread-Specific Data**

Pthread-specific data are bound to a pthread key with **pthread\_setspecific()**. The key is created with **pthread\_keycreate()**. Note that all pthread-specific data for a process reside in one memory page, which on Paragon systems is 4K bytes. Hence, you may have **pthread\_keycreate()** fail with an ENOMEM when your system still has memory available.

## **setjmp() and longjmp() Functions**

If automatic and register variables are changed after the **setjmp()** (or **sigsetjmp()**), their values are indeterminate after the **longjmp()** (or **siglongjmp()**). You cannot assume that these variables are restored to what they were when the **setjmp()** (or **sigsetjmp()**) was called. Nor can you assume that they are set to whatever value they had when the function sequence containing the **longjmp()** (or **siglongjmp()**) was started.

If you want them to retain the values they had when the function sequence containing the **longjmp()** (or **siglongjmp()**) is started, declare them as **volatile**.

## **Global Operations**

The global operations (**g...()** calls) are not tuned for maximum performance in the Release 1.3 system software. Only minor performance improvements can be expected over the Release 1.2 system software. A side effect is that the global operations may scale in unexpected ways. For example, the **gdsun()** call may exhibit a substantial linear component when purely logarithmic scaling might be expected.

## **SIGUSR2 Signal**

The **SIGUSR2** signal value is not available to parallel applications because the system uses it to implement gang scheduling. This is true whether or not gang scheduling is used.

## **Message Passing**

This section provides release information for operating system message passing.

## Global Sends

Global sends use a -1 value for the *node* parameter in an operating system send system call. Global sends are now implemented using a tree-structured store-and-forward message passing strategy. This requires each node in an application to completely receive the message, otherwise, the next node in the tree will not receive the message. Use any of the operating system receive calls to do the receive.

## Synchronous Sends on Large Paragon™ Systems

An application that uses the **csend()** call for synchronous message passing can block or hang if you do not allocate enough memory for the application to handle messages. The memory allocated to a given node for message passing is based on the number of nodes in the application and the total memory available to the application for message passing. Therefore, as the number of nodes for an application increases, the memory available to each node for message passing decreases. With a large application running on a large Paragon system, this decrease can happen faster than expected causing the application to block. For example, an application that uses the **csend()** call for message passing may run fine on 64 nodes, but have problems running on 128 nodes, because there is not enough memory for message passing. The following workarounds can prevent this problem from happening:

- Post a receive before doing the synchronous send.
- Use a non-blocking send, for example, the **isend()** call.
- Decrease the message size for the application.
- Increase the memory buffer size for the application using the **-mbf** switch.
- Use the **-noc** switch to control the number of correspondents.

## Setting the Process Type of a Controlling Process

Do not call **setptype()** in a controlling process that does not do message passing, because this call assigns memory for message buffering that will be unused in this process. This memory is wired-in; that means it cannot be paged.

## Message Handlers

You must use **masktrap()** around any code in the main program that could interfere with calls in a handler established with one or more **h...()** calls.

Because it is often not obvious which calls could interfere with each other, use **masktrap()** to protect *all* library calls in the rest of the program that could call the same subsystems as the calls in the handler while the handler is active. For more information, see the discussion of **masktrap()** in the *Paragon™ System User's Guide*.

## Forced Message Types

If you use a force message type and don't post the receive in advance then the message is dropped. The message is also dropped if the receiving buffer is paged out. If the application is run with the **-plk** switch, the receiving buffer is guaranteed to be in memory. Forced messages have no performance advantage on Paragon systems.

Forced messages are useful for backward compatibility. Because earlier systems (iPSC/860 systems) did not provide paging, any program trying to be backward compatible should either use **-plk** or not use forced messages at all.

## Using the Allocator

With Release 1.3, you now specify how the allocator controls partitions and applications using the allocator configuration file */etc/nx/allocator.config*, instead of using command line switches. This functionality is described in the **allocator** and **allocator.config** manual pages in the *Paragon™ System Commands Reference Manual*. You can also find information on this in the *Paragon™ System Administrator's Guide*. You should be particularly aware of the following changes to the allocator.

## Verifying MACS Accounts with the Allocator

Verifying MACS accounts is now controlled with the **USE\_MACS** parameter in the allocator configuration file */etc/nx/allocator.config*, instead of using the allocator's **-MACS** switch that was provided in previous releases. The description of the **USE\_MACS** parameter is included in the **allocator** and **allocator.config** online manual pages, but is not included in these manual pages in the *Paragon™ System Commands Reference Manual*.

The following parameter can be specified in the file *allocator.config* to specify that the allocator verify a user's MACS account before running an application.

**USE\_MACS=boolean**

Specifies whether the allocator must validate users' accounts with the Paragon Multi-User Accounting and Control System (MACS). This allows MACS to verify that users belong to valid MACS accounts. The *boolean* value specifies the following:

**0**

The allocator does not validate users' account IDs with MACS.

- 1 The allocator must validate users' account IDs with MACS.

The factory default for the *USE\_MACS* parameter is 0 (this line is omitted from the *allocator.config* file); if this line is omitted or commented out or the *allocator.config* file is missing, the default value is 0.

This parameter is equivalent to the allocator switch **-MACS** used in previous releases. This parameter should be used instead of the **-MACS** switch.

## I/O System

This section provides release information for the I/O system.

### Maximum Compute Nodes Per I/O node

Applications requiring high performance should not attempt simultaneous I/O from greater than 32 compute nodes to any one I/O node (MIO) at one time (the compute node/I/O node ratio). This is true for UFS, NFS, and PFS (striped on one I/O node). The default configuration accepts compute node/I/O node ratios less than or equal to 32. Compute node/I/O node ratios greater than the configured amount can cause system hangs when multiple nodes are accessing that same I/O node.

To allow for larger compute node/I/O node ratios, set the bootmagic variable *NORMA\_RDMA\_GROUP\_XMIT\_ALLOC* to the desired compute node/I/O node ratio plus 16. For example, to allow for a compute node/I/O ratio of 48, set *NORMA\_RDMA\_GROUP\_XMIT\_ALLOC* to 64. The larger *NORMA\_RDMA\_GROUP\_XMIT\_ALLOC* is, the more kernel memory is used. This is memory that you cannot use for an application and that cannot be paged. Hence, it is desirable not to use more than you need.

The system administrator must be careful to configure the PFS file systems so that the compute node/I/O node ratio is enforced across multiple simultaneous applications.

For more information about configuring PFS file systems, see the *Paragon™ System Administrator's Guide*. Users must understand PFS I/O modes and stripe attributes in order to code their applications within this restriction. For more information about programming PFS applications, see the *Paragon™ System User's Guide*.

### Logical Volume Manager

Although documented in the OSF/1 manuals, the logical volume manager (LVM) is inappropriate for a Paragon system. The LVM software and online manual pages have been removed from the system. The LVM software is a set of resources that OSF/1 provides for managing disk storage in a system. The following LVM commands are not installed on the Paragon OSF/1 system:

<b>lvchange</b>	<b>lvremove</b>	<b>pvmove</b>	<b>vgreduce</b>
<b>lvcreate</b>	<b>lvsync</b>	<b>vgchange</b>	<b>vgremove</b>
<b>lvdisplay</b>	<b>pvchange</b>	<b>vgcreate</b>	<b>vgsync</b>
<b>lvextend</b>	<b>pvcreate</b>	<b>vgdisplay</b>	
<b>lvreduce</b>	<b>pvdisplay</b>	<b>vgextend</b>	

## Verifying fsck Results

When you boot the system, the **fsck** command is automatically run on each file system defined in */etc/fstab* (such as */home*, */pfs* and the PFS stripe directories typically mounted on */home/.sdirs/vol\**).

In the case of a severely damaged file system, the **fsck** will fail and a message will be displayed to the screen. The message will scroll off the screen before the reboot is complete and you may not notice it. The system will continue to boot to a multiuser state with no other indication that anything is wrong.

If the **fsck** step fails, the specified directory will not be mounted on the affected file system. Any files written to that directory will be written into the file system that contains the parent directory. Later, if you notice that the file system did not mount and you **fsck** and mount it, the files that had been written into the directory become hidden. To avoid this problem, after each boot you can manually compare the mounted file systems with the contents of the *fstab* file.

For example, if the PFS stripe directory */home/.sdirs/vol1* does not mount because the **fsck** step failed, then any data written to that stripe directory will actually be written to the file system that contains the */home* directory.

