

Scali System Guide

Copyright © 1999, 2002 Scali AS. All rights reserved.

Table of contents

Chapter 1 Introduction	11
1.1 Purpose of the Scali System Guide	11
1.2 Overview of the Scali System Guide	11
1.3 How to read this guide.....	12
1.4 Acronyms and abbreviations	12
1.5 Terms and conventions.....	13
1.6 Typographic conventions.....	13
Chapter 2 Scali system overview	15
2.1 Description of a Scali system	15
2.1.1 Hardware system architecture	16
2.1.2 Software system architecture	16
2.2 Scali system examples	17
2.2.1 Rackmount systems.....	17
2.2.2 Scali system cabinet	18
Chapter 3 Hardware installation	19
3.1 Hardware organization.....	19
3.1.1 Node naming scheme.....	19
3.1.2 Labelling	20
3.1.3 Checklist	20
3.2 Node BIOS preparation	20
3.2.1 Set MP level	20
3.2.2 Interrupt level mapping.....	20
3.2.3 Power on startup	20
3.2.4 Enable ECC.....	21
3.2.5 BIOS redirection to serial port	21
3.3 Ethernet setup	21
3.3.1 Using the site LAN	21
3.3.2 Using a private network.....	22
3.3.3 Using channel bonding.....	22
3.4 Dolphin SCI interconnect installation.....	23
3.4.1 SCI hardware overview	23
3.4.2 SCI adapter installation.....	24
3.4.2.1 64bit/66 MHz PCI bus operation	25
3.4.3 SCI Cabling.....	26
3.4.3.1 General rules	26

3.4.3.2 Single SCI ring cabling	27
3.4.3.3 Two-dimensional SCI torus cabling.....	28
3.4.3.4 Three-dimensional SCI torus cabling.....	30
3.5 Console switching.....	32
3.6 Power switching	33
Chapter 4 Software installation.....	35
4.1 Software installation overview.....	35
4.2 Operating system preparation.....	35
4.2.1 (M) Enable rsh root access from/to all hosts.....	36
4.2.2 (O) Enable rsh access from/to all hosts for common users	36
4.2.3 (O) Enable rlogin and login as root from front-end.....	36
4.2.4 (M) Ensure the 'at' service is working	36
4.2.5 (O) Set up console over the serial port.....	37
4.2.5.1 Configuring serial console on Linux	37
4.2.5.2 Configuring serial console on Solaris:	38
4.2.6 (M) Use a common file system for MPI applications.....	38
4.2.7 (O) Use NIS	38
4.2.7.1 Red Hat Linux NIS setup.....	38
4.2.7.2 SuSE Linux NIS setup	38
4.2.7.3 Solaris NIS setup.....	39
4.2.8 (M) Provide OpenGL for Scali Universe GUI.....	39
4.3 Scali software installation	39
4.3.1 SSP installation	40
4.3.1.1 SSP install program options	40
4.3.2 Installation explained step by step.....	41
4.3.2.1 Introduction	41
4.3.2.2 Installation Configuration	45
4.3.2.3 Software installation	54
4.3.2.4 Post installation fix	56
4.3.2.5 Functional testing.....	57
4.3.3 SSP Uninstall.....	59
4.3.4 Uninstallation example	59
Chapter 5 Scali Universe	61
5.1 Overview.....	61
5.2 Getting started with Scali Universe.....	62
5.2.1 Prerequisites	62
5.2.2 Installation.....	62
5.2.3 Running.....	63
5.2.4 Configuration of the Scali Universe GUI.....	63
5.2.5 Logging in to a system.....	66

5.3 Using Scali Universe	67
5.3.1 Node selection	67
5.3.2 MainWindow menu overview.....	68
5.3.3 Different system views - the View menu.....	69
5.3.4 Running programs - the Run menu.....	70
5.3.4.1 Working with runsets	70
5.3.4.2 Running ScaMPI MPI programs	71
5.3.4.3 Running MPICH programs.....	72
5.3.4.4 Running Parallel Shell commands.....	73
5.3.4.5 Node terminal sessions	73
5.3.4.6 Frontend terminal session.....	74
5.3.5 Management menu.....	74
5.3.6 Software installation - Software menu.....	75
5.3.6.1 Software Installation Window.....	76
Chapter 6 System Monitoring	77
6.1 Overview.....	77
6.1.1 Architecture.	77
6.2 Using monitoring from Scali Universe	78
6.2.1 The Status menu	78
6.3 Graphical monitoring.....	79
6.3.1 Common functionality	80
6.3.1.1 Pop-up menu.....	80
6.3.1.2 3D view camera controls.....	80
6.3.2 Compact View	81
6.3.3 Two dimensional 2D bar view.....	81
6.3.4 3D bar View	82
6.3.5 2D history view	82
6.3.6 3D History View	83
6.3.7 3D System View - compound view	83
6.4 Using Alarms	84
6.4.1 The Alarm window.	84
6.4.2 The Alarm Editor.	85
6.4.3 Alarm log viewer.....	87
6.4.4 Example: Defining a new alarm	87
6.5 The monitoring daemon: scamond	88
6.5.1 Manual start/stop	88
6.5.2 Configuration file.....	89
6.5.3 Default monitoring variables.....	89
6.6 The SNMP daemon: scasnmpd.....	90
6.6.1 Manual start/stop	91
6.7 Defining new monitoring variables.....	91

6.7.1	Obtaining the SNMP OIDs.....	91
6.7.2	Editing the ScaMon configuration file	92
6.7.2.1	Editing the class definition	92
6.7.2.2	Editing hwgroup classes.....	92
6.7.2.3	Adding the OIDs	93
6.7.2.4	Adding the variable definition for Universe	94
Chapter 7 Interconnect configuration		95
7.1	Overview.....	95
7.1.1	Architecture.....	95
7.2	SCI configuration from Scali Universe	96
7.2.1	The Interconnect menu.....	97
7.2.2	Link Status Window - interconnect monitoring.....	97
7.2.2.1	SCI link status graphics.....	98
7.2.2.2	SCI link status window options.	98
7.3	Command line interface - scaconftool	99
7.3.1	Starting scaconftool	99
7.3.2	Using scaconftool	100
7.3.2.1	User modes.....	100
7.3.2.2	Command truncation.....	100
7.3.2.3	Node selection	101
7.3.2.4	Disabling server log messages	101
7.3.2.5	Batch mode	102
7.3.2.6	On-line help	102
7.3.3	Status check of nodes and daemons.....	102
7.3.4	The fix command.....	104
7.3.5	Setting SCI nodeID manually	104
7.3.6	Reloading SCI-driver	105
7.3.7	Access to console on one node.....	105
7.3.8	Hot restart of the configuration server	105
7.3.9	Failing nodes	106
7.3.10	Daemon control with scaconftool	106
7.3.11	Setting server options.	106
7.4	Routing.....	107
7.4.1	Ring topology.....	107
7.4.2	2D Torus topology	107
7.4.2.1	Dimensional routing in 2D torus	108
7.4.2.2	Maxcy routing algorithm for 2D torus.....	108
7.4.3	3D torus topology	109
7.4.3.1	Dimensional routing in a 3D torus	109
7.4.3.2	C3 routing in a 3D torus	109
7.4.4	Routing partitions.....	109

7.4.5 Testing the routing.....	110
7.5 The configuration server daemon	110
7.6 The ScaConf node daemon	111
Chapter 8 Batch system ScaOPBS.....	113
8.1 Overview.....	113
8.2 ScaOPBS Installation	114
8.2.1 Installing OpenPBS during SSP installation.....	114
8.2.2 Installing ScaOPBS after SSP installation.....	115
8.2.3 Running the ScaOPBS installation tests manually.	115
8.3 OpenPBS configuration	115
8.3.1 Manual reconfiguration of ScaOPBS.....	115
8.3.2 Backup of local configuration files.....	116
8.3.3 The node property file	117
8.3.4 Server configuration - qmgr	117
8.3.4.1 Listing the current server configuration.....	117
8.3.4.2 Adding a queue.....	118
8.3.5 Removing all ScaOPBS configuration files.	118
8.3.6 ScaOPBS file locations	119
8.3.7 Using aliases for host names	119
8.3.8 PBS enforcement	119
8.3.9 Node definition using Virtual Processors.....	120
8.3.10 xpbs and xpbsmon	121
8.4 Example of ScaOPBS usage	121
8.5 Open PBS commands.....	122
8.6 scasub	122
8.7 Starting MPI jobs from within scripts submitted to OpenPBS.....	123
8.8 Using ScaOPBS from Scali Universe.....	124
8.8.1 Submitting jobs using Scali Universe	124
8.8.2 Monitoring jobs from Scali Universe	127
8.8.3 Enable and disable OpenPBS queue enforcement.....	128
8.9 Source code	128
8.10 Software License	129
Chapter 9 ScaSH - parallel shell tools	131
9.1 scash - the parallel shell command.....	131
9.2 scahosts - nodes availability check	132
9.3 scacp - local file copy	133
9.4 scarcp - remote file copy	133
9.5 scaps - process information	134
9.6 scakill - parallel kill command	135
9.7 scarup - node status	136

9.8 scawho - user information.....	137
9.9 ScaSH configuration file	137
Chapter 10 Troubleshooting	139
10.1 Hardware installation problems.....	139
10.2 Software installation problems.....	139
10.3 License problems	140
10.4 MPI problems	141
10.5 SCI Interconnect problems	141
10.5.1 SCI error messages	141
10.5.2 SCI Troubleshooting	143
10.5.3 SCI link errors.	144
10.5.3.1 sci_hwid.....	144
10.5.3.2 sci_hwtop.....	145
10.5.3.3 sci_hwdiag.....	146
10.6 Technical Support	149
10.6.1 Support Contracts.....	149
10.6.2 How to report the problem	149
10.6.3 Other Feedback.....	150
10.6.4 Keeping informed.....	150
Appendix A Scali software details.....	151
A-1 General information	151
A-1.1 Scali software platform CD-ROM.....	151
A-1.1.1 Installation for the impatient	151
A-1.1.2 Uninstall.....	152
A-1.2 Scali software directory structure	152
A-2 Scali package file naming convention	152
A-3 Scali software packages overview.....	153
A-4 Scali software daemon overview.....	155
A-5 Scali configuration file overview.....	156
Appendix B Scali Software Licensing.....	157
B-1 Introduction	157
B-2 Requesting licenses	157
B-2.1 Automated demo licenses generation from customer ID tag.....	157
B-2.2 Demo licenses - general requests	157
B-2.3 Permanent licenses	158
B-2.3.1 Permanent license request format example.....	158
B-2.3.2 Generating the request during installation	159
B-2.3.3 Generating a new license request.....	159
B-2.3.4 Last resort	159

B-3 The license file.....	159
B-3.1 Default location.....	159
B-3.2 SCALM_LICENSE_FILE environment variable.....	160
B-3.3 License file example.....	160
B-3.4 Adding or updating features (licenses).....	160
B-4 License installation.....	161
B-4.1 License update/installation with SSPinstall.....	161
B-4.2 Manual license update/installation.....	161
B-5 Meta-licensing.....	162
B-6 Troubleshooting.....	162
B-6.1 Error: Invalid FEATURE line.....	162
B-6.2 Error: Invalid version x.x > y.y.....	162
B-6.3 Error: FEATURE has expired.....	162
B-6.4 Error: Host ID is not found!.....	162
B-6.5 Missing license file.....	163
B-6.6 Missing license software - ScaLM package.....	163
B-7 Older versions with FLEXlm.....	163
Appendix C SCI Utility Programs.....	165
C-1 SCI hardware status programs.....	165
C-1.1 sciping.....	165
C-1.2 scimonitor.....	166
C-1.3 sciemsg.....	166
C-1.4 scidbx.....	166
C-1.5 scideb.....	166
C-1.6 scinode.....	167
C-1.7 scinfo.....	168
C-1.8 scireconf.....	168
C-1.9 scireload.....	168
C-1.10 scidle.....	169
C-1.11 scicards.....	169
Appendix D ScaConf Reference.....	171
D-1 Installation and package dependencies.....	171
D-2 scaconftool - command reference.....	171
D-2.1 ConfTool> console.....	172
D-2.2 ConfTool> daemon.....	172
D-2.3 ConfTool> fix.....	172
D-2.4 ConfTool> getopt.....	172
D-2.5 ConfTool> setopt.....	173
D-2.6 ConfTool> info.....	174
D-2.7 ConfTool> select.....	174

