

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

(503) 696-8080



June 1994

Dear Paragon™ Customer:

This package contains your Release 1.2 Paragon ParAide toolset software. Please read through the documentation and distribute it to anyone intending to use the tools.

Before using the tools:

- **Read this letter completely.**
- **Verify the contents of this package.**
- **Read the *Paragon™ Tools Release 1.2 Software Product Release Notes*.**

Package Contents

Your Paragon tools software package is shipped in a single box. Please verify that the box includes the items listed in Table 1 (Installation Media) and Table 2 (Documentation). If any items are missing, or if you have any questions, please contact Intel Supercomputer Systems Division as described in the section "Comments and Assistance" on page 3. Your tools manuals are shipped with the Paragon™ manual set.



Table 1. Installation Media

Description	Order Number
Paragon™ ParAide Software Release 1.2 cartridge tape	313088-001

Table 2. Documentation

Description	Order Number
<i>Paragon™ Tools Release 1.2 Software Product Release Notes</i>	313061-001

Restrictions and Limitations of Tools Release 1.2

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™ Tools Release 1.2 Software Product Release Notes* for known limitations and available workarounds.

Installation

For directions on how to install the Paragon™ tools software, refer to the *Paragon™ Tools Release 1.2 Software Product 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 Supercomputer Systems Division Customer Support for assistance in answering your questions.



Comments and Assistance

Intel Supercomputer 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

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.
Supercomputer Systems Division
5-6 Tokodai, Tsukuba City
Ibaraki-Ken 300-26
Japan
0298-47-8904

United Kingdom Intel Corporation (UK) Ltd.
Supercomputer System 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
Supercomputer Systems Division
15201 N.W. Greenbrier Parkway
Beaverton, Oregon 97006
U.S.A.
(503) 629-7600 (Monday through Friday, 8 AM to 5 PM Pacific Time)
Fax: (503) 629-9147

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 (503-629-5322)
joanne@ssd.intel.com (Internet)

Sincerely,



Steve Cannon

Product Marketing Manager
Intel Supercomputer Systems Division

Paragon is a registered trademark of Intel Corporation.

Copyright © 1994 Intel Corporation



June 1994

Order Number: 313061-001

Paragon™ Tools
Release 1.2
Software Product Release Notes

Intel® Corporation

Copyright ©1994 by Intel Supercomputer 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

ParaGraph was developed at Oak Ridge National Laboratory by M. Heath and J. Finger under a research grant from D.O.E.

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 trademark of UNIX System Laboratories

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 provide the latest information on Release 1.2 of the Paragon Tools.

Organization

- | | |
|-----------|--|
| Chapter 1 | Introduces the new features of the Release 1.2 Tools. |
| Chapter 2 | Provides installation instructions for this release. |
| Chapter 3 | Describes known limitations and workarounds of this release. |

Notational Conventions

This manual uses the following notational conventions:

Bold Identifies command names and switches, system call names, reserved words, and other items that must be used exactly as shown.

Italic 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 *italic* type style and flush with the right margin.

Bold-Italic-Monospace

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.
- ... (Ellipsis dots) 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 Documents

For more information about the Paragon tools, refer to the *Paragon™ Application Tools User's Guide*.

For more information about Paragon documentation, refer to the *Paragon™ System Technical Documentation Guide*.

Comments and Assistance

Intel Supercomputer 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

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.
Supercomputer Systems Division
 5-6 Tokodai, Tsukuba City
 Ibaraki-Ken 300-26
 Japan
 0298-47-8904

United Kingdom Intel Corporation (UK) Ltd.
Supercomputer System 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
Supercomputer Systems Division
 15201 N.W. Greenbrier Parkway
 Beaverton, Oregon 97006
 U.S.A.
 (503) 629-7600 (Monday through Friday, 8 AM to 5 PM Pacific Time)
 Fax: (503) 629-9147

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

Table of Contents

Chapter 1 Product Features

New Features	1-1
ParAide	1-1
SPV	1-1
ParaGraph	1-2
XIPD	1-2

Chapter 2 Installation

Installing the Native Paragon™ Tools	2-1
Installing the Cross Development Tools	2-3