## HIPPI Limitations

Use the bootmagic string *TCP\_SPACE\_SIZE* with caution. A value larger than 64K (65536) can cause the system to panic. Use this bootmagic string at your own risk.

## IPI-3 Limitations

Under certain conditions, PFS striped across IPI-3 Unix partitions may cause the Paragon system to panic. For example, if PFS is striped across all the Unix partitions in an IPI-3 facility partition and if an attempt is made to create a PFS file greater than the available disk space with **lsize**, a panic occurs. The error message is as follows:

```
panic: getblk: size too big when IPI-3 PFS filesystem is full
```

The only response at this point is to reboot the Paragon system. If the system were operating properly, lsize would return with an error, allowing the user or the system administrator to delete the file and free up disk space. For more information about IPI-3 refer to the *Paragon™ System High-Performance Parallel Interface Manual*.

## The ParAide Toolset

The ParAide toolset does not support SMP programming. If you perform IPD **prof** or **gprof** instrumentation, performance data will be collected. However, the percentage of time spent in the procedures (first three columns in **prof** output data) will only reflect the initial or main thread, while the call counts (fourth column in **prof** output data) will reflect performance data for all threads.

## Using IPD on Applications with NX Handler Routines

If an application contains a call to any of the NX handler-invoking routines (**hsend()**, **hsendx()**, **hrecv()**, **hrecvx()**, **hsendrecv()**), a message is displayed. The message is only displayed once. It is seen the first time any process in the application executes any one of these calls.

The warning is displayed because handlers are implemented using threads, and IPD does not provide a way to change which thread is targeted by a command. A thread is a point of execution control, so it maintains its own register set and stack frame. All other data is shared among threads. Because of this, it is only important to understand which thread you are in when viewing the contents of registers or examining a stack trace.

There is no means of specifying which thread IPD should control at any given time. The thread that received a signal that stopped it is the one visible to IPD. So, if you are in a handler routine when a fault is generated (e.g. **SIGBUS**) or a breakpoint is hit, the stack trace and register values displayed by IPD belong to the handler thread.

Whenever you are stopped, you can determine which thread you are seeing by using the **frame** command. The stack trace of a process stopped in an **hrecv** handler will look similar to the following:

```
(all:0) > frame(1:0)
***** (1:0) *****
myhandler() [test_hrecv.c{ }#70]
_nx_port_recv_thread() [nx_port.c{ }0x00021c28]
```

Once you are stopped in the handler routine, there is no way to view the main thread without continuing the process (and vice versa). You have to place a breakpoint in the main thread, and if you encounter the breakpoint (or the handler thread completes and exits), you would then be able to view the stack frame and register data for that thread.



There is a side effect when debugging an application that has invoked a handler. If you encounter a breakpoint in the handler, the main thread gets stopped to wherever it happens to be when the break is encountered (and vice versa). Likewise, source stepping in the handler causes the main thread to start and stop also, but NOT one source line at a time. This is because even though the debugger is in control of only one thread at a time, the signals that control the starting and stopping of the application apply to all of the threads.

Setting watchpoints in handler-invoking applications can have unpredictable results. If the watchpoint is originally set when the main thread is visible to IPD (as determined by a **frame** command), data accesses performed by the handler thread will not cause execution to stop until the handler is halted for some other reason (e.g. a breakpoint) and then resumed. The opposite is true if a watchpoint is originally set when a handler thread is visible to IPD. In this case, data accesses that occur in the main thread will not stop execution until execution is stopped in the main thread for some other reason and then resumed.

## The msgqueue Command and Global Sends

The **msgqueue** command may appear to be incorrectly reporting messages that have been sent but not received if the program being debugged is using global sends. The implementation of a global send is such that only a subset of nodes in the partition are initially sent the message. This subset then sends the message on to another subset and so on (spanning tree is used). The problem is that the send is not forwarded to the next subset until a receive of that message has completed on the node expected to do the forwarding send.

If you stop execution immediately after executing a global send and execute a **msgqueue** command for all nodes, you would expect to see an unreceived message waiting on every node. But what you see instead is just a few nodes with messages waiting. These are the subset of nodes targeted by the initial send. These nodes must now receive the message they were sent and then they will send that same message on to the next subset. The **msgqueue** command does not show the sends to the second level of the tree, as they have not occurred yet.

## Performance Monitoring

IPD supports performance monitoring on 66 or fewer nodes. Performance monitoring data collection can be done on more nodes, but will occasionally cause the system to hang. To do performance monitoring, use the IPD **instrument** command with the **-prof**, **-gprof**, or **-paragraph** switches. For example:

```
(all:0)> instrument -prof
```

Performance monitoring of applications running on a large number of nodes can take a long time.

## Scaling IPD

IPD does not currently scale well above 512 nodes.

## Graphical Tools

The graphical tools (SPV, ParaGraph, ParAide, XIPD, XProf, and XGprof) use Motif 1.2 for their interface, and Motif 1.2 display commands can crash non-current versions of Sun's XNeWS window server. If you are not running the current version of Sun's XNeWS window server, you must upgrade the XNeWS server in order to run these tools.

Due to the size of their binary executables, running multiple X applications slows down the system.

To improve performance when using the graphical tools, change the bootmagic variable *TCP\_SPACE\_SIZE* on the Paragon system from the default size of 4K to 64K. 64K is a better size when using X applications that transmit data through an Ethernet connection.

## CAUTION

64K is the suggested upper limit for the *TCP\_SPACE\_SIZE* variable. Using a larger value can result in undesirable system side effects.

## Paragraph and NX Handler-Invoking Routines

Currently, there is no support for identifying ParaGraph trace event information at the thread level. Thus, if both a main thread and a handler thread are generating trace information, both events will be interleaved in the resulting trace file with no means of sorting during post-processing. ParaGraph will report an error whenever it finds a sequence of events that is not properly sorted.

## XIPD

XIPD should be run on 32M-byte service nodes. This is especially important when debugging applications that run on a large number of nodes, because the more nodes an application runs on the more memory XIPD uses.

## SPV X Resources for Pre-X11R5 Servers

Users may need to set the fonts in the resource file(s) to execute applications on a pre- X11R5 server on the workstation. For a description of the SPV X resources, refer to the *Paragon™ System Performance Visualization Tool User's Guide*.

## SPV and Memory Usage

SPV computes memory usage as the total number of pages minus the current number of free pages. This does not give a complete picture of memory usage, because the operating system has a concept of an “inactive” pool separate from the free pool. This allows memory to become active quickly without the overhead associated with moving memory from the free pool to the active pool.

The operating system tries to keep pages in the inactive pool, so when the number of free pages gets small, a lot of paging occurs to reclaim pages from the inactive pool. This tends to increase the usage value over time, but should not be confused with a memory leak, where the amount of wired memory increases over time. For example, if someone logs in and then immediately logs out, a number of pages would be left in the inactive pool and SPV would show an increase in the usage percentage.

## SPV Not Started If Configuration Includes SUNMOS

The `/sbin/init.d/spv` script has been modified to not start the `spvdaemon` if an alternate kernel is specified to execute on the compute nodes. The script prints the following error message if `BOOT_ALT_NODE_LIST` is not null:

```
spv data collection NOT started: no support for alternate OSs.
```

## System Administration

This section provides release information for Paragon system administration.

### No Disk Quota Support

The Paragon system does not provide disk quota support. This means that the commands **edquota**, **quotacheck**, **quotaon**, **quotaoff**, and **repquota** are not available. The command **quot** is available; note, however, that *file system* must designate the raw device. Also, the **-f** and **-n** options on **quot** do not work.

## Shutting Down the Paragon™ System

Use the following command sequence to reboot the system. In a console window:

```
# cd /  
# shutdown now  
# umount -A  
# sync;sync;sync  
# halt
```

This writes the system buffers, kills all running processes, brings the system to single-user mode, unmounts the file systems, and halts the system. You can safely power down the system after this sequence. To get back to the diagnostic station prompt, enter a `~q`.

To reboot the system, use the following command after the **halt** command completes.

```
DS# cd /usr/paragon/boot
DS# reset
```

For more information about system shutdown, see the *Paragon™ System Administrator's Guide*.