D-2.8 ConfTool> unselect.....	174
D-2.9 ConfTool> list.....	174
D-2.10 ConfTool> fail.....	176
D-2.11 ConfTool> reconnect	176
D-2.12 ConfTool> log.....	177
D-2.13 ConfTool> nodeid	177
D-2.14 ConfTool> reload	177
D-2.15 ConfTool> reroute	177
D-2.16 ConfTool> status.....	178
D-2.17 ConfTool> link.....	179
D-2.18 ConfTool> update.....	179
D-2.19 ConfTool> restart.....	179
D-2.20 ConfTool> sciping.....	179
D-2.21 ConfTool> help	179
D-2.22 ConfTool> quit.....	180
D-2.23 ConfTool> version	180
Appendix E ScaPkg - Scali Software installation program.....	181
E-1 Using scapkg.....	181
E-2 Configuration.....	182
E-2.1 ScaPkg.conf - path to package files (repository).....	182
E-2.2 ScaPkg.conf - package list	183
E-2.3 ScaPkg.conf - dependency list.....	183
E-2.4 ScaPkg.conf - Super-categories.....	184
E-2.5 ScaPkg.conf - Node list	184
E-2.6 Package response files	185
Appendix F Related Documentation.....	187
F-1 References	187

1.1 Purpose of the Scali System Guide

The intention of the Scali System Guide is:

- to provide an overview of a Scali System.
- to provide instructions for building a Scali System with respect to hardware and software installation and configuration.
- to provide instructions on how to use and manage a Scali System through the Scali Universe cluster management system.

Scali deliver its products:

- as turnkey supercomputer cluster systems: TeraRacks
- as interconnect and management systems to OEMs and VARs.
- as “build your own supercomputer” kits (WulfKits).

Consequently, the wrapping and contents of your Scali products may vary but it is still called a Scali system throughout this document.

1.2 Overview of the Scali System Guide

The Scali System Guide is organised as follows:

- Chapter 2 presents an **overview** of Scali systems and software.
- Chapter 3 describes the **hardware** installation procedure including the high speed interconnect adapters.
- Chapter 4 describes the **software** installation procedure including the operating system configuration and the Scali Software Platform.
- Chapter 5 describes the **Scali Universe** graphical system management framework and how to install, configure and start using it.
- Chapter 6 describes the use and configuration of the Scali **monitoring system** in detail with graphical monitoring, alarms and user defined variables.
- Chapter 7 gives a detailed description of Scali **interconnect configuration** system with examples of use from Scali Universe and the scaconftool ASCII based configuration client.
- Chapter 8) describes the integration, configuration and use of the **OpenPBS** queue system with Scali systems
- Chapter 9 describes the Scali parallel shell tool suite, starting with the fundamental **scash** and then moving on to the more specialised tools like **scaps**.

Chapter 1: Introduction

Chapter 10 describes how to **troubleshoot** common problems using diagnostic tools and how to get assistance from Scali.

Appendix A collects useful meta information about the Scali software platform, like contents of the distribution media, daemon overview and configuration file overview.

Appendix B is a detailed description of the Scali licensing system ScaLM.

Appendix C is a detailed reference to the SCI interconnect utility programs.

Appendix D is a reference to the ScaConf interconnect configuration system.

Appendix E is a detailed reference to **scapkg**, the Scali software installation program

Appendix F provides a list of **related documentation** you may consult for additional information.

1.3 How to read this guide

This guide is written for skilled computer users and professionals. It is assumed that the reader is familiar with the basic concepts and terminology of computer hardware and software since none of these will be explained in any detail. Depending on your user profile, some chapters are more relevant than others. We recommend that:

- System installators should read: Chapters 2, 3, 4, 8, 10 and B
- System administrators should read: everything!
- Ordinary users should read: Chapters 2, 5, 6, 7 and 9.

1.4 Acronyms and abbreviations

Abbreviation	Meaning
HPC	High Performance Computer
NIC	Network Interface Card
MPI	Message Passing Interface
OEM	Original Equipment Manufacturer
SCI	Scalable Coherent Interface
SSP	Scali Software Platform is the name of all Scali software packages.
VAR	Value Added Reseller

Table 1-1: Acronyms and abbreviations

1.5 Terms and conventions

Unless explicitly specified otherwise, `gcc` (gnu c-compiler) and `bash` (gnu Bourne-Again-SHell) are used in all examples.

Term	Description.
Node	A single computer in an interconnected system consisting of more than one computer
Cluster	A cluster is a set of interconnected nodes with the aim to act as one single unit
Scali system	A cluster consisting of Scali components
Frontend	A computer outside the cluster nodes dedicated to run configuration, monitoring and licensing software
MPI process	Instance of application program with unique rank within <code>MPI_COMM_WORLD</code>
UNIX	Refers to all UNIX and lookalike OSes supported by the SSP, i.e. Solaris and Linux.
Windows	Refers to Microsoft Windows 98/Me/NT/2000

Table 1-2: Basic terms

1.6 Typographic conventions

Term	Description.
Bold	Program names, options and default values
<i>Italics</i>	User input
<code>mono spaced</code>	Computer related: Shell commands, examples, environment variables, file locations(directories) and contents
GUI style font	Refers to Menu, Button, checkbox or other items of a GUI
#	Command prompt in shell with super user privileges
%	Command prompt in shell with normal user privileges

Table 1-3: Typographic conventions

Chapter 1: Introduction

Chapter 2 Scali system overview

This chapter gives an overview of the different components in Scali systems and software.

2.1 Description of a Scali system

A Scali system can be described with a simple equation:

$$\begin{aligned} & \text{standard qualified nodes} \\ + & \text{ high speed interconnect} \\ + & \text{ Scali's Software Platform} \\ = & \text{ Affordable Supercomputer} \end{aligned}$$

Since a Scali system is built using high volume, low cost “off the shelf” nodes and a very powerful interconnect, the price vs. performance is extremely good. Scali systems are available in the range from small two node systems to large HPC systems with hundreds of nodes.

The Scali software integrates the **cluster** (nodes + interconnect) to a single system (parallel computer). All of the different Scali software components together constitutes the **Scali Software Platform (SSP)**.

The Scali Software Platform (SSP) may be divided in 3 domains of usage:

- **Installation**
Scali software installation, configuration and testing (SSPinstall)
- **Administration/Maintenance**
Parallel software installation tool (ScaPkg)
Configuration/management of the SCI interconnect (ScaConf)
Remote power switching (ScaPowd)
Console management and broadcasting (ScaCons)
Running OS commands in parallel on the cluster (ScaSH)
System diagnostics (ScaDiag)
- **Operational use**
Running OS commands in parallel on the cluster (ScaSH)
Running MPI applications (ScaMPI, MPICH)
Monitoring System performance and health (ScaMon)
Managing batch queue systems (ScaOPBS)

2.1.1 Hardware system architecture

A Scali system is usually set up with a dedicated frontend (sometimes called master) node. The frontend is used for compiling, monitoring, application launching, and other management tasks. If the system is set up with a private network, the frontend acts as a gateway between the private network and the local LAN (figure 2-1). For small clusters (4 to 8 nodes) the frontend node will often double as processing node.

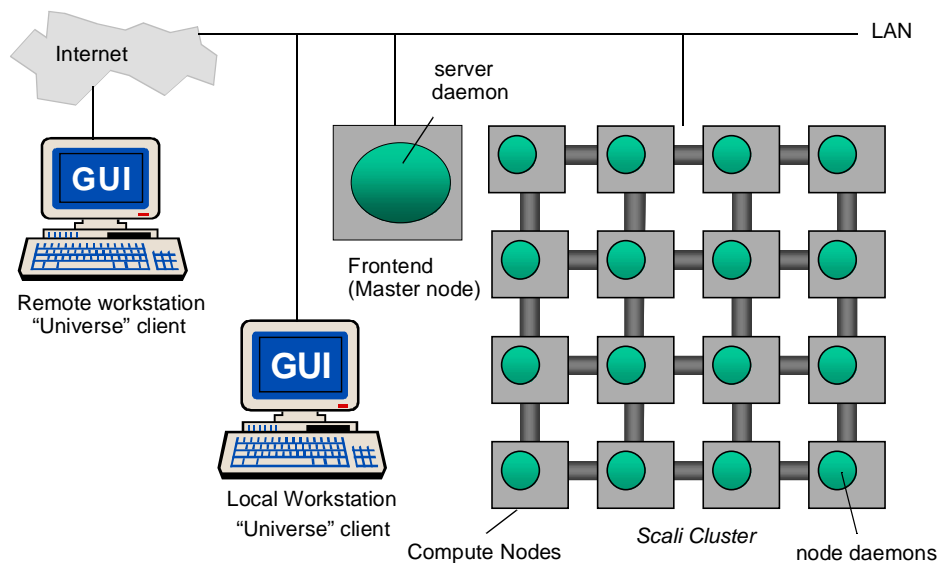


Figure 2-1: System architecture of a 4x4 Scali system

2.1.2 Software system architecture

Scali cluster software systems often follows the same architecture as the hardware with the frontend running a master process (daemon) or main process, while the compute nodes all running the similar node-processes. The main process is the "server" for Scali Universe clients. Examples are the Scali monitoring daemon: **scamond** or the interconnect configuration server daemon: **scaconfsd**. The node processes (daemons) and are typically "lighter" than the associated server process. One examples of node processes is the Scali SNMP daemon: **scasnmppd**. Communication between modules of control software is over the Ethernet on the local LAN. Management and use of the system through Scali Universe can be done from any workstation on the local LAN. Through Scali Secure Socket Communication (SSSC) clusters may even be completely managed over the Internet using Scali Universe clients from a remote workstation.

2.2 Scali system examples

2.2.1 Rackmount systems

Powerful rackmount computing nodes and easy integration of infrastructure hardware has made 19" racks the favoured solution for professional cluster based computers. Scali's turn-key systems can be tailored towards customer requirements (fig 2-2). Scali systems from our OEM partners has a different visual appearance (fig 2-3).



Figure 2-2: Scali system cabinet examples: Standard half-size Terarack (left) and a ruggedized version for portable supercomputing (right)



Figure 2-3: OEM partner packaging: Fujitsu- Siemens hpcLine (left) and Dell (right)

2.2.2 Scali system cabinet

For systems built from mini/miditowers, or other non-rack mountable cases the Scali system cabinet provides a practical solution for organizing hardware and infrastructure. Each node is put in a stackable enclosure with interconnect cable guides on the rear (figure 2-5). A LED panel on the front door of the enclosure may be configured to show a monitoring parameter like the CPUs activity..



Figure 2-4: A 4x8 Scali system boxed in Scali cabinets



Figure 2-5: Open front and rear side of a Scali system in a Scali cabinet

Chapter 3

Hardware installation

Aimed at OEMs, VARs and kit-builders this chapter explains how to organize and configure your node hardware, and how to install and connect both Ethernet and SCI interconnects to prepare a cluster system for SSP installation. The main focus is on the high speed interconnect installation and cabling since this has proved to be the most challenging part. If you own a turn-key Scali system (TeraRack) this chapter can be viewed as supplementary information.