Chapter 3

Guidelines and Limitations

Interactive Parallel Debugger (IPD)	3-1
Using IPD on Applications with NX Handler Routines	3-1
The msgqueue Command and Global Sends	3-3
Performance Monitoring	3-3
Scaling IPD	3-3
Graphical Tools	3-3
Paragraph and NX Handler-Invoking Routines	3-4
XIPD	3-4
SPV X Resources for Pre-X11R5 Servers	3-4
SPV and Memory Usage	3-4
PostScript Copies of the Manuals and Release Notes	3-5
Open Bugs	3-5
Bugs Fixed Since Release 1.1	3-11

New Features

The following sections describe new features and changes to the Paragon tools for this release.

ParAide

- ParAide has a new *Load Application* dialog. The new dialog simplifies the process of loading an application. The new dialog contains areas to do the following:
 - Specify an application to load
 - Specify application options
 - Select a partition to load into
 - Specify nodes to load into

The new *Load Application* dialog replaces the separate partition selection and node selection dialogs that existed in the previous release.

SPV

- The *Change Data Display Options* dialog has been expanded to allow you to display the following memory utilization values:
 - Memory usage
 - Wired memory usage
 - Page faults per second

ParaGraph

- You can now set start and stop times for displays that scroll with time by pressing and dragging the middle mouse button. Holding down the mouse button and dragging the mouse pointer forms a rectangle. When you release the mouse button, the rectangle defines the new start and stop times, and the time unit is adjusted so the new interval fits the default window size. These new values are shown in the *Configure* dialog, which automatically pops up. You can apply them or dismiss them with the *Reset* or *Cancel* buttons.

This allows you to examine a specific portion of a simulation simply by dragging over that portion to form a rectangle, applying the changes, resetting the simulation, and starting a new simulation run. The easiest method to restore the default start time, stop time, and time unit is to reload the tracefile using the *Open* menu item from the *File* menu.

When you set the start and stop time by forming a rectangle over a portion of a simulation, the start time is defined by the first event that occurs within the rectangle, and the stop time is defined by the first event that follows the end of the rectangle. It is possible to form a rectangle that contains no events. If this happens, the following message is displayed:

```
Defined stop time reached
```

- Displays that scroll with time now have a legend that shows the current value for one simulation time unit. This feature makes it easier to correlate simulation time units to real execution time.
- The message log now has a *Help* button.
- The *Coordinate Info* display now has a scrollbar to allow you to scroll through the information printed into the display.
- The *Coordinate Info* display now pops up automatically if you press the left mouse button while the cursor is in one of the displays supported by the *Coordinate Info* display. In addition, you can still bring up the *Coordinate Info* display by selecting the menu entry from the *Other* menu.

XIPD

- The following X resources have been added to XIPD:

XIpd.partitionName

A string that defines the default partition into which XIPD loads. If the partition does not exist, XIPD defaults to the *.compute* partition. This resource overrides the setting of *NX_DFLT_PART*.

XIpd.applicationName

A string that defines the default application XIPD puts in the *Load Application* dialog text field for the program to load.

XIpd.viewpointExecutionDefault

A boolean value that controls whether the *Viewpoint Applies to Execution* should be on or off by default when a new session is started. The default is **True**.

XIpd.doRemoteLogin

A boolean value that controls whether or not the login dialog should always come up, even if using XIPD on a local machine. The default is **False**.

If you are logged on to the Paragon system on which you are invoking XIPD, the default is not to display the login dialog. However, you must still specify the local Paragon hostname on the **xipd** command line to avoid bringing up the login dialog.

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

-pn *partition_name*

Sets the **XIpd.partitionName** resource to *partition_name*.

{-program | -prog | -app} *application_name*

Sets the **XIpd.applicationName** resource to *application_name*.

- XIPD has a new *Load Application* dialog. The new dialog simplifies the process of loading an application. The new dialog contains areas to do the following:
 - Specify an application to load
 - Specify application options
 - Select a partition to load into
 - Specify nodes to load into

The new *Load Application* dialog replaces the separate partition selection and node selection dialogs that existed in the previous release.