## Backing Up a File System to the DAT Tape Drive

The following example allows you to efficiently use the **dump** command to back up a file system to the DAT tape drive on the Paragon system:

```
/usr/sbin/dump -f /dev/io0/rmt6 -c /dev/io0/rrz0a
```

The **-f** switch specifies using the `/dev/io0/rmt6` storage device for the dump. The **-c** switch specifies that the dump medium is a cartridge tape. The command dumps the file system on `/dev/io0/rrz0a` to the DAT drive.

If you specify the file system by its mount point name (for example, `/home`) rather than by its raw disk name (for example, `/dev/io0/rrz0f`), the specified file system must be listed in `/etc/fstab`. If a specified mount point name is not in `/etc/fstab`, the dump is aborted. The resulting error message does not readily indicate what the problem is. The error message is as follows:

```
.ioctl DIOMRINFO: Not a typewriter
dump: The ENTIRE dump is aborted
```

## Disabling Exception Debug Code

When an exception occurs, the kernel displays debug information on the node console. By default, this information includes the contents of the kernel and user registers. On Paragon MP systems (BOOT\_CPU\_MODE=ama), this may cause a hang. To disable the display of this additional information and protect against the hang, set the bootmagic variable `EXCEPTION_DEBUG_HINT` to 0. To do this, add the line `"EXCEPTION_DEBUG_HINT=0"` to the file `MAGIC.MASTER` in `/usr/paragon/boot` on the diagnostic station and reboot the Paragon system.

## Shared Virtual Memory

The OSF/1 operating system supports a feature called *shared virtual memory* that can be used to share data between processes. Both the System V shared-memory calls and the mapped-memory interface are supported (see `shmget(2)` and `mmap(2)` in the *OSF/1 Programmer's Reference* for

information about these calls). However, these calls can only be used to share memory between processes running on the *same node*. Any attempt to share memory between processes running on different nodes will fail.

## NQS Enhancements

### **Fix for PTS #9491 (NQS partition names can conflict when submitted from workstations).**

To make it possible to distinguish between NQS jobs submitted with the same request ID from different remote workstations, NQS has been changed to create partitions named *NQS\_mid\_rid*, where *mid* is the machine ID of the submitting workstation and *rid* is the request ID of the NQS job.

### **Fix for PTS #6589 (NQS does not spawn your job's shell as a login shell).**

The NQS *sched\_param* file now has a new parameter called *use\_login*. If *use\_login* is set to 1, NQS spawns the user's shell as a login shell (with *argv[0]* set to *-sh*, *-csh*, or *-ksh*), which makes the shell read its *.profile* or *.cshrc.login* file as it starts up. If *use\_login* is set to 0 or omitted, the user's shell is spawned as a non-login shell (as in earlier releases).

### **Fix for PTS #9950 (NQS needs a "start only when notified" flag).**

The NQS *sched\_param* file now has a new parameter called *nosched*. If *nosched* is set to 1, NQS will not schedule any jobs at all. If *nosched* is set to 0 or omitted, NQS schedules jobs normally.

A new NQS command called **qstart** has also been added. This command submits a small do-nothing job, which forces NQS to re-read its *sched\_param* file immediately.

*nosched* and **qstart** can be used to prevent NQS from rescheduling a job that crashed the system. If an NQS job crashed the system and will start again when the system comes up, use the following procedure to prevent it from running:

1. Boot the system to single-user mode.
2. Edit the *sched\_param* file and change *nosched* to 1.
3. Continue booting to multiuser mode.
4. Remove the problem job from the queue.
5. Edit the *sched\_param* file and change *nosched* to 0.
6. Use the **qstart** command to make NQS begin scheduling immediately. If you don't do this, NQS begins scheduling within 15 minutes.

## NQS For Workstations

The Paragon NQS Network Queueing System (NQS) supports a networked environment. However, the NQS executable files shipped with the operating system software are *only* for Paragon systems. If you want to use NQS from your workstation, you have to separately obtain the NQS software for your workstation.

## NQS Limitation

The NQS **qmgr** subcommand **lock local\_daemon** incorrectly wires down all memory on the service node, causing the system to hang. This problem is designated as PTS #3825.

## MACS Database File

If the *macd.data* file is invalid, MACS tries *macd.data.new*, *macd.data.old*, and *macd.data.bak* before giving up. MACS also now ensures that an existing *macd.data* is valid before renaming it to *macd.data.old*.

These changes make it less likely for the *macd.data* file to become truncated or corrupted, and make MACS attempt to recover automatically if this file does become truncated or corrupted. However, these changes do not reduce the system administrator's responsibility to back up the *macd.data* file at frequent intervals.

An important side effect of these changes is that a *macd.data* file created by any earlier version of MACS will be considered invalid. If all of the following are true, MACS will fail to start up:

- You have installed R1.3 or later on a system where MACS was in use prior to R1.3.
- The previous version of */usr/spool/macs/private/macd.data* is present on the system.
- MACS is configured in macwatch mode.

The error message displayed on the console during boot looks like this:

```
No /usr/spool/macs/private/macd.data file found; cannot start MACD.  
/usr/spool/macs/private/macd.data is either pre-1.3 or corrupted:  
checksum mismatch, stored=66, computed=394 (use "dbconvert" to  
convert an older database)... Could not open  
/usr/spool/macs/private/macd.data.new for reading...  
/usr/spool/macs/private/macd.data.old is either pre-1.3 or  
corrupted: checksum mismatch, stored=66, computed=394 (use  
"dbconvert" to convert an older database)...  
/usr/spool/macs/private/macd.data.bak is either pre-1.3 or
```

```
corrupted: checksum mismatch, stored=66, computed=394 (use
"dbconvert" to convert an older database)... No more database files
to try!
MACPD failed to start up!
```

This message is also written to */var/adm/syslog/daemon.log*, and the following mail is sent to *root*:

```
No valid /usr/spool/macsd/private/macsd.data file found! MACS
stopped.
See /var/adm/syslog/daemon.log for more information.
Sincerely,
MACPD process
```

If MACS fails to start up as described above, you can use the new **dbconvert** command to convert the old MACS database (*/usr/spool/macsd/private/macsd.data*) to the new format. The syntax of this command is: **dbconvert oldfile newfile**

**dbconvert** reads the MACS database specified by *oldfile* and writes the same data in the new format to *newfile*. Here is an example of its use.

```
# cd /usr/spool/macsd/private
# mv macsd.data macsd.data.R1.2
# dbconvert macsd.data.R1.2 macsd.data
new database version=1.3
Database conversion completed successfully
```

Once you have converted the MACS database, you can then start MACS manually

```
# /sbin/init.d/macsd start
MACS services provided.
```

## NFS For Workstations

The diagnostic station OS installation instructions do not currently point out that SCO UNIX requires that a "key" be entered during software installation. This is an activation key. If you use the same source media (floppies/tape) to load the SCO UNIX on another system, and then try to use NFS, the system detects that another system is on the network with the same activation key, and then NFS is "disabled" and refuses to work.





# Bug Lists for System Software

2

## Introduction

This chapter contains a list of open bugs and a list of fixed bugs. These lists are updated just before shipment and are also available online in the files */usr/share/release\_notes/ss\_buglist* and */usr/share/release\_notes/ss\_fixed* on the Paragon system.

The open bug list lists the open bugs since Release 1.2 of the Paragon system software. The open bug list is organized in alphabetical order by subsystem name. The bug list includes the following:

- Bug number
- Subsystem name
- Bug synopsis
- Bug description

The fixed bug list lists the bugs fixed since Release 1.2 of the Paragon system software and included in Release 1.3. The fixed bug list is organized in numerical order by bug number. The bug listing includes the following:

- Bug number
- Subsystem name
- Bug synopsis

These bug lists were generated on 4/9/95.

## Open Bug List

The following lists the open bugs for Release 1.3 of the Paragon system software:

### 7954 BOOT PROCESS

**Synopsis:** When booting, problems detected with fsck are sometimes handled improperly.

If fsck encounters unrepairable file system problems during the boot process, the system is supposed to leave you in single-user mode so you can repair the file system before you allow users on the system. If the file system is in bad shape (i.e., its superblock is corrupted), the boot process may continue to multiuser mode (with the errors detected by fsck scrolling off the screen) instead of going to single-user mode.

**Workaround:** Bring the system up in single-user mode and run fsck manually. If fsck is not able to clean up the file system, you may need to rebuild the partition using newfs.