3.1 Hardware organization

The most practical way to organize a cluster of interconnected computers is to stack them in a suitable system cabinet or rack. With large numbers of nodes a 2-dimensional (2D) arrangement is usually required. For rackmount nodes any 19" rack of reasonable quality could be used, just make sure airflow/cooling is sufficient. For nodes with deskside/mini tower cabinets Scali recommends using a Scali system cabinet or another boxed solution known to serve the requirements of cluster hardware management.

3.1.1 Node naming scheme

We recommend that you select a naming scheme where the node names are constructed from a base name followed by a numeric part. Ideally, the numeric part should reflect the physical location in the cabinet, i.e. node-XY where X and Y reflects the X and Y position respectively (figure 3-1). A naming scheme like this has a lot of benefits, especially on large systems, when operating the system from command line tools and when mapping between names and physical node locations.

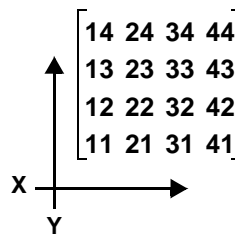


Figure 3-1: A 4x4 cluster with XY coordinates: physical view

3.1.2 Labelling

When node names have been decided, you should add labels with the node-name on the *back side* of the cabinet of all nodes. This is where you will be standing when inserting cables later on, so being able to tell which nodes you are working on makes things a lot easier.

3.1.3 Checklist

Some issues to remember when installing a cluster:

- The cluster must be placed in a location with sufficient cooling. Also be sure not to obstruct air ventilation channels on nodes and infrastructure equipment.
- Be extremely aware of cooling requirements for rack-mount systems since nodes are packed so close together. Fully enclosed 19" racks must not be used, remove front/back doors if necessary.
- The power outlets must be capable of supporting the cluster's accumulated power consumption.
- Provide easy access to all equipment in the entire cluster for maintenance.
- A cluster has a lot of interconnect cabling; try to keep the cables as short as possible.

3.2 Node BIOS preparation

In order to prepare the nodes for cluster use you will have to check a number of BIOS settings as described in the following sections.

3.2.1 Set MP level

In a multiprocessor node the BIOS sometimes has an option of configuring the so called 'MP level'. This specifies the protocol version used by the CPUs to interoperate. Possible values are usually 1.1 and 1.4. We recommend that you set this option to **1.4**.

3.2.2 Interrupt level mapping

Some BIOSes have the option of enabling interrupt level mapping. If possible turn this feature **on** to reduce the possibility of conflicts with other devices.

3.2.3 Power on startup

If your system includes external power switches it is important that you configure the power section in the BIOS so that the node is automatically started when power is applied to the node.

3.2.4 Enable ECC

Make sure ECC is **enabled** in BIOS - this is not always the default setting. The use of memory with ECC capabilities is essential when working with systems with tens to hundreds of gigabytes of memory.

3.2.5 BIOS redirection to serial port

Some BIOSes now provides the possibility to redirect BIOS output to the serial port. If your system has a serial console switch, and your BIOS supports redirection, we recommend that you **enable** this since it will allow for centralized and efficient BIOS setup and inspection.

3.3 Ethernet setup

Scali systems requires Ethernet access between the frontend and all nodes in the system. The Ethernet is used for control and monitoring - even for systems with a high-performance interconnect installed. Generally you have two choices for setting up the Ethernet, either to include all nodes in the local LAN, or to provide a private network for your Scali system.

3.3.1 Using the site LAN

This is the simplest and least secure option. There's not really a lot to consider here. Just connect all nodes to a HUB or preferably a switch in your local network.

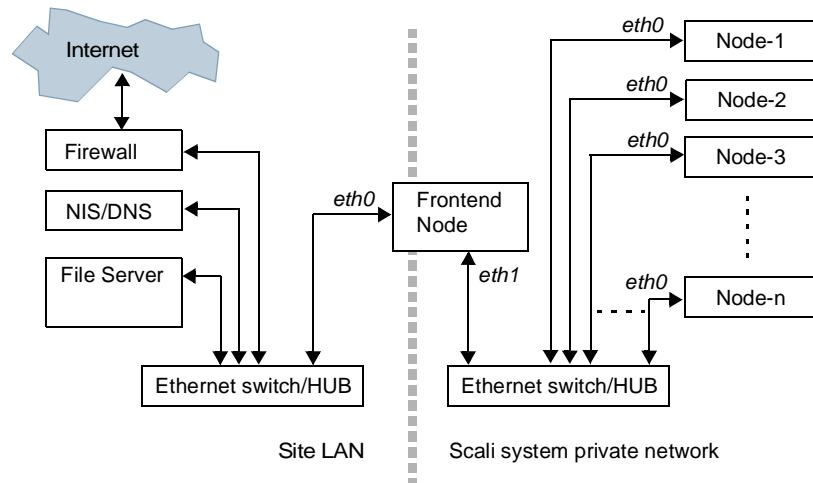


Figure 3-2: Using private network for the Scali System

3.3.2 Using a private network

Using a private LAN for your system provides better security and performance. With this solution the frontend is set up as the gateway between the local LAN and the processing nodes. The frontend therefore needs one more Ethernet port/adapter than the compute nodes. Scali recommends that the primary Ethernet port/adapter (giving eth0) on the frontend is used for the connection to the local LAN while the second Ethernet port/adapter (giving eth1) is connected to the Ethernet switch/HUB of the private network. For all processing nodes, the primary Ethernet port/adapter (eth0) should be used. Please refer to figure 3-2 for a diagram.

3.3.3 Using channel bonding

Channel bonding with fast Ethernet adapters is a low cost solution to increase the throughput of systems which lacks a high performance interconnect like SCI. For the moderate cost of one additional Ethernet adapter per node and one additional Ethernet switch, you may double the bandwidth of the Ethernet (latency however will remain unchanged). Some cluster-enabled nodes even comes with dual Ethernet controllers on-board which makes this option very attractive. Note that the use of channel bonding implies the use of a private network. Figure 3-3 shows one cabling diagram based on the recommendations from 3.3.2.

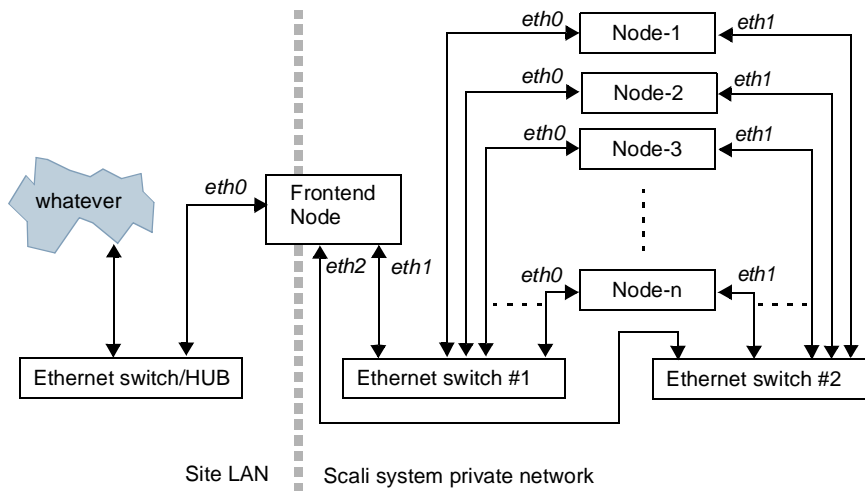


Figure 3-3: Scali system with channel bonding

3.4 Dolphin SCI interconnect installation

SCI is an international standard (IEEE1596) of a high speed interconnect. Dolphin Interconnect Solutions makes a number of SCI products among which their PCI-SCI adapter cards are used by Scali. The performance of these cards is amazing: Several hundreds of megabytes per second and sub microsecond latency on the SCI link layer with PCI performance approaching the limits of the bus.

3.4.1 SCI hardware overview

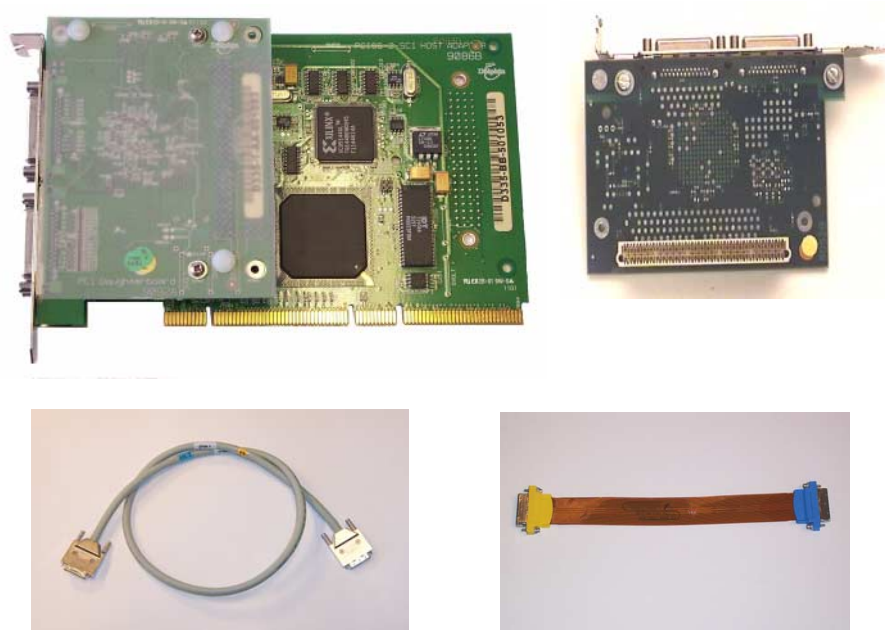


Figure 3-4: SCI hardware from Dolphin: PCI adapter, daughter card, standard cable (left) and flexi cable (right)

There are currently two generations of PCI-SCI adapter in use. The old D31x series and the newer D33x series. The main difference is that the new cards supports 64bit/66Mhz PCI and 3D SCI topologies. Needless to say, all new Scali systems with SCI are equipped with D33Xcards, but the SSP still supports D31x series.

Chapter 3: Hardware installation

Depending on model, an SCI card supports up to three SCI links, which we refer to as L0, L1 and L2. On newer boards the connectors are clearly marked, if not you can safely assume that connector numbering starts with the connector pair closest to the mainboard PCB being L0 (ref 3-5).

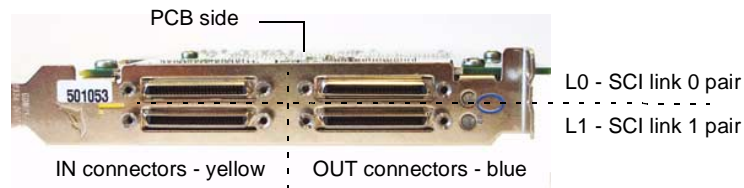


Figure 3-5: Identifying L0 and L1 connector pairs on SCI cards (D335)

3.4.2 SCI adapter installation

For older D31x cards, please make sure that all 4 jumpers are installed as indicated by figure 3-6 before inserting the adapter card. D33x series cards are jumperless..

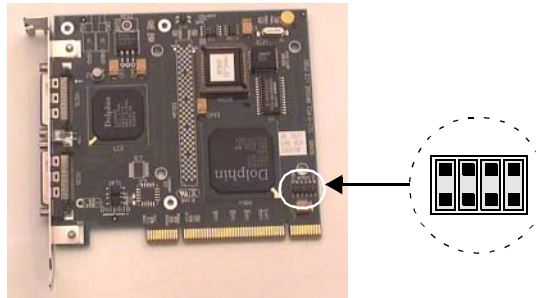


Figure 3-6: D31x SCI adapter card jumpers

If you have purchased a multi-dimensional SCI network using an SCI daughter card for the second (D311/D312), or third (D337) set of SCI links, please make sure the daughter card is securely mounted to the main SCI card *before* the “combo” is inserted into the node.