NOTE

These instructions assume that SCO UNIX and the SCO Open Desktop are installed on the diagnostic station. If this software is not installed, contact Intel SSD Customer Support.

Installing the Native Paragon™ Tools

Installation Time:

Approximately 15 minutes.

Installation Medium:

One 0.25-inch QIC 150 cartridge tape labeled Paragon™ ParAide Software (313088-001).

The tape contains installation **tar** files for the Paragon tools and a documentation **tar** file. The installation **tar** file for the Paragon system is called *tools.tar.Z*. The installation **tar** file for the cross-development environment is called *crstools.tar.Z*. The documentation **tar** file is called *tools.doc.tar.Z*.

NOTE

You can install the native Paragon tools as part of the system software *postboot* installation process. For information on installing the native tools with *postboot*, refer to the *Paragon System Software Release Notes*.

1. Log in to the diagnostic station as *root*.
2. Copy the *tar* files from the release tape into */u/tmp* on the diagnostic station.
 - A. Make */u/tmp* your working directory.


```
DS# cd /u/tmp
```
 - B. Insert the release tape into the cartridge tape drive on the diagnostic station.
 - C. Copy the files for the Paragon system.


```
DS# tar xvf /dev/rStp0 tools.tar.Z tools.doc.tar.Z
```
3. Log in to the Paragon system as *root*.
4. Establish an *ftp* connection with the server containing the files *tools.tar.Z* and *tools.doc.tar.Z*:

```
# cd /tmp
# ftp IP address of server with distribution files
Name: login_name
Password: password
ftp> cd path to distribution files
ftp> bin
ftp> get tools.tar.Z
ftp> get tools.doc.tar.Z
ftp> bye
```

5. Use the following steps to uncompress and untar the files into the */* directory.

```
# uncompress tools.tar.Z
# cd /
# tar xf /tmp/tools.tar
# rm /tmp/tools.tar
# cd /tmp
# uncompress tools.doc.tar.Z
# cd /usr/share
# tar xf /tmp/tools.doc.tar
# rm /tmp/tools.doc.tar
```

If you are having disk space problems, use *zcat* to uncompress and extract the files. This uses less space, but takes longer.

```
# cd /
# zcat /tmp/tools.tar.Z | tar xvf -
# rm /tmp/tools.tar.Z
# cd /usr/share
# zcat /tmp/tools.doc.tar.Z | tar xvf -
# rm /tmp/tools.doc.tar.Z
```


Installing the Cross Development Tools

Installation Time: Approximately 15 minutes.

Installation Medium: One 0.25-inch QIC 150 cartridge tape labeled Paragon™ ParAide Software (313088-001).

The tape contains installation **tar** files for the Paragon tools and a documentation **tar** file. The installation **tar** file for the Paragon system is called *tools.tar.Z*. The installation **tar** file for the cross-development environment is called *crstools.tar.Z*. The documentation **tar** file is called *tools.doc.tar.Z*.

1. Log in to the diagnostic station as *root*.
2. Copy the installation **tar** file from the release tape into */u/tmp* on the diagnostic station.
 - A. Make */u/tmp* your working directory.

```
DS# cd /u/tmp
```
 - B. Insert the release tape into the cartridge tape drive on the diagnostic station.
 - C. Copy the file for the cross-development system.

```
DS# tar xvf /dev/rStp0 crstools.tar.Z
```
3. Log in to the Paragon system as *root*.
4. Use the *mksysfiles* script which creates in the current working directory a *sysfile.tar.Z* file that includes libraries, include files and documentation.

```
# cd /tmp
# /usr/bin/mksysfiles
```

5. Establish an ftp connection with the cross development system, and put *sysfiles.tar.Z* into the /tmp directory:

```
# ftp IP address of cross development system
Name: login_name
Password: password
ftp> cd /tmp
ftp> bin
ftp> put sysfile.tar.Z
ftp> bye
```