### 11895 BOOT PROCESS

**Synopsis:** Fscan causes "node not responding" messages.

Sometimes when fscan says a node is not responding, it is actually bogged down and quickly recovers. No reboot is necessary.

**Workaround:** To eliminate incorrect messages like "Node not responding", you can turn off notify, as follows:

```
FSCAN> set notify off
```

### 12765 BOOT PROCESS

**Synopsis:** fsck -y in /sbin/bcheckrc can destroy data.

In the course of bringing up a customer system after a disk error, fsck -y produced the following message:

```
root inode unallocated  allocate ?
```

Because the -y switch was specified, fsck did allocate the root inode, thus wiping out data on the file system.

**Workaround:** The problem is probably due to an extremely rare condition in the customer's file system. However, if this

possibility bothers you, you may want to consider replacing fsck -y in /sbin/bcheckrc with fsck -p.

#### 12789 BOOT PROCESS

Synopsis: Cannot boot paragon from the TEK X terminals.

The xterminal locks up when the little spinning propeller is supposed to start spinning.

#### 11213 C++

Synopsis: "iCC -A" reveals many header file errors.

There are known header file problems, which are only hinted at in the release notes. These problems are magnified when using strict ANSI (-A). In the include/CC directory, there are three classes of files. One is called the "C++ headers," which correspond to the C++ libraries and the language support library. They include:

common.h	generic.h	libc.h	stream.h
complex.h	iomanip.h	new.h	strstream.h
fstream.h	iostream.h	stdiostream.h	vector.h

These should be able to coexist without problems. However, there are some problems, especially when using strict ANSI, because these files are from the cfront library (which are the libraries we distributed with this compiler), and cfront was much more lax about various things. Please report specific items about these headers so that we can get them corrected.

The second class of include files is the "C headers for the standard library." These are the C components that are included in the working draft of the ANSI/ISO C++ committee. They include, for example, stdio.h, math.h, errno.h, ctype.h, limits.h, etc. These should be allowed to be included in any order. So we'll have to work out problems with these as well.

These last class of include files is the group of C headers that are not part of the standard C library include above, such as termio.h, raw\_hippi.h, vector.h, mach.h, etc. These belong to the system software and not the compiler. The versions massaged for C++ have been provided as a service, and are the versions that are provided with our cfront implementation. There is an effort currently going on to modify our regular system headers so that they are compilable with C++ code. This won't be integrated until R1.4, likely, and may still have some problems, because major modifications are probably needed to correct ordering

dependencies.

Workaround: The current best alternative is to provide your own prototypes in your C++ code. SSD will accept problem reports about these, so that we can check the future versions for correctness.

12771 DOC

6495 HIPPI

Synopsis: No way to report ULA on HIPPI controller from user level.

There is currently no easy way to determine the ULA of a HIPPI controller. Using the "arp -a" command fails with the error message "arp: could not allocate 0 arptab entries."

Workaround: You can determine the ULA by looking on the controller card for its ULA label. Assuming that the system was booted with fscan, you can type -# from the console and then enter the node number of the HIPPI board.

12720 HIPPI

Synopsis: hippy\_open(... O\_EXCL) fails to return error when device already open.

hippy\_open() fails to return -1 and set errno to EACCES when attempting to open a device exclusively and the device is already open, or when two opens come from two separate applications. Also, for the case where a device is already open exclusively, hippy\_open() should return EACCES instead of EBUSY.

8133 IPD

Synopsis: Beyond 64 nodes event tracing is too slow.

Collecting an event trace for an nx application running on more than 64 nodes causes abnormal pm behavior. This behavior is characterized by high perturbation by the performance monitor. Moreover, the amount of time required to output the event data maybe misinterpreted as an application hang.

9100 IPD

Synopsis: system() call not supported by IPD.

Other calls that use system() (such as getenv()) are also not

supported. The error message is \*\*\* ERROR: Executable not found:  
unable to access file /home/<user>/sh.

12792 IPD

Synopsis: Stepping through a call to chdir or fchdir causes  
internal error.

Workaround: Set a breakpoint on the next line rather than use  
step.

12823 IPD

5029 LIBNX

Synopsis: The led() system call has no effect on the LED displays.

11651 LIBNX

Synopsis: exec in compute partition causes memory corruption.

Performing an execl() in the compute partition causes memory  
corruption on subsequent programs. The memory corruption does not  
occur with the standard Unix fork and exec nor if only a fork is  
performed in the compute partition. The failure appears to only  
occur when an exec is performed in the compute partition and occurs  
for programs linked with either -nx or -lnx.

The system must be rebooted to use the nodes.

12283 LIBNX

Synopsis: gcol() has a number of bugs in it that need to be fixed.

10928 LOAD LEVELING

Synopsis: Load leveler takes too large (14%+) performance hit.

6959 MACS

Synopsis: MACS does not track underused node time in batch jobs.

9752 MACS

Synopsis: Multiple applications in a batch job may not be charged  
correctly.

This problem occurs when a batch job is run with qsub and the

script for the batch job runs applications for accounts other than the account in effect when qsub is invoked. In this situation, MACS views all the applications run from the batch job as running under the account in effect when the qsub command is issued and bills the run time for all the applications to that account. From the point of view of acctrep reports, time is not lost, only mis-assigned.

### 3000 MESH UTILS

Synopsis: nx\_nfork() and nx\_loadve() are not returning -1 when they should.

nx\_fork() is not returning -1 in some circumstances, such as:

- \* When the num\_nodes parameter is greater than the size of the partition.
- \* When the node\_list contains invalid node numbers (such as -1 or node numbers greater than the size of the partition).
- \* When the node\_list contains duplicate node numbers. (In this case nx\_nfork() succeeds, but you get a "setptype: Ptype already in use or No active process" error from one of the duplicate children.)
- \* When the process type is invalid (such as -1). (In this case nx\_nfork() succeeds, but you get a "setptype: Invalid ptype" error from the child.)

In other circumstances, nx\_nfork() returns -1 as expected, but the errno is wrong. For example, when the num\_nodes parameter is invalid (for example, -2 or 0), you get a "nx\_nfork(): Not enough space" error instead of the expected EPBADNODE error.

nx\_load() and nx\_loadve() also sometimes fail to correctly indicate an error condition. For example, calling nx\_load() with the pathname of a nonexistent file does not return -1.

Also, whenever nx\_loadve() is given a text file to load (such as /etc/motd):

- \* Patch R1.2.5.1 and earlier: nx\_loadve() returns 1 (success).
- \* Patch R1.2.6/7: nx\_loadve() hangs until the user hits CTRL-C. (no error is reported)

### 8284 MESH UTILS

Synopsis: Higher priority does NOT roll out lower priority job in active layer.

Jobs are assigned to layers based on whether the job will fit in the layer. Priority is not a consideration. Here is an example of the problem in a 16 node partition:

```
Job #1: priority 10 size 8
Job #2: priority 5 size 8
Job #3: priority 10 size 8
```

Jobs #1 and #2 are issued first, followed by job #3. Since job #3 is high priority than job #2, job #3 should preempt job #2, but it is not allowed to. Job #3 must wait until job #1 or #2 finishes.

Workaround: Manually schedule jobs in the order you wish them worked on.

#### 8432 MESH UTILS

Synopsis: Using `nx_pri()` to lower job priority does NOT roll job out for other `ovlp` jobs.

When two jobs of different priorities are run in an overlapping partition and `nx_pri()` is used to lower the priority of the higher priority job, the job is not rolled out when its priority becomes lower than the other job.

There is no workaround for this problem.

#### 9889 MESH UTILS

Synopsis: Partial system hang makes all windows lock on `lspart`, `pspart`, and `mpart`.

#### 10011 MESH UTILS

Synopsis: `nx` applications that `exit(!0)` return 0 exit code.

The exit code returned by a parallel application is the exit code set by the proxy process. If you compile with `-nx`, then you will get an exit code of 1 if any of the internal calls fail (e.g., `nx_waitall()`, `nx_loadve()`), otherwise you get an exit status of 0.

Workaround: Compile via `-lnx` so that you can handle the exit code as you desire.

## 12346 MESH UTILS

Synopsis: Some mkpart -nt doesn't work (rtn error) even though request nodes are available.

## 12769 MESH UTILS

Synopsis: chpart . kills allocator.

## 4703 MESSAGE PASSING

Synopsis: Repeatedly nx\_nfork()ing children crashes machine with assertion in mcmsg\_inq.c.