3.4 Dolphin SCI interconnect installation

Then, insert the SCI adapter card (with daughter card if applicable) into a free PCI slot in the node. If there are ISA slots present you should, if possible, avoid the PCI slots sharing interrupt lines with ISA slots. Be sure to fix the board properly in the slot for mechanical stability. Repeat the procedure on all nodes in the cluster.

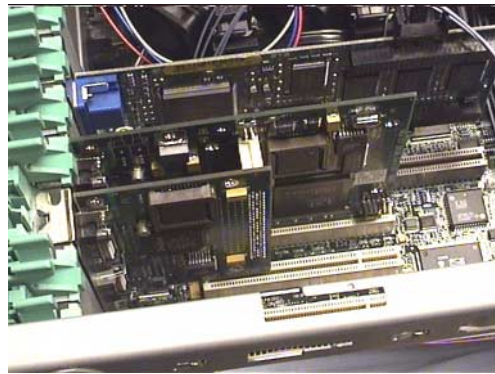


Figure 3-7: Node with SCI adapter card and daughter card inserted

Note: On older D31x cards, the LEDs on the back of the SCI board will be red/yellow until the proper software has been loaded - this is quite normal. When the software is loaded and the SCI link is OK the LEDs will turn to green. On newer D33x cars the LEDs will turn to green as soon as the SCI link is OK, even if no software has been loaded. For more details on how to interpret the SCI card LEDs please refer to “SCI Troubleshooting” on page 143.

3.4.2.1 64bit/66 MHz PCI bus operation

With the new D33x cards you can get tremendous speed-up from the faster 64/66 PCI bus operation, a few things to remember here are:

1. Make sure your SCI card does not share the PCI bus with a “slow” device. This will force the entire PCI bus to operate at the lower speed.
2. If used, make sure the “raiser-card” supports full 64/66 operation. For 1U and 2U rack-mount cabinets it is common to use a “raiser-card” to enable one or two full size PCI cards to be mounted horizontally.

3.4.3 SCI Cabling

The number of nodes in the system and the available SCI hardware determines your options for selecting an SCI topology. Our recommendations can be found in table 3-1. Cabling suggestions for the different topologies will be described in the following sections. With Scali turn-key systems you can find a detailed description of the SCI topology the interconnect section of the *“Scali System Configuration Description”*.

Topology	#nodes	links/node	Supported SCI cards
Ring	2 - 8	1	Any D31x or D33x
2D Torus	4 - 128	2	D311/D312, D315, D316, D335, D336, D337, D339
3D Torus	64 - 256	3	D336, D337

Table 3-1: Topology vs. SCI hardware compatibility

3.4.3.1 General rules

Interconnect cabling of simple topologies are somewhat complicated by system cabinets with another size and shape than the interconnect topology. The need to avoid long SCI cables adds another level of complexity to the interconnect cabling pattern. Some general rules apply though, when connecting the cables:

- Always connect the SCI cables from the OUT connector to the IN connector. Cables are colour coded to help you getting it right.
- The basic component in any SCI network is a ring. Scali system rings must always be connected between the same SCI link on all nodes, i.e. L0 outputs to L0 inputs and L1 outputs to L1 inputs.
- Interleave cabling to avoid long cables (see figure 3-8).
- Keep the number of nodes on the SCI rings in each dimension of the torus as even as possible. Many short SCI rings is better than a few long ones. A 16 node system should therefore be connected as 4x4, *not* as 8x2

3.4 Dolphin SCI interconnect installation

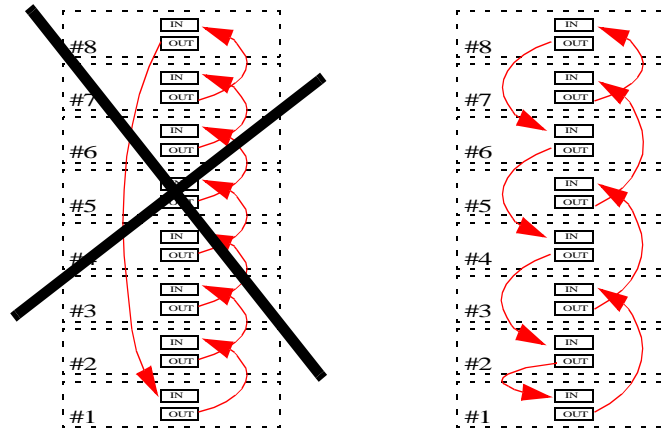


Figure 3-8: Avoid long cables by interleaving interconnect in each ring

3.4.3.2 Single SCI ring cabling

In small systems a single ring can be efficient. Figure 3-9 shows how to connect an 8 node SCI ring based on the table below:

From (OUT)	To (IN)
Node #1	Node #3
Node #3	Node #5
Node #5	Node #7
Node #7	Node #8
Node #8	Node #6
Node #6	Node #4
Node #4	Node #2
Node #2	Node #1

Table 3-2: Connection list example for cabling an 8 node SCI ring

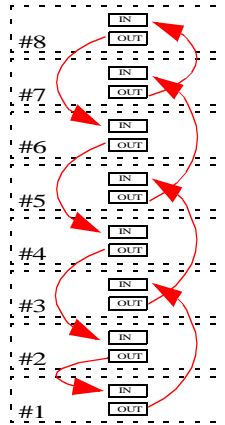


Figure 3-9: Example of 1x8 single ring cabling in a 1x8 cabinet

3.4.3.3 Two-dimensional SCI torus cabling

The 2D torus topology consists of rings in two dimensions, hence we use the same interleaving technique as for a single ring, but this time we have multiple rings in two dimensions which makes things slightly more complex. For 2D torus topologies we use the following additional guidelines:

- Connect the first dimension on SCI link 0: L0
- Connect the second dimension on SCI link 1: L1

Following this connection scheme enables you to use the default suggestions for connections when installing the SSP software later on.

From L0 (OUT)	To L0 (IN)
Node #x1	Node #x3
Node #x3	Node #x4
Node #x4	Node #x2
Node #x2	Node #x1

Table 3-3: Example cabling of one vertical ring in a 4x4 system (x=[1-4])

3.4 Dolphin SCI interconnect installation

From L1 (OUT)	To L1 (IN)
Node #1y	Node #3y
Node #3y	Node #4y
Node #4y	Node #2y
Node #2y	Node #1y

Table 3-4: Example cabling of one horizontal ring in a 4x4 system (y=[1-4])

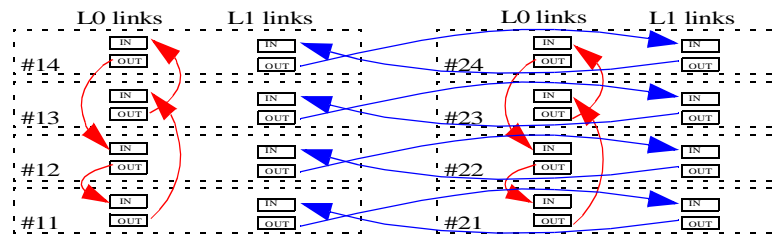


Figure 3-10: Example of 2x4 two-dimensional system cabling in a 2x4 cabinet

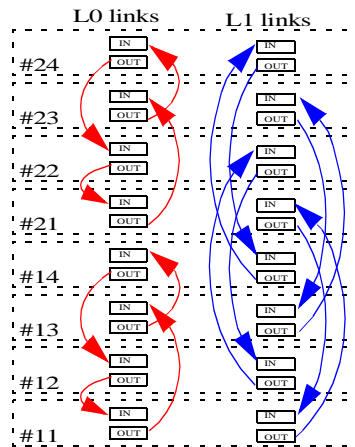


Figure 3-11: Example of 2x4 two-dimensional system cabling in a 1x8 cabinet

3.4.3.4 Three-dimensional SCI torus cabling

With the 3D (three-dimensional) SCI torus topology cabling becomes more of a challenge. The 3D topology is harder to merge with the natural row-column way of arranging node hardware. More effort is needed for optimal node placement and it can be hard to avoid long cables. Cabling of 3D systems follows the guidelines set up for rings and 2D torus topologies with one addition:

- Connect the third dimension on SCI link 2: L2

In the example below we will explain how to connect a small 18 node 3x3x2 3D torus. The structure of the torus is shown in Figure 3-12 (for readability the SCI network is drawn without “rings” and node names are only represented by the numeric part only).

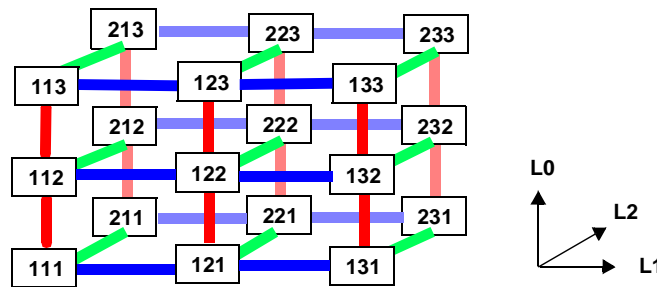


Figure 3-12: A 3x3x2 3D SCI torus

From this figure we can produce some general connection tables (3-5 through 3-7) for each of the SCI links. The resulting cabling on a 2x9 physical node arrangement can be inspected in figure 3-13. The cabling scheme works equally well for a 1x18 node rack, but will probably require somewhat longer cables on the L2 links. This illustrates how node placement becomes more important with 3D systems.

From L0 (OUT)	To L0 (IN)
node-zx1	node-zx3
node-zx3	node-zx2
node-zx2	node-zx1

Table 3-5: Example cabling of L0 on a 3x3x2 torus (z=[1,2], x=[1,2,3])

3.4 Dolphin SCI interconnect installation

From L1 (OUT)	To L1 (IN)
node-z1y	node-z3y
node-z3y	node-z2y
node-z2y	node-z1y

Table 3-6: Example cabling of L1 on a 3x3x2 torus (z=[1,2], y=[1,2,3])

From L2 (OUT)	To L2 (IN)
node-1xy	node-2xy
node-2xy	node-1xy

Table 3-7: Example cabling of L2 on a 3x3x2 torus (x=[1,2,3], y=[1,2,3])

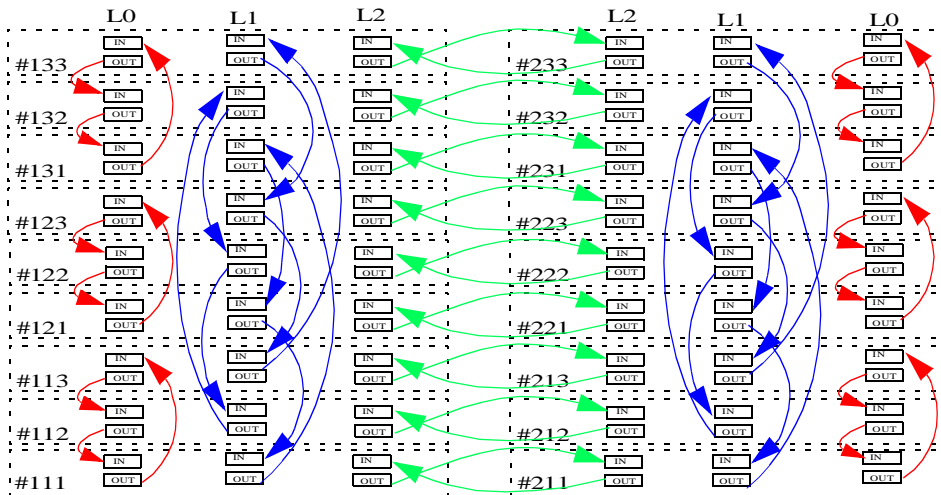


Figure 3-13: Example cabling of a 3x3x2 3D torus, please note that SCI links have been drawn mirrored for clarity.

Universe XE 3.5 Console switching

The advantages of centralized and remote access to node consoles are obvious when performing tasks like boot monitoring, BIOS configuration, OS installation and so on. The SSP products UniverseXE and ClusterEdge both supports remote console switching and console broadcasting provided that you:

- Use a qualified remote serial switch (see SSP release notes).
- Enable console redirection to the serial port in the nodes.