6. Log in to the cross development system, and use ftp to get the cross development tar files from the distribution server.

```
CROSS# cd /tmp
CROSS# ftp IP address of server with distribution files
Name: login_name
Password: password
ftp> cd path to distribution files
ftp> bin
ftp> get crstools.tar.Z
ftp> bye
```

7. Create a /usr/xdev directory (or other directory of your choice) to install the cross development tools into. Set and export the *PARAGON_XDEV* variable to refer to the directory. Change your *PATH* to include the *\$PARAGON_XDEV/paragon/bin.sun4* directory (or for SGI platforms, *\$PARAGON_XDEV/paragon/bin.sgi*).

```
CROSS# mkdir /usr/xdev
CROSS# cd /usr/xdev
CROSS# PARAGON_XDEV=/usr/xdev
CROSS# export PARAGON_XDEV
CROSS# PATH=$PATH:$PARAGON_XDEV/paragon/bin.sun4
```

8. Use the following steps to uncompress and untar the distribution files into the *\$PARAGON_XDEV* directory.

```
CROSS# cd $PARAGON_XDEV
CROSS# zcat /tmp/crstools.tar.Z | tar xf -
CROSS# rm /tmp/crstools.tar.Z
```

9. Uncompress and untar the *sysfiles.tar.Z* file into the *\$PARAGON_XDEV/paragon* directory.

```
CROSS# cd paragon
CROSS# zcat /tmp/sysfiles.tar.Z | tar xf -
CROSS# rm /tmp/sysfiles.tar.Z
```

Guidelines and Limitations

3

This chapter describes limitations to the Paragon tools for this release and provides some guidelines for using the tools. At the end of this chapter is a current list of bugs for the tools and a list of the bugs fixed since Release 1.1. The list of bugs is updated just before shipment, and the lists are also available online in the files `/usr/share/release_notes/tools_buglist` and `/usr/share/release_notes/tools_fixed` on the Paragon system.

Interactive Parallel Debugger (IPD)

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()`), the following INFO message is displayed:

```
*** INFO:   A NX handler-invoking routine such as hrecv(), hsend(),
            or hsendrecv() has been encountered by at least one
            process under IPD.
```

IPD has limited control of the new thread created by this call. Use "frame" to determine which thread is visible to IPD whenever the application is stopped. ParaGraph usage is restricted to programs that do not use handler routines. For details, see the release notes in `/usr/share/release_notes` on the Paragon.

This 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.

“Limited control” means that 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_rcv_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.

PostScript Copies of the Manuals and Release Notes

PostScript copies of the Paragon manuals are available in the directory */usr/share/ps.docs* on the Paragon system. This directory also contains the file *README.tools*, which lists the tools manuals contained in the directory.

PostScript copies of the tools release notes can be found in the directory */usr/share/release_notes*.

Open Bugs

This section lists the currently-open bugs for the Paragon tools. The next section lists the bugs fixed since Release 1.1. An ASCII-only version of the list of tools bugs (*tools_buglist*) and list of bugs fixed since Release 1.1 (*tools_fixed*) can also be found in the */usr/share/release_notes* directory. In the bug lists, the number on the left side of the first line for each bug is the bug number. Use this number when communicating with SSD Customer Support about the bug.

9327 DGL DGL has performance problems on R1.1.5 and R1.2 systems.

DGL applications have exhibited some performance problems on Paragon R1.1.5 and R1.2 systems. The general problem is poor TCP/IP communication rates. The following things can be done to improve performance, if you experience DGL performance problems:

1. Tune your DGL code to minimize the amount of I/O between the Paragon and your Sun or SGI workstation.
2. Change the bootmagic variable `TCP_SPACE_SIZE` on the Paragon from the default size of 4K to 64K. 64K is a better size when using X applications that transmit data through an Ethernet connection.
3. If you still experience performance problems after these changes, you can run the code directly on the Ethernet node, instead of on a compute node. To run on the Ethernet node, compile the code without the `-nx` option.