Under some circumstances, creating very large numbers of child processes by repeatedly calling nx\_nfork() in a single application crashes the system with the error message "Assertion failed: file ../../../../src/mk/kernel/i860ipsc/mcmsg/mcmsg\_inq.c, line 548". Occasionally the machine hangs with no assertion also. The number of child processes that can be created before the problem is seen depends on the process types of the child processes and whether or not they do message passing, but the problem does not appear unless at least 600 child processes are created.

## 4788 MESSAGE PASSING

Synopsis: For a global send to complete, each node must officially receive it.

Global sends (send to node -1) are currently implemented using a tree-structured store-and-forward strategy. This implementation is much faster than the previous strategy, but it requires that each node in the tree must completely receive the message (by calling crecv() or irecv()/msgwait()) before it can pass it on. This means that if one node does not receive the message, other nodes (those "further down the tree") will not receive the message.

## 4886 MESSAGE PASSING

Synopsis: msgcancel doesn't cancel a message id.

Calling msgcancel() does not currently release the specified message ID. After a call to msgcancel(), the number of available asynchronous message IDs is not increased.

## 5838 MESSAGE PASSING

Synopsis: After an "exec", compute node no longer recv's messages



from controlling process.

If a controlling process forks itself onto compute nodes (using `nx_nfork()`) and the child process calls `execvp()` to execute a new program, the newly exec'ed program does not receive messages sent by the controlling process, even if it calls `setpype(0)`.

#### 6168 MISCELLANEOUS

Synopsis: `BADNODES.TXT` is not updated with failed nodes nor accessed by `bootpp`.

When a node is detected as bad, and has been the cause of 'n' successive reboots, the watchdog should add this node to a file called `BADNODES.TXT`, and "bootpp" should read this file and remove any node from the list of valid nodes. This would prevent the watchdog from booting a machine over and over again. However, this file is currently not created by the watchdog and "bootpp" does not use it.

#### 10621 MP NODE

Synopsis: `pfld.l` and `pfld.q` instructions are slower on MP than on GP.

#### 12695 NFS

Synopsis: nfs disk access through second ethernet locks up.

This has only been seen when the user's home directories are on a Sun file server which is on the subnet that contains the Paragon(TM)'s second ethernet. Most of the Sun directories are nfs mounted onto the Paragon through this second ethernet, but a few are mounted through the Hippi/FDDI node. Occasionally, the system goes into a mode where any attempt to access the nfs mounted directories (an `ls`, for example) that are mounted through the second ethernet causes a hang and lock. Even `^C` will not work. Access to the directories that come through the Hippi/FDDI are fine, however. This may not be a Paragon(TM) problem, but a local network problem. Unfortunately, it makes the Paragon unusable.

#### 9184 NQS

Synopsis: NQS dies when wrong path for a pipe queue client is used.

If you create a pipe queue using a wrong pathname to the client, the following three things happen:

1. The NQS qmgr does not detect the problem.
2. If a job is submitted to the (not installed) pipe queue, NQS dies without the user being informed.
3. You must set up NQS again from scratch to get a working configuration again.

## 12195 NQS

Synopsis: "qstat -a" shows "killed" processes as running.

## 12564 NQS

## 12724 NQS

Synopsis: NQS does not enforce per\_req cpu\_lim correctly for multi-app jobs

NQS does not correctly enforce the per\_req cpu\_lim. If an NQS job spawns multiple applications, then the cpu limit on the queue is multiplied by the number of jobs that have been spawned. For example, if you have a 4-node queue with a 60-second per\_req cpu\_lim, if the NQS job simultaneously starts up two different 2-node jobs, the jobs will both run for 120 seconds, thereby defeating the per\_req cpu\_lim.

## 7563 OSF COMMANDS

Synopsis: Transitions between run levels are not clean.

Using the init command to go from multiuser mode to single user mode and back again to multiuser mode does not work cleanly.

Workaround: To go back to multiuser mode reliably, you will need to reboot the system.

## 8474 OSF COMMANDS

Synopsis: "tcopy" fails with an I/O error.

## 8778 OSF COMMANDS

Synopsis: fsck hangs during reset if reset occurred while devices were being formatted.

If the system is reset while one or more RAID devices are being

formatted, fsck will hang during the reboot process.

Workaround: Go into single-user mode and edit /etc/fstab to remove references to the devices that were being formatted when the system was reset. Then, reboot the system and reformat the devices. After reformatting the devices, you can partition the devices and add entries to /etc/fstab again.

#### 9607 OSF COMMANDS

Synopsis: rwhod does not work.

The rwhod command does not work properly. It sends a packet to other hosts only once on startup, instead of every 30 seconds. Also, the initial packet usually contains the wrong load average. The command also does not receive packets from other hosts properly.

There is no workaround for this problem.

#### 9632 OSF COMMANDS

Synopsis: ls -l on directory with extremely large files can hang session.

In directories with extremely large files (for example, a file that fills the entire partition), the command ls -l sometimes hangs. You can kill the process to unhang the session, but processes are left upon exiting that do not go away until the system is rebooted.

#### 10528 OSF COMMANDS

Synopsis: MAKEDEV does not create any devices for 3480 tapes.

Makedev only looks at raidlist and dat tapes and not 3480 tapes:

```
disklist=`getmagic -w BOOT_DISK_NODE_LIST`  
tapelist=`getmagic -w BOOT_DAT_NODE_LIST`
```

Workaround: Manually figure out what the major, minor number is and create a device with rmknod.

#### 10716 OSF COMMANDS

Synopsis: /etc/group gets zeroed out occasionally.

#### 10995 OSF COMMANDS

Synopsis: ps shows incorrect information about server processes on nonlocal nodes.

ps shows all server processes as having essentially the same attributes as the server process on the local node. For all but the local node, ps gives wrong process information.

#### 11678 OSF COMMANDS

Synopsis: newfs after format should fail, but doesn't.

After a low level format, disklabel says it is damaged (as expected), but newfs does not complain. In addition, for every sized partition, the superblock numbers displayed are the same. This is incorrect as they are different sized partitions. An fsck on any of the partitions says everything is clean and a previously mounted filesystem is displayed. This is definitely wrong.

#### 12067 OSF COMMANDS

Synopsis: sed intermittently misses a substitution.

#### 12158 OSF COMMANDS

Synopsis: cron stops intermittently.

#### 5943 OSF LIBS

Synopsis: Use of libc\_r.a results in multiply defined symbols.

If you link to the library libc\_r.a (the reentrant version of the C library), which is required when using pthreads calls, you may see "multiply defined symbol" errors at link time. These errors occur because some functions (such as setgrent() and tzset()) are defined in both libc.a and libc\_r.a.

#### 6409 OSF LIBS

Synopsis: National Language Support locale categories LC\_COLLATE & LC\_CTYPE no longer work.

If you use setlocale() to change the value of LC\_COLLATE or LC\_CTYPE to a language other than the default, the calls tolower(), toupper(), strcoll() and strxfrm() return incorrect values.

#### 8561 OSF LIBS

Synopsis: Floating-point accuracy of printf() and other functions is not IEEE compliant.

The floating point accuracy of atof(), \_dsto2fp(), fcvt(), and ecvt() are not up to IEEE standards when converting double-precision numbers. These functions in turn affect printf() and scanf(). For example, when converting double-precision numbers using printf() or scanf(), the last couple of digits (least-significant) may not be correct.

There is no workaround for this problem.

#### 12165 OSF LIBS

Synopsis: floor and ceil don't work right on denorm and NAN.

#### 7552 OSF MICROKERNEL

Synopsis: Long parallel SAT runs temporarily stall.

#### 7700 OSF MICROKERNEL

Synopsis: kernel vm memory\_object.c line 464 panic encountered running slalom.

#### 8616 OSF MICROKERNEL

Synopsis: OLD NORMA: Heavy paging in an I/O node can cause the node to hang; free page=290.

If the compute-to-I/O node ratio in a system exceeds approximately 32:1 and heavy paging to an I/O node occurs (due to a Mach microkernel memory starvation problem) the I/O node can potentially hang. This problem can also occur if the compute-to-I/O node ratio is very large and the highly parallel M\_RECORD file sharing mode is used to access a PFS file simultaneously from many compute nodes. Depending on the number of nodes and the application's read()/write() request size, this may far exceed the I/O node's ability to buffer incoming or outgoing PFS file data, resulting in heavy paging and possibly the memory starvation hang.