Cabling of a remote serial switch is quite simple: Cables must be connected from the ports of the serial switch to the serial port on each node - possibly by the aid of a RJ-45 to serial port adapter. If you connect the nodes “sequentially” you should also get sensible default values when configuring the console server during SSP installation later on. What we mean by sequential connection is best illustrated by an example: Say that you have 3 x 8-port serial switches and 20 nodes named node-1 through node-20. If nodes are defined in this sequence during SSP installation this is the order in which the console server expects them to be connected to the switches. You should therefore:

- Connect nodes 1 through 8 to switch #1: ports 1 through 8.
- Connect nodes 9 through 16 to switch #2: ports 1 through 8.
- Connect nodes 17 through 20 to switch #3: ports 1 through 4.

If you decide to use another connection scheme, this presents no problem for the system, but you will spend more time with the console server configuration without the benefit of “intelligent” default values.

Note: If you use a private network with your system we strongly recommend that the serial switches are connected to the *public* network. If not you may have severe problems reaching the consoles of your nodes if the frontend/gateway goes down.

With Scali turn-key systems (TeraRack) console switching is preconfigured. A description can be found in the “Scali Configuration Description” delivery document.

Universe XE 3.6 Power switching

Like with console switching, centralized and remote power switching is an obvious advantage for cluster systems management. The SSP products UniverseXE and ClusterEdge both supports remote console switching and console broadcasting provided that you use one of the qualified remote power switches. Please refer to the SSP release notes for an updated list of supported hardware.

Cabling of the remote power switches is really simple: Power cords must be connected between the power switch power outlets and the nodes' power input. If you follow a simple "sequential" connection scheme you should be able to get sensible default values also when configuring the power server during SSP installation later on. An explanation of what we mean by "sequential connection scheme" is described in "3.5 Console switching".

Just like with the console switches, another connection scheme will not present any problem for the system, but you will spend more time with power server configuration without the benefit of "intelligent" default values.

Note: If you use a private network with your system we strongly recommend that the power switches are connected to the *public* network. If not you will have severe problems using the power switches if the frontend/gateway goes down.

With Scali turn-key systems (TeraRack) remote power switching is preconfigured. A description can be found in the "Scali Configuration Description" delivery document.

Chapter 3: Hardware installation

Chapter 4

Software installation

Aimed at OEMs, VARs and kit-builders this chapter explains how to set up the OS and install Scali software on the cluster. If you own a turn-key Scali system (TeraRack) this chapter can be viewed as supplementary information.

4.1 Software installation overview

Before software installation is attempted, the cluster and interconnect hardware must be installed according to the guidelines in Chapter 3. The software installation can then be divided into two distinct steps:

- **OS installation and configuration:** Install and configure the operating system according to the guidelines outlined later in this chapter.
- **Scali software installation:** Install the Scali software with the SSP (Scali Software Platform) installation program: **install**. The installation program installs selected Scali software products on the entire cluster. Installation includes configuration and testing of the installed hardware and software on different levels.

Scali software and instructions can be found on the SSP CD-ROM or downloaded from:

<http://www.scali.com/download>

4.2 Operating system preparation

Before the SSP can be installed, each node in the cluster needs a properly configured installation of a supported operating system. Supported operating systems and versions are clearly listed in the SSP release notes, which can be found in the `NEWS` file in the SSP distribution, or on the Scali web-site.

Please also read the `/doc/os` file included in your SSP distribution (CD-ROM /download). This file may contain last minute information that did not make it into the manual.

The following sections contains both general and platform specific operating system configuration settings which again may be either mandatory (M) or optional (O). Scali recommends that you apply *all* the suggested configuration settings for your platform. You *must* however apply all the mandatory '(M)' ones.

4.2.1 (M) Enable rsh root access from/to all hosts

From all hosts/nodes, including frontend, have root rsh access to all other hosts/nodes, including frontend. Please do the following steps on every node:

- Insert frontend name and all nodenames into the `$HOME/.rhosts` file as root.
- Enable rsh commands as root from the frontend to all nodes.

RH 7.x: Add rsh to the file `/etc/securetty`. Enable the rsh service by doing:

```
# chkconfig rsh on
```

4.2.2 (O) Enable rsh access from/to all hosts for common users

Enable rsh access to all hosts/nodes for all non-root users.

Please do on all nodes, including frontend:

- Add all node names, including frontend name, in the file `/etc/hosts.equiv`.

4.2.3 (O) Enable rlogin and login as root from front-end

This step is optional, but will ease administration of the nodes. Note that this will have implications on the access security to the nodes in the cluster. If access security is an important issue, we will suggest to separate the cluster from the rest of the network by a firewall or by using the frontend as a gateway to the cluster (two NICs).

On Linux, enable rlogin as root from the frontend to all nodes.

RH 6.x:

Edit `/etc/pam.d/rlogin`, remove the `pam_securetty` check or add hosts to pam authentication.

RH 7.x:

Add rlogin to the file `/etc/securetty`. Enable the rlogin service by doing:
`chkconfig rlogin on`

SuSE:

Run `yast`, choose "System administration | Change configuration file" and set the entry `ROOT_LOGIN_REMOTE` to yes.

On Linux, enable login as root using telnet and enable login as root over serial console.

RH 6.x:

Edit `/etc/pam.d/login`, remove the `pam_securetty` check or add hosts to pam authentication.

RH 7.x:

Add login to the file `/etc/securetty`.

SuSE:

Run `yast`, choose "System administration | Change configuration file" and set the entry `ROOT_LOGIN_REMOTE` to yes.

4.2.4 (M) Ensure the 'at' service is working

You must have a working 'at' daemon (atd) running on all nodes. Verify that it works:

4.2 Operating system preparation

```
# echo touch /tmp/atfile | at now
# test -r /tmp/atfile && echo OK; test ! -r /tmp/atfile && echo ERROR
```

If not please do the following steps:

Linux:

Check that package is installed:

```
# rpm -q at
```

Ensure that the at daemon will be started at boot:

RH:

```
# chkconfig atd on
```

SuSE:

Run `yast`, choose "System administration | Change configuration file" and set the entry `START_ATD` to yes.

Check if the daemon is running:

RH:

```
# /etc/rc.d/init.d/atd status
```

Start the daemon if not:

```
# /etc/rc.d/init.d/atd start
```

SuSE:

```
# ps aux | grep atd | grep -v grep
```

Start the daemon if not.

Solaris:

Ensure that the at cron daemon will be started at boot:

```
# test ! -r /etc/rc2.d/S75cron && ln -s /etc/init.d/cron
/etc/rc2.d/S75cron
```

Check if the cron daemon is running:

```
# ps -ef | grep cron | grep -v grep
```

Start the cron daemon if not:

```
# /etc/init.d/cron start
```

4.2.5 (O) Set up console over the serial port

This is only necessary if you want to make use of console switching hardware, or for any other reason would like to have the console output routed to the serial port.

4.2.5.1 Configuring serial console on Linux

Edit `/etc/inittab` to set up console over serial port

```
RH 6.x, 7.0: s0:2345:respawn:/sbin/getty ttyS0 9600 vt100
```

```
RH 7.x: s0:2345:respawn:/sbin/agetty ttyS0 9600 vt100
```

SuSE : Run `yast`, choose "System administration | Change configuration file" and set the entry `SERIAL_CONSOLE` to `ttyS0,9600`

Add `ttyS0` to `/etc/securetty`.

Chapter 4: Software installation

Edit `/etc/lilo.conf` to enable `lilo` to operate over serial port and inform kernel to use console over serial port. Include these lines in `/etc/lilo.conf`:

```
serial=0,9600n8
append="console=ttyS0,9600n8"
```

and run `/sbin/lilo` to make the settings active.

Reboot the system

4.2.5.2 Configuring serial console on Solaris:

Configure with `eeprom` as root:

```
# eeprom ouput-device=ttya
# eeprom input-device=ttya
```

4.2.6 (M) Use a common file system for MPI applications

You should have a common file system for MPI applications. Usually this means mounting `/home/*` to a common NFS server.

4.2.7 (O) Use NIS

We recommend using NIS on the cluster to ease management. Here is a short description of how to enable it.

4.2.7.1 Red Hat Linux NIS setup

On the NIS master (usually the frontend):

- set NIS domainname, run `# nisdomainname <yourdomain>`
- edit `/etc/yp.conf`
- edit `/etc/nsswitch.conf`
- edit `/var/yp/securenets`
- run `# sh /etc/rc.d/init.d/ypserv start`
- cd to `/var/yp` and run `# make`
- run `# sh/etc/rc.d/init.d/ypbind start`

On the NIS clients (all other nodes):

- run `# authconfig` and specify NIS domainname and request server via broadcast.

4.2.7.2 SuSE Linux NIS setup

On the NIS master (usually the frontend):

- Run `yast`, choose "System administration | Change configuration file" and:
 - set `YP_DOMAINNAME` to NIS domainname
 - set `YP_SERVER` to correct IP address
 - set `START_YPSERV` to yes
 - set `START_YPBIND` to yes

On NIS clients (all other nodes):

- Run `yast`, choose “System administration | Change configuration file” and:
 - set `YP_DOMAINNAME` to NIS domainname
 - set `YP_SERVER` to correct IP address

4.2.7.3 Solaris NIS setup

On the NIS master (usually the frontend):

- set NIS domainname, run `# domainname <yourdomain>`
- EITHER have these files under `/etc`:
 - `timezone`
 - `netgroup`
 - `bootparams`
- OR edit `/var/yp/Makefile` to not use these maps.
- run `# ypinit -m`
- edit `/etc/nsswitch.conf`
- edit `/var/yp/securenets`
- run `# /usr/lib/netsvc/yp/ypstop`
- run `# /usr/lib/netsvc/yp/ypstart`

On the NIS clients (all other nodes):

- set NIS domainname, run `# domainname <yourdomain>`
- run `# ypinit -c`
- run `# /usr/lib/netsvc/yp/ypstop`
- run `# /usr/lib/netsvc/yp/ypstart`

4.2.8 (M) Provide OpenGL for Scali Universe GUI

The Scali Universe GUI requires OpenGL library for some of its monitoring views. Note that this applies only to workstations actually running the graphical application (remote or local).

- Linux: Mesa has been tested.
- Microsoft: MS OpenGL has been tested.
- SunOS/Solaris: - Mesa and SunOpenGL has been tested (on Sparc).

4.3 Scali software installation

When the operating system has been properly configured you may install your Scali software products. Note that you will need to obtain a valid license for your product in order to complete the installation. The recommended procedure of doing this is to first use your customer ID tag to obtain a demo license from **demolicense@scali.com** (see “Requesting licenses” on page 157 for details), and use this demo license for the initial installation. Then you should request the permanent license.

Chapter 4: Software installation

4.3.1 SSP installation

If the Scali software comes on a CD, start the installation as root by typing the following:

```
# /mnt/cdrom/install
```

If the Scali software has been downloaded from Scali's web site, unzip and untar the tarball in a temporary directory on the frontend machine. Start the installation as root by typing the following (in this example we assume the software has been unzipped and untarred in the /tmp/SSP directory):

```
# /tmp/SSP/install
```

If for any reason the installation has to be terminated before it is finished, use Ctrl-c to terminate. When the installation is to be completed, run as root on the frontend the previous mentioned commands or run the main installation program directly by typing:

```
# /opt/scali/sbin/SSPinstall
```

4.3.1.1 SSP install program options

When running the installation program by one of the methods mentioned above, the actual program running is /opt/scali/sbin/SSPinstall.

```
Usage: SSPinstall [-ciftvlmVuh?]
```

```
-c          start directly in configuration stage.
-i          start directly in installation stage.
-t          run system functional tests only.
-v          run system verification tests only.
-l          create permanent license request.
-V          print SSP version.
-m <media path> path to media packages.
-f          skip frontend check.
-u          upgrade license file.

-h/-?      show this help message.
```

If the configuration went OK but there were problems with the package installation you may restart from the package installation again using the -i option.