4040 IPD Unable to access bit field members of a struct

IPD cannot display or assign bitfield members of structures in C programs. Any attempt to access a bitfield results in the following error message:

```
*** ERROR: search failed *** Not found: <bitfield-name>
```

4790 IPD display -string only lists first element of array of strings

When displaying an array of pointers to strings with the command “display -string”, only the first string is displayed; also, the subscript value displayed is incorrect and any count specification is ignored. For example, if “str” is defined as follows:

```
char *str[] = { “now”, “is”, “the”, “time”, “for” };
```

The command “disp -string str,5” prints the response:

```
*** string5.c{ }main(str ** ***** (all:0) ***** str[-1] = “now”
```

4791 IPD Radix conversion for display is limited.

The “display” command’s “-hex”, “-dec”, and “-oct” switches (which control the radix of the displayed value) can currently only be used with integer types.

4885 IPD IPD may say load completed when the application failed to start up

When you load a program, IPD attempts to run the new processes to the first line of user code. Once each of the processes has stopped, IPD reports “load completed” without checking to ensure that the processes were created and initialized successfully. Therefore, it may be possible to get the “load completed” message even if not all of the processes were successfully loaded.

5589 IPD IPD has problems displaying FORTRAN assumed-size arrays.

In FORTRAN, it is possible to declare an array argument without specifying the last dimension, as in the following example:

```
program main integer ary(10,30) call sub( ary ) end
```

```
subroutine sub( a ) integer a(2,17,*) ... return end
```

If the preceding program is loaded under IPD and run to a breakpoint in sub(), you may have trouble displaying the array “a”. The problem is that there is not enough information available to IPD to determine the size of the array, and so IPD makes an assumption that the size of the missing dimension is 1. As a result, IPD displays a warning message and then services the display request as if the last array dimension was 1.

```
(all:0)> display a
```

```
*** WARNING: Out of range: Fortran arg: dimension info unavailable Assuming dimension size of 1 and lower bound of 1
```

```
** hello.f{ }sub(a ** ***** (all:0) ***** a(1,1,1) = 0 a(2,1,1) = 0 a(1,2,1) = 0 a(2,2,1) = 0 ... etc
```


The “type” command has a similar problem:

```
(all:0)> type a
```

```
*** WARNING: Out of range: Fortran arg: dimension info unavailable Assuming dimension size  
of 1 and lower bound of 1 ** hello.f{ }sub(a ** ***** (all:0) ***** INTEGER a( 2, 17, 1 )
```

There is a workaround for displaying assumed-size arrays. The comma-count can be used to display beyond what IPD assumes to be the end of the array. For example, the array above has 300 elements and can be displayed in full with the command “display a ,300”. IPD will still issue a warning but will display the full array.

5811 IPD Fortran alternate entries not supported

IPD does not currently support debugging of FORTRAN subprograms with alternate entry points.

5813 IPD Variables in registers not assigned correctly

If the program being debugged was compiled by “icc”, if a floating-point variable is in a register and you assign it a new value, the value is assigned on the stack rather than the register where the variable actually is.

5907 IPD 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”.

6375 IPD Default context not updated when process exits

When a process reaches the “Exited” state (not “Exiting”), it is not removed from the default context. If you try to do something with that context, you will see a “process not found” message for the process that exited.

6480 IPD IPD does not capture commands typed ahead while wait command active

If you run a program under IPD with the command “run; wait”, then type IPD commands ahead (before the prompt appears), when the “wait” command returns the prompt, the commands you typed ahead are ignored.

7070 IPD IPD fails if a process running in .compute calls nx_initve().

If a process that is already part of a parallel application and is running in the compute partition calls nx_initve(), IPD should simply stop tracing the process and let it go. Instead, lots of internal errors are printed. There is no workaround for this problem.

7440 IPD IPD gets lost and hangs nodes when nx app. makes nx_load() calls

When running an nx application under IPD, IPD gets lost if the application forks to other compute nodes with nx_load() calls. When you attempt to quit IPD in this state, the system will hang. There is no workaround for this problem.