#### 10816 OSF MICROKERNEL

Synopsis: Scripts hang with rsh and qsub in infinite loop.

#### 10999 OSF MICROKERNEL

Synopsis: With MCP on, LEDs are 100% busy when blocked at crecv().

The CPU may be in a spinning loop during `crecv()` when the MCP is on. The green led will then indicate that the CPU is 100% usage.

#### 12244 OSF MICROKERNEL

Synopsis: `ddb` sometimes terminates the `t/uT` command before all threads are shown.

#### 12272 OSF MICROKERNEL

Synopsis: Encountered `assert_wait` during tape operation.

#### 7934 OSF SERVERS

Synopsis: `hostname` or `settime` commands hang system during single-user mode.

In single-user mode, the `hostname` and `settime` commands hang. These commands only work after `bootmesh` has run.

#### 8398 OSF SERVERS

Synopsis: System crashes when FORTRAN `open()` is called from hundreds of nodes.

Opening the same file from hundreds of nodes in a FORTRAN program using an `open()` call can cause a performance bottleneck, which can result in a system crash.

Workaround: Remove any synchronization functions (such as `gsync()` calls) from the program that precede the `open()` call.

#### 8720 OSF SERVERS

Synopsis: `acct` is not usable with Paragon systems.

The standard Unix accounting functions under `/usr/lib/acct` do not function well on Paragon systems. When using these functions, some of the data that is collected is incomplete, due to incompatibility with the multinode environment.

There is no workaround for this problem.

#### 8732 OSF SERVERS

Synopsis: Server does not print all critical warnings to console.

Some O/S errors are only printed to the local node, not to the console.

Workaround: To see these messages, the sysadmin must explicitly use fscan to scan over to the node where the error occurred.

#### 9323 OSF SERVERS

Synopsis: Shutdown commands don't always bring down the system cleanly.

The commands "shutdown," "init 0," "fastboot," "halt," "fasthalt," "reboot," and other shutdown commands do not always shut the system down cleanly.

#### 9741 OSF SERVERS

Synopsis: "ls -l" of a circular symbolic link may panic the system.

This bug occurs when there are circular symbolic links between files on the boot node disk and a remote I/O node disk (non-boot node) or when there are circular symbolic links between files on two or more remote I/O node disks. Performing an "ls -l" in this situation will cause the server to generate thousands of threads, which will eventually panic the system.

Workaround: Do not create circular symbolic links between files on remote I/O nodes. This practice is considered an improper operation on the system.

#### 9993 OSF SERVERS

Synopsis: Dump dies with SIGBUS when reading from raw devices.

#### 11012 OSF SERVERS

Synopsis: QCD and choldm sometimes hang on IO to remote IO node after program completes.

QCD and choldm sometimes hang after completion. Both of these are nx programs that have been running on 1024 nodes. After they complete the nx portion, the proxy running on the service node hangs. The Proxy program is always blocked in fsvr\_unref(). If you then go to the referenced IO node, you find a number of threads with some of the following tracebacks:

1. mutex\_lock\_solid+148

```
mf_token_not_found+144
S_fsvr_token_not_found+50
Xfsvr_token_not_found+74
fsvr_server
```

2. thread\_block+154
  - fp\_unref\_port+218
  - S\_fsvr\_file\_unref+50
  - Xfsvr\_file\_unref+9c
  - fsvr\_server+ac

The system doesn't exhibit the hang when the filesystem is moved to the bootnode. This appears to be a timing issue and can be reproduced only on a large (1024 nodes) system.

#### 11642 OSF SERVERS

Synopsis: Parallel SATs hang on exit with proxies paged out.

#### 11743 OSF SERVERS

Synopsis: Code executing kill(0,9) in IPD will wire down 4 pages of memory.

Running the following code under IPD, kills the TAM and wires down four pages of memory. Running the same executable without IPD leaves the wired page count unchanged.

```
#include <nx.h>
#include <stdio.h>
main()
{
  if(mynode() == 0) {
    printf("Now killing application...\n");
    kill(0,9);
  } else {
    sleep(10);
  }
}
```

#### 12005 OSF SERVERS

Synopsis: Killing IPD causes server panic at spin\_try\_lock()+8 in pproc\_set\_attr()+54.

#### 12245 OSF SERVERS

Synopsis: [node 2] server panic: vproc\_db2: NULL vproc.



## 12261 OSF SERVERS

Synopsis: \*part commands hang after rmpart -f tried to kill hung application.

## 12336 OSF SERVERS

Synopsis: \*part commands hang with NO compute nodes wired down.

The \*part commands hanging are just a symptom of an underlying problem with the server or kernel. There are probably multiple causes, multiple bugs. Several bugs share some similar characteristics:

Bug #11787 At least one compute node has only 29 free pages.

Bug #12261 R1.3 software, all nodes have reasonable number of free pages.

Bug #12336 R1.2 software, all nodes have reasonable number of free pages.

If your symptoms match one of the above scenarios, then please add a comment to the appropriate bug report. If you have other unique symptoms, then please open a new bug.

## 12482 OSF SERVERS

Synopsis: Mixed compilation code hangs, won't kill, and causes \*part hang.

## 12593 OSF SERVERS

Synopsis: A user code can trigger a deadlock in get\_data\_token().

A user parallel application that performs a large number of open(), close(), fstat(), chmod(), and write() calls to the same file can cause a deadlock situation in the get\_data\_token () routine of the server.

Basically, some thread wants the token and asks the owner to release it. But the owner does not think it has the token and replies with a "fsvr\_token\_not\_found". This request/reply cycle seems to go on forever once it gets started.

Workaround: Unblock the token queue in get\_data\_token(). Doing so causes the "fsvr\_fstat" request in e\_fstat() to receive a reply and

`e_fstat()` to eventually call `fdt_unref_entry()`. This time around the `refcnt` of the `fdte` will be zero and `fsvr_token_release()` will be called to release the token.

#### 10318 OSF SYSTEM CALLS

Synopsis: `plock(2)` with data segment greater than physical memory hangs machine.

If you call `plock(2)` to lock your data segment (`DATLOCK`) and your data segment is greater than the physical memory on the node will hang the paragon. The `plock()` call uses the `vm_wire()` mach call which cannot bail out in trouble. It will wire up pages until it is wedged.

#### 10898 OSF SYSTEM CALLS

Synopsis: Processes reading `stdin` in iomode `M_UNIX` don't have their own file pointers.

A test program reads `stdin` and redirects it to a file. Each node should read from the beginning of the file, but they read in sequence. The test program allows for using the default iomode (`M_UNIX`), or setting the iomode through a command argument. In either case, the nodes share a file pointer rather than having their own. That is to say, the nodes read from the file in turn, and they read sequentially through the file instead of each process reading from the start of the file. In at least one case, just calling `setiomode` caused an emulator error.

#### 9138 PAGING TREES

Synopsis: "`PAGE_TO rz0c`" in `DEVCONF.TXT` handled improperly.

Placing a `PAGE_TO rz0x` list in `DEVCONF.TXT` in an attempt to construct a paging tree in RAID partitions does not work as expected. A workaround for this problem is to enter the `PAGE_TO` list in `MAGIC.MASTER` manually, instead of placing it in `DEVCONF.TXT`.

#### 9139 PAGING TREES

Synopsis: Double colon incorrectly placed in `PAGER_NODE` list for auto paging tree.

When attempting to generate an automatic paging tree using the `-P1` flag for `bootpp` in the "reset" script, an error may occur on reset. This error may be the result of a double colon (`::`) that is

incorrectly generated in the PAGER\_NODE line of bootmagic.

Workaround: Enter the PAGER\_NODE line in MAGIC.MASTER manually and avoid using "-P1" to generate an automatic paging tree.

#### 11414 PARAIDE

Synopsis: Having multiple SPV sessions from Paraide freezes all windows.

#### 12690 PFS

Synopsis: Applications that open hundreds of PFS files cause boot node to slow.

An application that attempts to read hundreds of files in PFS from each node uses the boot node to resolve the pathnames. The boot node eventually becomes overloaded and eventually slows down the machine so much that the autobooter decides to reboot it.

#### 12881 PFS

#### 11848 PMAKE

Synopsis: pmake does not return macs accounting msgs from nx\_initve().