4.3 Scali software installation

If you are running with a demo license and want to create a permanent license request, use the `-l` option. And if you have obtained a new license (demo or permanent), you may use the `-u` option to distribute and install the license on the entire cluster system.

4.3.2 Installation explained step by step

The installation program will guide you through the installation process which installs the selected Scali software on the whole cluster. The installation program will detect if you are performing a clean installation or an upgrade. The installation process is done in different stages: configuration, package installation and test. One may restart the installation at any of these stages without the need to run the previous stage again.

4.3.2.1 Introduction

The introductory part of the installation mainly deals with giving the user information about practical issues for the installation. The license terms for the different software modules have to be accepted for the installation to continue. The following example was run after the initial installation was aborted. Questions about using savestate information will only be addressed when there has been an incomplete installation prior to the current installation run.

```
[root@scalii1-11 /root]# /tmp/SSP/install
Welcome to the Scali Software Platform (SSP) installation launcher.
This program will install the Scali SSP installation package
and launch the installer program.
```

```
This program should only be started on the machine you want to be
the frontend. The frontend is among other responsible of:
```

- installing Scali software and upgrades on the cluster
- running the configuration and monitoring servers

```
In small systems (< 8 nodes) this is usually one of the compute nodes.
In larger systems a separate node is preferred.
```

```
The name of this machine: scalii1-11
```

```
Q: Are you sure you want to use this machine as an frontend: [y] y
```

```
...Installing ScaSSP
package ScaSSP is not installed
...launching /opt/scali/sbin/SSPinstall...
```

Chapter 4: Software installation

```
== Introduction =====

Welcome to the Scali Software Platform (SSP) installation program
This program will guide you through the installation procedure.

It is divided into several sections:
  Introduction
    - some general information
    - some sanity checks
  Configuration
    - specify software to install
    - specify nodes to install on
  Installation
    - check that the nodes are correctly prepared
    - the actual software installation on all nodes
    - post installation setup/configuration
  Verification
    - verify that the installation was successful

If you abort in the middle of the installation process you have the
option of saving the current state and restore it at the next invocation.
You may also confirm each single step of the restore to do modifications
anywhere in the restore process.

Whenever your input is required you will see a line like this:
  Q: ...? [<default option>]
The default option is given in square braces and is chosen if you
just press return.

If you experience problems during installation please contact:
  support@scali.com
Wulfkit customers please use the e-mail address:
  wulfkit-support@dolphinics.no

(version SSP_3_0_2 - SSPinstall : 1.96 )

-- Checking permissions ----- OK --
-- Restoring info from savefile ----- ? --
We have detected a state file from a previous installation attempt.
You have the option of restoring the information recorded up to the
point where you last exited.
If you choose to restore you also have the option of qualifying each step
if you want to change something in the recorded information.
Please choose one of the following:
  y : restore (default)
  s : restore with single step qualifying
```

4.3 Scali software installation

```
n : ignore restore possibility

Q: Do you want to restore the recorded state: [y] n

-- Accept license agreements      ----- ?  --

This is licensed software. Before installation can proceed,
you must accept the following license terms for Scali and
integrated third party software:

    1 Scali AS Software License terms
    2 GNU General Public License terms
    3 Console Server License terms
    4 MPICH License terms
    5 OpenPBS Software License terms

Q: Select the license terms to be read, or type "yes" to confirm
   that you have read and do accept the license terms above: [ ] 1

ATTENTION: USE OF THE SOFTWARE IS SUBJECT TO THE SCALI AS SOFTWARE
LICENSE TERMS SET FORTH BELOW. USING THE SOFTWARE INDICATES YOUR
ACCEPTANCE OF THESE LICENSE TERMS. IF YOU DO NOT ACCEPT THESE LICENSE
TERMS, YOU MAY RETURN THE SOFTWARE FOR A FULL REFUND.

The License Terms below govern your use of the accompanying
Software unless one of the following exceptions apply:

- You have a separate signed agreement with Scali AS.
- The Software is licensed under GNU Public License (see GPL.txt).

                SCALI AS SOFTWARE LICENSE TERMS

1. You acknowledge that the software and the documentation
for the Scali software products are copyrighted by Scali AS, and that
while you may acquire a license to use them on a non-exclusive basis,
you will acquire no title or intellectual property rights.

2. You may not modify the content of any of the files of the software
or the online documentation.

3. The software contains proprietary algorithms and methods. You may
not attempt to reverse engineer or disassemble the software.

4. The software is operated under the control of a license manager
that regulates licensed usage of the software. You may not attempt to
modify or tamper with any function of this license manager.
```

Chapter 4: Software installation

5. Transfer. Your license will automatically terminate upon any transfer of the Software. Upon transfer, you must deliver the Software, including any copies and related documentation, to the transferee. The transferee must accept these License Terms as a condition to the transfer.

6. Copies and Adaptations. You may only make copies or adaptations of the Software for archival purposes or when copying or adaptation is an essential step in the authorized Use of the Software. You must reproduce all copyright notices in the original Software on all copies or adaptations. You may not copy the Software onto any public network.

7. You are permitted to print and distribute paper copies of the unmodified online documentation freely. In this case you may not charge a fee for any such distribution.

8. Export Requirements. You may not export or re-export the Software or any copy or adaptation in violation of any applicable laws or regulations.

9. Termination. Scali AS may terminate your license upon notice for failure to comply with any of these License Terms. Upon termination, you must immediately destroy the Software, together with all copies, adaptations and merged portions in any form.

version: \$Id: LICENSE_TERMS,v 1.2 2001/06/14 09:11:56 taa Exp \$

This is licensed software. Before installation can proceed, you must accept the following license terms for Scali and integrated third party software:

- 1 Scali AS Software License terms
- 2 GNU General Public License terms
- 3 Console Server License terms
- 4 MPICH License terms
- 5 OpenPBS Software License terms

Q: Select the license terms to be read, or type "yes" to confirm that you have read and do accept the license terms above: [] yes

4.3 Scali software installation

```
-- Expert mode ----- ? --
This program tries to reduce user interaction to a minimum
in standard mode. However, if you would like to be able to
control the process in more detail:
- Use secure shell (ssh/scp) for installation
- Modify default install set
- Install ScaShMem, ScaIP or ScaFPDisp (*)
- Modify node configuration in an upgrade
- Choose a different package repository
please answer yes to the question below.
Under normal circumstances this should not be necessary.
(*) Available for a limited number of platforms only.

    Q: Do you want to run it in expert mode: [n]

-- Checking source media ----- OK --

4.3.2.2 Installation Configuration
The second part of the installation deals mainly with configuration of the cluster. Next
is an example from a ClusterEdge installation. Note the use of brace expansion when
specifying the node names in the cluster. For a big system this way of specifying the
node names will greatly reduce the workload when installing.

== Installation configuration =====

-- Checking upgradeability ----- INSTALL --
NOTE: Information from previous installation is not found.
      Upgrade is not a choice.

-- Specifying frontend name ----- ? --
This program should only be started on the machine you want to be
the frontend. The frontend is among other responsible of:
- installing Scali software and upgrades on the cluster
- running the configuration and monitoring servers
In small systems (< 8 nodes) this is usually one of the compute nodes.
In larger systems a separate node is preferred.

The name of this machine: scalil-11

    Q: Are you sure you want to use this machine as a frontend: [y] y

-- Read HW documentation ----- ? --
Do you want to read the HW installation guide before continuing?

    Q: Read it now: [n]
```

Chapter 4: Software installation

```
-- Read OS documentation ----- ? --
Do you want to read the OS installation guide before continuing?

Q: Read it now: [n]

-- Specifying node names ----- ? --
Please specify nodenames for your system.
You may use expression with Brace Expansion (see bash man pages).
Include the frontend if it is supposed to be used as a processing node.
Press <return> on an empty line after adding your node(s)

Q: Please enter node name(s) 1 : scali1-{1,2}{1,2}
Q: Please enter node name(s) 5 :

The following nodes have been defined :
scali1-11
scali1-12
scali1-21
scali1-22

Q: Do you accept the above configuration: [y] y

-- Specifying repository ----- DEFAULT-
-- Preparing repository ----- ... --
Creating non-existent path /opt/scali/repository/Linux2.i86pc ...
-- Copying packages to repository ----- ... --
ScaSH      : Removing old versions and copying files to repository: +
ScaPkg     : Removing old versions and copying files to repository: +
ScaDiag    : Removing old versions and copying files to repository: +
ScaLM      : Removing old versions and copying files to repository: +

-- Preparing ScaSH config ----- ... --
-- Installing installation tools ----- ... --
ScaSH      : Preparing package configuration: +
ScaSH      : Installing package
ScaDiag    : Preparing package configuration:
ScaDiag    : Installing package
ScaLM      : Preparing package configuration:
ScaLM      : Installing package
ScaPkg     : Preparing package configuration:
ScaPkg     : Installing package

-- Checking access to nodes ----- OK --
-- Preparing nodes for install ----- OK --
```

4.3 Scali software installation

Entering the license by specifying an existing license file will reduce the risk for errors compared with copy and paste of each FEATURE line.

```
-- Setup license(s) ----- ? --
Please select which type of license(s) you have:
  1) FEATURE line(s), typically demo license key(s)
  2) License file, typically permanent license key(s)

  3) Skip license information! I will do it later...
      NOTE: Later tests may fail due to missing license key(s)!

Q: Please enter your selection: [1] 2

Q: Please enter the path to the license file: /root/license.dat
```

Node categories is a way of specifying what type of software to install on each node. eth_node and sci_node, depending on the presence of a SCI-adapter or not, will be the default selection for a node. The choice of eth_node category will imply installation of software needed for running MPICH applications over ethernet. The choice of sci_node category will imply installation of software needed for running ScaMPI applications over SCI network.

```
-- Determine node categories ----- ? --
Please, wait while retrieving information from nodes ...

The following node categories have been defined :
  scali1-11  sci_frontend  eth_frontend  sci_node  eth_node
  scali1-12  sci_node    eth_node
  scali1-21  sci_node    eth_node
  scali1-22  sci_node    eth_node

Q: Do you accept the above configuration: [y] y
```

If you install a product which includes extended hardware monitoring features, this will be automatically set up at this stage given that qualified hardware for such purpose is used. Monitoring of CPU temperatures and cooling fan speeds are typical examples of such features.

```
-- Specifying node hardware ----- ? --
You may benefit from extended system monitoring when using qualified
node hardware from certain vendors. If your hardware is not listed
just use the default value (Hardware not listed).
It is assumed that all hosts are using the same hardware.
```

Chapter 4: Software installation

```
1 Hardware not listed
2 Dell PowerEdge 1550
3 Dell PowerEdge 1650
4 Dell PowerEdge 2650
5 SuperMicro SuperServer 6010H
6 SuperMicro SuperServer 6022C
7 SuperMicro SuperServer 6012P-6
```

Q: Please select what hardware is used: [1]

```
1 Hardware not listed
```

Q: Do you accept the above configuration: [y]

```
-- Specifying software to install ----- ?    --
-- Selecting console switch ----- ?    --
If you have the serial ports connected to a terminal server you must
install the Scali console server to be able to use them as consoles.
```

Q: Do you want to install the console server: [n] y

```
-- Selecting power switch(es) ----- ?    --
If you have power switches in the system you must install the
Scali power switch server to be able to switch power on and off.
```

Q: Do you want to install the power switch server: [n] y

If installation of OpenPBS (Parallel Batch System) is chosen, a valid non-root user has to be specified in order to enable some tests of this software after installation. However, the tests may be cancelled by not specifying a user.

```
-- Selecting OpenPBS ----- ?    --
You may install the free queue system OpenPBS (www.openpbs.org).
The ScaOPBS package is a binary distribution of OpenPBS with easy
installation and configuration of OpenPBS and job submission script
for ScaMPI and MPICH applications.
```

Q: Do you want to install OpenPBS queue system : [n] y

Testing of OpenPBS requires a non-root username, the username must be defined at all nodes.
Use empty username (default) to skip the OpenPBS test.