7958 IPD 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.

8487 IPD "step ,count" fails if a system call is encountered while stepping

If you step with a count greater than 1 and attempt to step over a system call (like printf()), a warning is printed, the step is aborted, and you are left inside the system call routine. To avoid this problem, you execute steps individually (i.e., step, step, step). Or if you don't want to step into your own function calls, you can specify "step -call ,count".

8688 IPD Pulling of PM data fails in ETS 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, ETS and/or the application can appear to hang. This practice can also lead to incomplete data being collected. To collect complete performance data without hanging ETS and/or the application, instrument either the controlling process or the node processes but not both.

8924 IPD IPD doesn't handle large character strings gracefully

IPD may generate a fatal internal error when trying to display large character strings. There is no workaround for this problem other than to keep the size of your diagnostic strings to a modest length.

8996 IPD IPD and XIPD don't handle include files correctly

When using IPD or XIPD, you cannot display source code lines that were inserted in a program with an include() function. One ramification of this problem is that you are not able to set breakpoints in a section of code included with an include() function.

9100 IPD An application that calls the Unix fork() function will hang the IPD

If you use IPD to monitor an application program that calls the Unix function fork() (either directly or through the system() function), IPD will hang when fork() is executed. There is no workaround for this problem. IPD does not currently support debugging of applications that use Unix fork(). This problem can also occur when using functions like system(), such as getenv().

9241 IPD Rerunning an application with instr -paragraph can cause errors and core dump

If you run a parallel application using instr -paragraph and then rerun it again to obtain additional performance data, IPD may generate errors and a core dump. There is no workaround for this problem.

9407 IPD IPD gets a bus error if the instrument -force switch is abbreviated.**9011 LIBPM** ETM fails to merge data generated with “flush -stop” option.

The workaround for this problem is to rerun the code, but make the size of the instrumentation buffer bigger. The command in IPD to do that is “instrument -bufsize xxx”, where ‘xxx’ is in Kbytes. The size of the buffer depends on the code. The general rule would be that the buffer must be big enough to fit all of the program’s instrumentation data.

7773 LIBTGI Loading app. on .compute using default rectangle may cause allocator error msg.

When loading an application in a partition using the default rectangle, the allocator may send an error message: “autoinit nx_initve: Allocator internal error”. This message may mean that your default rectangle does not fit in the partition you have defined. To solve this problem, select a rectangle size that fits into the dimensions of the partition.

8078 PARAGRAPH Instr -paragraph in applications that call flick() causes error

Instrumenting for ParaGraph within an application that has calls to flick() causes an error when the application is run. This error occurs because when flick is instrumented, two trace records are generated, causing the trace to quickly grow huge. The only workaround for this problem is to comment out the flick() calls in the application program.

9201 PARAGRAPH 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.

6761 SPV SPV’s partition information is not dynamic.

If a partition is created or removed after SPV is invoked, SPV’s list of partitions (shown in the Partition Color Mappings dialog) does not change.

6795 SPV User is unable to zoom/pan or change options after data replay

When you replay a previously-recorded data file, after the data file has been replayed the “Options” and “Zoom” menus are grayed-out and cannot be used.

5362 XIPD XIPD code window does not properly indicate all nodes executing line

XIPD's code window normally puts a checkmark next to a line about to be executed. When you move the mouse pointer over this checkmark, a message is printed in the code window indicating which nodes are executing the line. For non-zero process types (ptypes), the executing nodes are sometimes not shown. To find out which nodes are executing which lines, select "Traceback" from the "Display" menu.

6313 XIPD Nodes not loaded properly when all nodes are applied in Load dialog

If you select ALL nodes with XIPD, select an executable to load, select APPLY, and then press OK, XIPD should load the nodes, once. Instead, it attempts to perform a load once for each time the APPLY button was pressed and then once more for pressing the OK button. XIPD does not try to perform any of these loads until the OK button is pressed. The net result of this operation is that XIPD tries to load the same program on the same set of nodes with the same ptype. Load error messages are then generated, stating that the ptype is already in use. There is no workaround for this problem.