#### 7468 PTHREADS

Synopsis: pthread\_setcancel sometimes fails.

When general cancellation of pthreads is blocked by pthread\_setcancel(), pthreads are sometimes cancelled anyway, when functions that set up "cancellation points" are used. The pthread\_cond\_wait(), pthread\_join(), and pthread\_cond\_timedwait() functions (which are "cancellation points") incorrectly carry out cancellation requests even though general cancellation of threads is blocked.

Workaround: Add additional synchronization to insure that no cancels are pending prior to the call of pthread\_cond\_wait(), pthread\_join(), and pthread\_cond\_timedwait() and that cancels do not occur during the wait.

#### 8136 PTHREADS

Synopsis: SIGCONT does not always restart all pthreads.

The SIGCONT signal does not always restart the pthreads of a process that have been suspended with the SIGSTOP signal.

There is no workaround for this problem.

#### 8180 PTHREADS

Synopsis: Pthread internal error occurs during a file I/O operation.

You cannot cancel a pthread during a file I/O operation such as `creat()`. Doing so will result in a "pthread internal error in thread\_suspend" message and the thread will be left in an indeterminant state.

#### 9086 PTHREADS

Synopsis: `sigwait()` causes hang on an MP system.

The `sigwait()` function should return the expected asynchronous signal number when the signal is received. Instead, when run on an MP system, the function causes a hang. CTRL-C will relieve the hang.

There is no workaround for this problem.

#### 11554 PTHREADS

Synopsis: Can only create 17 pthreads on GP system with 64 M bytes stacksize.

A test program is unable to create 21 pthreads. The test fails on `pthread_create` Error: Not enough space. The test, `hello`, is a unix process running on a service node. The test passes on MP systems and fails on GP system.

#### 12534 RAID UTILITIES

Synopsis: Deleting or adding LUN forces low level format.

Using the `add LUN` or `Create LUN` in ACE forces a low level format on 4Gbyte drives when the drive geometry values for spare sectors per track do not match the RAID controller default values.

Workaround: Turn on bits 0 & 1 of byte 24 of mode page 2b when requesting the RAID to "add LUN". This prevents the RAID controller from trying to modify the drive parameters.

## 12899 RAID UTILITIES

## 4401 S/W INSTALLATION

Synopsis: If netmask/broadcast not set as Paragon expects, installation fails.

If the netmask/broadcast in the install procedure is set to a C class address, the install fails. It succeeds if the netmask/broadcast is set to a B class address.

Workaround: Start the installation over from scratch and install a B class address. Then, once the system is successfully installed, reset the appropriate parameter(s) in /etc/rc.config to the correct value(s) for your site and reboot the system to complete the installation.

## 11378 SAT

Synopsis: Output report not updated if sat exits on first error or on interrupt.

If you run `sat -x -o <report>` the report will not be updated prior to exit. The report is currently updated after each iteration of the test list but should always be updated prior to exit unless system resources are unavailable. This problem also exists for exiting because of an interrupting signal other than the timer alarm (-m option).

## 12886 SPV

## 12684 SUNMOS

Synopsis: Sunmos doesn't access MDC memory.

## 9967 TCP/IP

Synopsis: Route delete caused `catch_exception_raise: _rn_delete(f016d2c0,1000,2000)+108.`

## 11090 TCP/IP

Synopsis: traceroute command core dumps.

When the traceroute command is invoked, 1 line gets printed and then it core dumps.

Workaround: Use the -s switch.

## 11956 TCP/IP

Synopsis: Changing routes corrupts the routing table displayed by netstat -r.

Each time the route command is used to change a route in the routing table, lines get deleted from the routing table displayed by netstat -r.

## 12584 TCP/IP

Synopsis: Server panic: ifa\_real\_with\_af: no real i/f for af=2.

A partially-edited /sbin/ini.d/inet file in which a second ifconfig command was added for a device that didn't exist caused the following panic message:

```
ifconfig: ioctl (SIOCGIFFLAGS): no such interface
```

## 4708 UFS

Synopsis: Mount of two partitions on same directory permitted.

It is currently possible to mount more than one disk partition on the same directory. The second partition mounted "masks" the first (that is, the contents of the mount point appear to be the contents of the second partition mounted). Once this has occurred, attempting to unmount one of these partitions may actually unmount the other.

## 11622 UFS

Synopsis: Paragon loses days or hours of file data after an unsuccessful shutdown.

## 12124 XIPD

Synopsis: XIPD can't display class methods of same-name classes defined in functions.

XIPD does not have enough information to specify a method in order to list the method's source code in the following situation:

In a single file, file.C, two routines, RoutineOne() and RoutineTwo(), declare and define a class call LocalClass. This class is local only to the routine that it's defined in.

LocalClass has a method called Print() in both classes.

Trying to display either of the Print() methods for LocalClass, IPD emits the following error:

```
*** ERROR: search failed
***      ambiguous name: file.C{}LocalClass::Print(void)
```

Since there's more than one LocalClass::Print(void) in file.C{}. XIPD actually needs to have some way of getting the start and end line number of the routine.

There is no workaround.

## Fixed Bug List

The following lists the bugs fixed since Release 1.2 of the Paragon system software:

### 8330 BOOT PROCESS

Synopsis: Watchdog incorrectly says "Node not responding."

Sometimes when fscan says a node is not responding, it is actually bogged down and quickly recovers. No reboot is necessary. To eliminate incorrect messages like "Node not responding", you can turn off notify, as follows:

```
FSCAN> set notify off
```

### 9110 BOOT PROCESS

Synopsis: fscan is reporting incorrect "Node xx (cbs=xx) is in the DEBUGGER!" message.

The fscan program occasionally outputs incorrect "Node xx (cbs=xx) is in the DEBUGGER!" messages on GP nodes. To eliminate these incorrect messages, you can turn off notify, as follows:

```
FSCAN> set notify off
```

### 9989 BOOT PROCESS

Synopsis: reset autocfg produces corrupt SYSCONFIG.BIN

### 11286 BOOT PROCESS

Synopsis: system not config'd to reboot may reboot anyway..

### 9674 DEVICE DRIVERS

Synopsis: remote fsck hangs when disk array is in degraded state

### 9673 DOC

Synopsis: MACS commands have no return codes documented

The return codes for the commands in the MACS code set (macadmin, macalloc, maclist, jrec, acctrep, si, macupdate, dbcreate) are not documented.

### 9808 DOC



Synopsis: R1.2 Fortran Systems Calls Manual is missing last 20 pages

9836 DOC

Synopsis: Release Notes recommend "tar" for backing up /home - should use "dump"

10045 DOC

Synopsis: A crecv call in application caused crecv: Too many requests

10117 DOC

Synopsis: More than 6 pthreads crashes the emulator

10247 DOC

Synopsis: Sample Fortran example on Paragon is incorrect.

10906 DOC

Synopsis: Missing diskquota support should be mentioned in Release notes Limitations

11489 DOC

Synopsis: Bringing the DS down will sometimes hose the Paragon needs documentation.

5907 IPD

Synopsis: instrument -write is non-functional

Under some circumstances, the IPD command "instrument -write" does not create a performance data file and leaves instrumentation set to "on".

7958 IPD

Synopsis: Profiling data not written if IPD is exited prematurely

If you exit IPD prior to receiving a message stating that the performance monitoring data has been written, the data may be lost. If you are monitoring many nodes with IPD, the writing of this data may take from several minutes to one half hour depending

on the number of nodes being monitored.

#### 8688 IPD

Synopsis: Collecting performance monitoring data fails if host and nodes are instrumented.

If you attempt to collect performance data while simultaneously monitoring both the controlling process and the compute node processes, the application can appear to hang. This practice can also lead to incomplete data being collected.

Workaround: To collect complete performance data without hanging ETS and/or the application, instrument either the controlling process or the node processes but not both.

#### 9772 IPD

Synopsis: setting a watchpoint and using the step command hangs IPD

If you set a watchpoint in a subroutine to watch a parameter being passed to the subroutine and then use the step command after the program has stopped at the watchpoint, IPD will hang and the whole IPD debug session will be lost. This is not a general problem. It only happens when you set watchpoints to monitor parameters being passed to functions.

#### 11192 IPD

Synopsis: Trying to turn prof, gprof, & paragraph instr. off causes int. err.