Q: Please give username : [] testuser

4.3 Scali software installation

The following OpenPBS selection has been made :

- OpenPBS will be installed on the system.
- Test of OpenPBS will be performed as user: testuser.

Q: Do you accept the above configuration: [y]

The following section shows you how configuration of serial console switch and power switches is performed during installation. This requires the use of qualified switch hardware and a product that supports console and power switching.

-- Specifying console switch map ----- ? --
Please specify the console mappings.

Q: Please enter console switch hostname for host scali1-11: [] scali-cs1
Q: Please enter console switch TCP-port number for host scali1-11:
[10001] 10001

Q: Please enter console switch hostname for host scali1-12: [scali-cs1]
Q: Please enter console switch TCP-port number for host scali1-12:
[10002]

Q: Please enter console switch hostname for host scali1-21: [scali-cs1]
Q: Please enter console switch TCP-port number for host scali1-21:
[10003]

Q: Please enter console switch hostname for host scali1-22: [scali-cs1]
Q: Please enter console switch TCP-port number for host scali1-22:
[10004]

The following console switch mappings have been defined:

Host:	Console-switch:	Port:
scali1-11	scali-cs1	10001
scali1-12	scali-cs1	10002
scali1-21	scali-cs1	10003
scali1-22	scali-cs1	10004

Q: Do you accept the above configuration: [y] y

-- Specifying power switches ----- ? --
Please classify your power switch(es).

Please select what type of switch the power switch number 01 is.
Select switch type by entering a number from the list below:
1) Baytech RPC-3 with ethernet connection
2) Pulizzi with ethernet connection (serial switch)

Chapter 4: Software installation

9) Finish configuration

Q: Please enter your selection: [1] 1

Q: Please enter hostname for power switch number 01: scali-pow3

Please select what type of switch the power switch number 02 is.
Select switch type by entering a number from the list below:

- 1) Baytech RPC-3 with ethernet connection
- 2) Pulizzi with ethernet connection (serial switch)

9) Finish configuration

Q: Please enter your selection: [1] 9

The following power switch(es) have been defined :

Power-switch:	Type:	Configuration:
powsw01	baytech	scali-pow3

Q: Do you accept the above configuration: [y]

-- Specifying power switch map ----- ? --
Please specify the power mappings.

Please specify which power switch node scalil-11 is connected to:

Power-switch:	Type:	Configuration:
powsw01	baytech	scali-pow3

Q: Please enter power switch for host scalil-11: [] powsw01

Q: Please enter power switch port (outlet) number for host scalil-11:

[1] 1

Please specify which power switch node scalil-12 is connected to:

Power-switch:	Type:	Configuration:
powsw01	baytech	scali-pow3

Q: Please enter power switch for host scalil-12: [powsw01]

Q: Please enter power switch port (outlet) number for host scalil-12:

[2]

Please specify which power switch node scalil-21 is connected to:

Power-switch:	Type:	Configuration:
powsw01	baytech	scali-pow3

Q: Please enter power switch for host scalil-21: [powsw01]

Q: Please enter power switch port (outlet) number for host scalil-21:

[3]

4.3 Scali software installation

Please specify which power switch node scali1-22 is connected to:

Power-switch:	Type:	Configuration:
powsw01	baytech	scali-pow3

Q: Please enter power switch for host scali1-22: [powsw01]

Q: Please enter power switch port (outlet) number for host scali1-22:

[4]

The following power switch mappings have been defined :

Host:	Power-switch:	Port:
scali1-11	powsw01	1
scali1-12	powsw01	2
scali1-21	powsw01	3
scali1-22	powsw01	4

Q: Do you accept the above configuration: [y]

Please wait while retrieving information from the system...

```
-- Preparing repository ----- ... --
-- Copying packages to repository ----- ... --
ScaComd      : Removing old versions and copying files to repository: +
ScaConfC     : Removing old versions and copying files to repository: +
ScaConfNd    : Removing old versions and copying files to repository: +
ScaConfSd    : Removing old versions and copying files to repository: +
ScaCons      : Removing old versions and copying files to repository: +
ScaDesk      : Removing old versions and copying files to repository: +
ScaDiag      : Removing old versions and copying files to repository: +
ScaDiagSC    : Removing old versions and copying files to repository: +
ScaEnv       : Removing old versions and copying files to repository: +
ScaExecd     : Removing old versions and copying files to repository: +
ScaFPDisp    : Removing old versions and copying files to repository: +
ScaFPMB2     : Removing old versions and copying files to repository: +
ScaFmpich    : Removing old versions and copying files to repository: +
ScaIP        : Removing old versions and copying files to repository: +
ScaIPadap    : Removing old versions and copying files to repository: +
ScaLM        : Removing old versions and copying files to repository: -+
ScaMAC       : Removing old versions and copying files to repository: +
ScaMACadap   : Removing old versions and copying files to repository: +
ScaMACddk    : Removing old versions and copying files to repository: +
ScaMPE       : Removing old versions and copying files to repository: +
ScaMPI       : Removing old versions and copying files to repository: +
ScaMPICht    : Removing old versions and copying files to repository: +
ScaMPItst    : Removing old versions and copying files to repository: +
ScaMond      : Removing old versions and copying files to repository: +
ScaOPBS     : Removing old versions and copying files to repository: +
```

Chapter 4: Software installation

```
ScaOPBSdk : Removing old versions and copying files to repository: +
ScaOSusbrm : Removing old versions and copying files to repository: +
ScaPkg : Removing old versions and copying files to repository: -+
ScaPowd : Removing old versions and copying files to repository: +
ScaSCI : Removing old versions and copying files to repository: ++
ScaSCIadap : Removing old versions and copying files to repository: +
ScaSCIddk : Removing old versions and copying files to repository: +
ScaSH : Removing old versions and copying files to repository: --+
ScaSISCI : Removing old versions and copying files to repository: +
ScaSNMPd : Removing old versions and copying files to repository: +
ScaSNMPT : Removing old versions and copying files to repository: +
ScaSSP : Removing old versions and copying files to repository: +
ScaSensor : Removing old versions and copying files to repository: +
ScaShMem : Removing old versions and copying files to repository: +

-- Copying config to repository ----- ... --
-- Preparing ScaSH config ----- ... --
-- Modifying ScaPkg config ----- ... --
```

Some pre-installation tests will be run in the last part of the installation configuration part.

```
-- Checking hostnames ----- OK --
-- Checking some package versions ----- OK --
-- Checking for glibc memory leak ----- OK --
-- Checking required applications ----- OK --
-- Testing AT daemon ----- OK --
-- Checking bigphysarea patch ----- OK --
-- Checking FPU control word bug ----- OK -
-
-- Checking kernel versions ----- OK --
-- Checking HW support ----- OK --
-- Checking OS support ----- OK --
-- Checking SCI boards ----- OK --
-- Checking MPI service ----- OK --
```

Next is an example showing how to change the default node category suggestion. The Determine node categories sub-section is part of the Installation configuration section. The example starts with the default category selection of `sci_` and `eth_node/-frontend`, and shows how to add `smp_node` and `smp_frontend` categories to the frontend machine in the system. Other predefined categories may also be chosen. These are defined in `/opt/scali/etc/ScaPkg.conf` in the lines prefixed with the keyword `supercategory`. The choice of `smp_node` and `smp_frontend` category will

4.3 Scali software installation

imply installation of software needed for running ScaMPI applications on an SMP machine without involving the SCI network. Hence an SCI adapter is not needed. This will typically be a choice for development and porting purposes.

```
-- Determine node categories ----- ? --
Please, wait while retrieving information from nodes ...

The following node categories have been defined :
  scali1-11  sci_frontend  eth_frontend  sci_node  eth_node
  scali1-12  sci_node     eth_node
  scali1-21  sci_node     eth_node
  scali1-22  sci_node     eth_node

Q: Do you accept the above configuration: [y] n

Please specify nodename(s) to modify (brace expansion allowed):
Press <return> on an empty line after adding your node(s)

Q: Please enter node name(s) : scali1-11
Q: Please enter node name(s) :

The following nodes have been selected for modification :
  scali1-11

Q: Do you accept the above selection: [y]

Please select one or more of the following categories :
  workstation
  sci_node
  smp_node
  eth_node
  sci_frontend
  smp_frontend
  eth_frontend

Categories : sci_frontend eth_frontend sci_node eth_node smp_frontend
smp_node

The following categories have been selected :
  sci_frontend
  eth_frontend
  sci_node
```

Chapter 4: Software installation

```
eth_node
smp_frontend
smp_node
```

Q: Do you accept the above selection: [y] y

```
The following node categories have been defined :
scalil-11  sci_frontend eth_frontend sci_node eth_node
smp_frontend smp_node
scalil-12  sci_node eth_node
scalil-21  sci_node eth_node
scalil-22  sci_node eth_node
```

Q: Do you accept the above configuration: [y] y

4.3.2.3 Software installation

During the Software installation part most of the Scali software will be distributed to the nodes and installed. This is also true for the frontend. Normally there will be no need for user interaction during this part of the installation.

```
== Software installation      =====
-- Installing packages      ----- Wait ... --
Checking accessibility of package repositories ...
Checking availability on hosts chosen to be updated, please wait ...
This program installs software on specified (or default) hosts
version      : 1.52
repository   : /opt/scali/repository

Checking configuration
Cleaning up from previous runs on all hosts ...
Find platforms on hosts ...

Find out on each host which packages are needed
Find out which packages we have for the platforms detected
Removing invalid platforms
Removing invalid hosts due to invalidated platforms
Find out which packages which go to each host (based on configuration
data)
Find out which packages are valid for each category
Find out which packages are valid for each host
```

4.3 Scali software installation

```
Match valid package names with available packages in repository
Removing hosts not allowed for updates or no packages available
Starting remote polling script on hosts ...
Copy host dependent package list to respective hosts ...

Waiting on feedback from hosts which packages are needed
Poll for list of needed packages on remote hosts ...
Remove hosts not in need of any updates

Copy packages to hosts
Copy files to hosts ...
Checking platform: Linux2.i86pc
Package ScaComd      is now copied to      4 host(s) ...
Package ScaCons     is now copied to      1 host(s) ...
Package ScaDesk     is now copied to      1 host(s) ...
Package ScaEnv      is now copied to      4 host(s) ...
Package ScaExecd    is now copied to      1 host(s) ...
Package ScaFPMB2    is now copied to      4 host(s) ...
Package ScaLM       is now copied to      3 host(s) ...
Package ScaOSusbrm  is now copied to      4 host(s) ...
Package ScaPowd     is now copied to      1 host(s) ...
Package ScaSCI      is now copied to      4 host(s) ...
Package ScaSH       is now copied to      3 host(s) ...
Package ScaSISCI    is now copied to      4 host(s) ...
Package ScaSNMPd    is now copied to      4 host(s) ...
Package ScaSNMPt    is now copied to      1 host(s) ...
Package ScaSSP      is now copied to      3 host(s) ...
Package ScaConfNd   is now copied to      4 host(s) ...
Package ScaConfSd   is now copied to      1 host(s) ...
Package ScaDiagSC   is now copied to      1 host(s) ...
Package ScaFmpich   is now copied to      4 host(s) ...
Package ScaMond     is now copied to      1 host(s) ...
Package ScaMPI      is now copied to      4 host(s) ...
Package ScaMPICHt   is now copied to      4 host(s) ...
Package ScaMPItst   is now copied to      4 host(s) ...
Package ScaOPBS     is now copied to      4 host(s) ...
Package ScaSCIddk   is now copied to      1 host(s) ...
Package ScaConfC    is now copied to      1 host(s) ...
Package ScaMPE      is now copied to      4 host(s) ...
Package ScaSCIadap  is now copied to      1 host(s) ...
Tell all hosts that all packages have been copied ...
```