7181 XIPD Filtering routines with "show routines with mon. pt." doesn't work properly

When you set monitor points in code, then attempt to filter to code to show only the code with monitor points ("show routines with monitor points"), no code is shown. There is no workaround for this problem.

7218 XIPD Pressing Reset button in Locate Source Code dialog can cause core dump

This problem results from pressing the Reset button on the mouse in rapid succession while in the Locate Source Code dialog. There is no workaround other than to be patient in this dialog.

7673 XIPD Reset in the Load dialog doesn't reset properly

The Reset option in the Load Application dialog doesn't operate properly. A loaded node will be incorrectly reset to unloaded after a reset. The only workaround for this problem is to cancel the dialog and bring it back up.

7674 XIPD Cancel in the Load Application dialog doesn't undo fields

Executing a Cancel in the Load application dialog will dismiss the dialog. However, if you invoke the Load application dialog again, the values that were being entered in the fields before the cancel are still there (without selecting Apply before cancel). This is incorrect behavior for XIPD. There is no workaround for this problem.

9084 XIPD XIPD doesn't note functions with long names as active

If a C function name is longer than 13 characters, XIPD is not be able to update its process tables correctly and the function will not show as being active (no green check mark is shown). The workaround for this problem is to display the "Traceback" dialog to find how where you are.

9331 XPROF XProf incorrectly parses the command line option -g as -geometry

If you invoke XProf with the -g option (for example, "xprof -g -z"), XProf should be able to view static or non-global (local) functions. Instead, XProf incorrectly parses -g as -geometry and returns the error message:

Warning: Shell widget "xprof" has an invalid geometry specification: "-z".

The workaround for this problem is to include the -g option at the end of the command line (for example, "xprof -z -g").

Bugs Fixed Since Release 1.1

7668 IPD Removing partition out from under IPD hangs logins

7911 IPD Instrumenting host AND node application results in internal error

8135 IPD Need to print warning if application uses hrecv/hsend

8140 IPD paragon hangs if ipd exits while its application is scheduled out

8621 IPD Functions should not be instrumented if they branch to first instruction.

8623 IPD IPD hangs on quit when controlling process already exited

8852 IPD ipd: internal error during "exec" of command file at "run"

8898 IPD ipd context command still displays "previous ptypes" which is not allowed now

8965 IPD BETA PCCM BUG under IPD, does not pass environmental variables to application

8981 IPD frame command often gives incorrect and misleading information

8986 IPD If a stopped process is signaled, IPD may resume it incorrectly.

9111 IPD BETA repetitive IPD message for hrecv fills screen losing information

9227 IPD IPD reports an error when a breakpoint is set after a fork

8029 LIBPM Setting "flush -wrap" policy in ipd causes application to hang.

8052 LIBPM Changes being made to hsend()/hrecv() thread will break ParaGraph instrumentation

8148 LIBPM Prof and Gprof vectoring code not thread safe (hsend/hrecv can break inst).

8510 LIBPM Error and Debug printf's in libpm thread may cause application to hang.

- 8327 LIBTGI** Can't stack up the partition via different ptypes in ParAide.
- 8329 LIBTGI** Can't stack up the partition via different ptypes in XIPD.
- 8785 LIBTGI** The new XIPD load command when input is redirect from a file has problem.
- 7930 PARAGRAPH** ParaGraph fails when run on certain X-Terminals
- 8409 PARAIDE** Need Help On XIPD-Lite from the Help menu of the main window.
- 8987 PARAIDE** ParAide should print an error message and exit if all -no options are used.
- 8552 PMAKE** Pmake hangs during VSE.build testing with SEGV
- 6883 SPV I/O** Nodes partition does not match BOOT_IO_NODE_LIST
- 9312 SPV SPV** writes 8 lines of bogus error messages when starting up
- 7474 XIPD** XIPD does not handle file redirection for application input
- 8695 XIPD** XIPD core dumps instead of error msg when instrument 2nd time w/o reloading app.
- 8833 XIPD** If 'Save perfmon data to' field is blanked out deliberately, XIPD send err msg.