#### 11976 IPD

Synopsis: ipd complains bad symbol table while displaying Fortran structure

#### 12370 IPD

Synopsis: IPD produces internal error while attempting to profile a code

#### 8791 LIBNX

Synopsis: global csend() does not work with hrecv()

The handler interface between csend() and hrecv() does not work properly when sending messages globally (csend(-1)), resulting in

the message information being incorrectly received. The workaround is to use `csend()` only to send messages directly to nodes, rather than globally.

## 10971 LIBNX

Synopsis: global `csend` to `crecvx` hangs on 7 nodes or more..

## 7937 MACS

Synopsis: MACS database-dependent utilities don't handle "acctonly" mode correctly

The MACS database-dependent utilities (such as `macupdate`) are intended for use in the "macdmode", not in the "acctonly" mode. If you use these utilities in the `acctonly` mode, you will get an error message; however, the error message will not indicate that you are in the wrong mode.

## 8753 MACS

Synopsis: MACS doesn't notice when it can no longer write logfiles

If the `/var` file system fills up, preventing the writing of future MACS logs, the system administrator is not notified. To avoid losing log data, check `/var` periodically and remove old log data so that the `/var` partition does not fill up unnoticed.

## 9226 MACS

Synopsis: Due to MACS startup script errors, SI may not produce the correct log entry

A false boot record may get written due to the way the start script tries to determine if a reboot just happened. This problem occurs the day after a reboot when MACS is stopped and started but the system is not rebooted. There is no workaround for this problem.

## 9273 MACS

Synopsis: R1.2 MACS commands cannot process R1.1 log files

The only workaround for this problem is to install an R1.1 version of MACS to use in reading old log files.

## 9825 MACS

Synopsis: Records in logfile may get out of sync causing a double

charge

12494 MACS

Synopsis: macd dies and accounting confused after account runs out of time

7624 MESH UTILS

Synopsis: SMD returns inconsistent timing values to MACS

The CPU time reported by SMD can be significantly larger than the correct time, which is the (wall clock time) \* (number of nodes). There is not workaround for this problem if it occurs.

8464 MESH UTILS

Synopsis: Changing partition from space-share to gang-scheduled may cause sys. instability

Using the following format of the chpart command

```
chpart -rq $RQ <SPS_part>
```

to change a space-shared partition to a gang-scheduled partition can cause the REJECT\_PLK=1 control in the /etc/nx/allocator.config file to be bypassed, resulting in system instability.

9408 MESH UTILS

Synopsis: allocator does not recognise MIN\_RQ\_ALLOWED in config file if value is in minute

The allocator does not recognize values in minutes for the MIN\_RQ\_ALLOWED configuration parameter in the allocator.config file; however, it does recognize values in hours or seconds.

9561 MESH UTILS

Synopsis: application -sz hXw -pn partname attempts to run application on bad nodes

10016 MESH UTILS

Synopsis: nx\_chpart\_name() call fails when NQS tries to chg from prime to non-prime time

11083 MESH UTILS

Synopsis: nx\_loadve does not return an error if the setptype call fails

#### 12235 MESH UTILS

Synopsis: nx\_loadve takes unacceptably too long time.

#### 7647 MESSAGE PASSING

Synopsis: msgmerge(mid,mid) causes subsequent msgdone() to hang

Calling msgmerge(mid,mid), where you specify a valid mid twice, causes a subsequent msgdone() to hang if a matching message is pending. To avoid this problem, don't call msgmerge(mid1, mid2) where mid1 and mid2 are identical.

#### 8358 MESSAGE PASSING

Synopsis: Output is missing/corrupted/doubled when printing from hrecv() handler

This problem results from the print() functions being called from two different threads running in the same process. Many of the functions in libc (including printf()) are not thread safe. Invoking a handler with hrecv() starts a new thread for the handler. Then, if this thread and another thread call the printf() function at the same time, the resulting output can have missing, corrupted, and/or double characters.

#### 9434 MESSAGE PASSING

Synopsis: message exchange code hangs paragon with -plk option

#### 3803 NQS

Synopsis: QMGR set per-(process/request) limits not enforced.

#### 12062 NQS

Synopsis: NQS basis permission of queue works only for login/default group.

#### 9377 OSF COMMANDS

Synopsis: invoking rlogin daemon with -l option does not work

#### 9454 OSF COMMANDS

Synopsis: PAGE\_TO <1..2> rz0d bootmagic string doesn't work for >1 nodes.

The PAGE\_TO bootmagic, system-configuration parameter (used to page to a a non-default disk partition, i.e. not rz0b) may not work for more than one node. It is believed that this is a parsing problem with the delimiter (...). For example, if you manually enter a PAGE\_TO bootmagic string in MAGIC.MASTER in a different format (such as "PAGE\_TO=<1,2>rz0d") the string is handled correctly.

#### 5484 OSF MICROKERNEL

Synopsis: Get " sl0: silo overflow" messages while in vi editing session.

While editing any file using "vi" on the Paragon console with "async", you may occasionally see the message " sl0: silo overflow". If you see this message, undo the last change and press <Ctrl-L> to refresh the screen.

#### 6733 OSF MICROKERNEL

Synopsis: code executes with -Mnoxp, fails with -Mxp

Certain programs that exhibit unexpected behavior during asynchronous message passing when compiled with the default compiler flag -Mxp work as expected when compiled with -Mnoxp. The problem appears to be associated with the use of vectorized loops; if this happens to you, try recompiling any code that uses vectorized loops with -Mnoxp.

#### 6759 OSF MICROKERNEL

Synopsis: multiple ^C when loading // apps hangs window, sometimes hangs system

If you attempt to interrupt a parallel application as it is loading by repeatedly pressing <Ctrl-C>, you can hang the session and may hang the entire system. Note that pressing a single <Ctrl-C> works just fine; the problem only occurs if you press several <Ctrl-C>'s.

#### 6832 OSF MICROKERNEL

Synopsis: Writing PFS file from 64 node in M\_RECORD mode caused kernel panic

Under some circumstances, writing large amounts of data to a PFS file system in I/O mode M\_RECORD can cause the kernel to panic. One test case that shows the problem causes the panic by writing from 64 compute nodes to 1 I/O node using a 64K-byte request size. (Note that it is strongly encouraged that PFS applications use 32 or fewer compute nodes per I/O node.)

## 10968 OSF MICROKERNEL

Synopsis: Simple message passing code causes Assertion in msgp\_nxdat.c line 1800

## 11170 OSF MICROKERNEL

Synopsis: core dumping from compute partition can cause NORMA lockup

## 7923 OSF SERVERS

Synopsis: Multiple concurrent opens can slow down server such that it appears hung

Any application program that concurrently opens a large number of files in a PFS or UFS file system can cause the system to slow down so much that it appears to be hung. There is no workaround for this problem other than to limit the number of concurrent opens that an application conducts.

## 9992 OSF SERVERS

Synopsis: catch exception raise panic at getmntid()+4c

## 10702 OSF SERVERS

Synopsis: Cannot get more than 32 logins to the paragon

## 11040 OSF SERVERS

Synopsis: simple proxy program calling killcube() seems to cause memory leak

## 10551 OSF SYSTEM CALLS

Synopsis: system stats not returning correct values

## 12264 OSF SYSTEM CALLS

Synopsis: system stats not returning correct values for system

calls

9201 PARAGRAPH

Synopsis: Insufficient trace file space cause ParaGraph to dump core

This problem can occur if a trace file is generated in a file system that does not have sufficient space to hold the trace data. As a result, the trace merge may complete but leave a truncated trace file that will cause ParaGraph to dump core. To avoid this problem, make sure that enough file space is available to hold the trace file to be generated.

6573 PARAIDE

Synopsis: scanf() from stdin doesn't read more than one line under XTF

11248 SOURCE CODE PRODUCT

Synopsis: bld0/cmds/src/etc/termcap/termcap.src is truncated in PSCP

11250 SOURCE CODE PRODUCT

Synopsis: termcap fails to build using "reorder" file that is part of the PSCP

9614 TAPE DRIVE

Synopsis: Assertion failed: adapters/scsi\_53C94\_hdw.c", line 2579 from 120m tape

9397 TCP/IP

Synopsis: terminating telnet connections to paragon sometimes creates defunct process

9420 VSOCKET

Synopsis: NQS netdaemon gets stuck in accept on the port, can't kill it

9893 VSOCKET

Synopsis: After "route add -net" cmd, netstat displays garbage for netmasks.