Chapter 4: Software installation

```
Get results from hosts
  Poll for log files on remote hosts ...
  Checking log files
    scalil-11      : No errors
    scalil-12      : No errors
    scalil-21      : No errors
    scalil-22      : No errors

Cleaning up after us and exiting
```

4.3.2.4 Post installation fix

The Post installation fix section will take care of configuring parts of the software which has to be configured after the software has been installed. The first example shows a Clusteredge installation. Note that the configuration of the SCI network is automatically taken care of. When installing Open PBS, some of the configuration has to be performed after installation of the software.

```
== Post installation fix      =====
-- Specify SCI network      ----- Wait ...
--
  Please wait while the SCI network configuration is taking effect.
  Unable to get system information from server..
-- Configuring Interconnect  ----- ... --
  Configuring interconnect, please wait some seconds ...
  Requesting server to reroute with routingtype SCA_ROUTE_MAXCY.

-- Configuring OpenPBS      ----- ... --
```

In the second example of the Post installation fix section, a WulfKit installation is shown and the SCI network has to be specified manually.

```
== Post installation fix      =====
-- Specify SCI network      ----- ... --
  The SCI nodes should be connected in one out of three possible
  topologies.
  The simplest topology is a single ring where only SCI Link 0,
  or L0 for short, is used to connect all nodes.
  The next topology is a 2D torus where L0 is used to connect rings
  in one dimension and L1 is used to connect rings in the other dimension.
  The third topology is a 3D torus where connections are the same as in a
  2D torus and L2 is used to connect rings in the third dimension.
```


4.3 Scali software installation

- 1 Single ring (using L0)
- 2 2D torus (using L0 and L1)
- 3 3D torus (using L0, L1 and L2)

Q: Please specify which SCI network topology is used: [2] 2

Q: Please specify the number of nodes
connected on a ring using SCI Link 0 or L0 (1-16): 2

Q: Please specify the number of nodes
connected on a ring using SCI Link 1 or L1 (1-16): 2

The configuration needs to know the position of each node in the SCI network.

Please select which method of node position specification:

- 1) Automatically from node list, L0 rings first
- 2) Manually give the position of each node

Q: Please enter your selection: [1]

The following SCI configuration has been defined (name L0 L1 L2):

```
scali1-11      111
scali1-12      211

scali1-21      121
scali1-22      221
```

Q: Do you accept the above configuration: [y] y

Please wait while the SCI network configuration is taking effect.

Unable to get system information from server..

```
-- Configuring Interconnect ----- ... --
Configuring interconnect, please wait some seconds ...
Requesting server to reroute with routingtype SCA_ROUTE_MAXCY.
```

4.3.2.5 Functional testing

The last parts of the installation deals mainly with tests of the system. In the System Verification section some verification tests will be performed locally on each node.

```
== System Verification =====
-- Checking SCI driver ----- OK --
-- Checking SCI board jumpers ----- OK --
-- Checking SCI links ----- OK --
-- Testing SCI communication ----- OK --
```

Chapter 4: Software installation

In the System test section some system tests will be performed to verify that the system seems to work as supposed to.

```
== System test          =====
-- Testing MPI communication  ----- OK  --
-- Testing MPI performance    ----- OK  --
-- Testing MPICH communication ----- OK  --
-- Testing MPICH performance  ----- OK  --
-- Testing OpenPBS           ----- OK  --

== Install complete!    =====
```

In the Epilogue section some information will be printed before the installation is complete and the installation program will exit.

```
-- Epilogue            ----- ...  --

Scali Software Getting Started Guide
-----
Most of the Scali software is accessible through the graphical
desktop; ScaDesktop.
To start the desktop issue this command on the command line:

# /opt/scali/bin/scadesktop

Please note that the desktop will be different for root than
for ordinary users. Please also note that the desktop is not
available for "frontend only" installations.
For help on specific issues regarding the installation and
operation of the Scali software please refer to the "Scali
System Guide".

---
(document version: $Id: GettingStarted,v 1.3 2001/11/29 13:47:55 ae Exp $)

== Clean up and exit ... =====

Removing recorded information
```

4.3.3 SSP Uninstall

If you wish to remove the SSP at a later stage you can run the uninstall program directly on the frontend as root:

```
# /opt/scali/sbin/SSPuninstall
```

You also have the possibility of running the uninstall program from the directory where the Scali software once were untarred and unzipped:

```
# /tmp/SSP/uninstall
```

or run the uninstall program from the CD:

```
# /mnt/cdrom/uninstall
```

4.3.4 Uninstallation example

When starting the uninstall program you will be asked to confirm that you really want to start uninstalling the SSP. There will also be a second question where you can choose to keep the repository or not. The repository is where all the software packages in the SSP are kept. During installation the installation program will distribute and install single software packages from the repository. If you are going to uninstall the software on the system and do a reinstallation using the same software versions, you will typically choose to keep the repository.

```
[root@scali1-11 /root]# /opt/scali/sbin/SSPuninstall

== Checking installation      =====

Welcome to the Scali Software Platform (SSP) uninstallation program

This program will guide you through the uninstallation procedure.

Whenever your input is required you will see a line starting with 'Q:
...'.
The default option is given in square braces and is chosen if you press
return.

If you experience problems during uninstallation please contact:
  support@scali.com
Wulfkit customers please use the e-mail address:
  wulfkit-support@dolphinics.com
```

Chapter 4: Software installation

```
-- Checking permissions ----- OK --
-- Find existing frontend ----- OK --
  The following node will be used:
    scalil-11

-- Checking Frontend ----- OK --
-- Find existing nodes ----- OK --
  The following nodes will be used:
    scalil-11
    scalil-12
    scalil-21
    scalil-22

-- Find existing repository ----- OK --
  The following repository path will be used:
    /opt/scali/repository

-- Checking access to nodes ----- OK --

== Remove installation =====

-- Confirmation ----- ? --
  You will now remove the SSP from all involved nodes.

  Q: Please type "yes" to confirm, "no" to decline: [no] yes

  Will you also remove the repository on the frontend :

    (/opt/scali/repository)

  Q: Please type "yes" to confirm, "no" to decline: [no] yes
-- Remove packages and files ----- ! --
  Removing packages on nodes, please wait ...

  Removing config files on nodes ...
  Removing references to /opt/scali ...

  Removing references to /opt/scali on frontend
  Removing packages on frontend, please wait ...

  Removing config files on frontend

  Removing repository on frontend

== Uninstall complete! =====
```

Aimed at all users of Scali systems this chapter explains the organisation of, and how to use the **Scali Universe** graphical user interface. Although Scali Universe consists of a great number of other software components working together, the term “Scali Universe” is commonly used to refer to the GUI.

5.1 Overview

The Scali Universe system management GUI enables “single system view” monitoring, configuration and use of Scali cluster systems. The Scali Universe is currently available in two versions: Universe and Universe XE. Universe is the entry level cluster management solution which provides basic installation, monitoring and management features. Universe XE is the extended professional cluster management solution which incorporates more features for professional/industrial cluster computing. In this manual features available only in Universe XE will be marked with **Universe XE**. In the GUI, features not available will simply not show up or be greyed out in the menus.

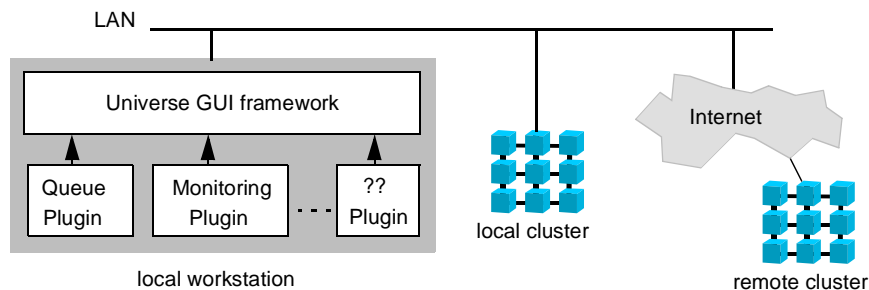


Figure 5-1: Scali Universe overview

The Scali Universe GUI is built around a plugin architecture which consists of a framework and a number of client-side plugins. The framework provides backbone functionality while the plugins provides the GUIs for different components of Scali Universe. In this chapter we will explain the framework and some basic plugins. The more specialized plugins will be explained in chapters that also address the related functionality like monitoring in chapter 6, or the queue system interface in chapter 8.

Chapter 5: Scali Universe

Figure 5-1 illustrates how the different pieces fit together. The plugins are all part of the binary distribution (rpm) of the Scali Universe GUI and will therefore run on the same system as the GUI no matter where the managed system is located. Plugins are loaded dynamically on demand - a demand determined by the licensing and capabilities of the managed system. When new plugins are loaded they will show up as additional menu items in the Scali Universe MainWindow.

The general idea is that the Scali Universe GUI is a somewhat “fat client” for the Universe management system. Available features and licensing are properties of the managed systems, but will be reflected in the GUI on a per system basis. This means that the GUI itself is free from licensing restrictions, something which enables you to install copies wherever it seems feasible. Your home-office, your portable, the system console and so on.

All communication between the Universe GUI and the managed systems, even the terminal emulators, runs through a single secure encrypted socket connection. This is the only connection needed for both local and remote cluster systems, providing full remote management capabilities.

5.2 Getting started with Scali Universe

5.2.1 Prerequisites

The only prerequisites for running Scali Universe is that the machine you are using has network access to the frontends of the Scali systems you want to use. If you want to use the 3D monitoring options you must also have an OpenGL 1.2 compatible library installed. On Linux systems this is provided by the Mesa version 3.2 or newer. You should also consider a graphics card which provides hardware acceleration of OpenGL under your operating environment.

5.2.2 Installation

The Scali Universe GUI is installed on the frontend node as part of the SSP installation, but in most cases the frontend is not your local workstation so you would want to install it locally too. After SSP installation you will find the `ScaDesk` software package in the repository `/opt/scali/pkg/<platform>/` on the frontend node. The `ScaDesk` name is a relic from the time the Scali Universe GUI was called the Scali Desktop. To install the package on Linux systems use the rpm package manager program like this:

```
# rpm -ivh ScaDesk.Linux2.i86pc-<version>.rpm
```

5.2 Getting started with Scali Universe

For instructions on how to install the Scali Universe GUI on other platforms please consult the text file instructions in `/opt/scali/doc/ScaDesk/INSTALL`.

5.2.3 Running

Due to the heavy use of graphic presentations, we recommend that you run Scali Universe on your local workstation. To start Scali Universe on Unix systems, type:

```
% /opt/scali/bin/scadesktop
```

On Windows systems use the “Run” option from the start menu and select - or double-click the file:

```
C:\Program Files\ScaDesktop\bin\ScaDesktop.exe
```

and the program will start. After an introductory glimpse of the Universe startup screen, the System Management window will appear. Scali Universe can control multiple Scali Systems which again may consist of hundreds of nodes. The Scali System Management window is where you select which Scali System(s) to manage.

5.2.4 Configuration of the Scali Universe GUI

Before you can start using the Scali Universe GUI, you must provide some information about the systems you want to manage. This is known as configuration of the Scali Desktop. As a result, you will have a personalized desktop configuration (user profile). The user profiles are stored in the file: `$HOME/.scali/ScaDesk.conf`.

When you start Scali Universe for the first time you will see a dialogue box with the following warning: “Could not read configfile, using system defaults”. This is because no configuration file was found in `$HOME/.scali`. Just press “OK” to get on with the configuration. The Scali System Management window will then appear.

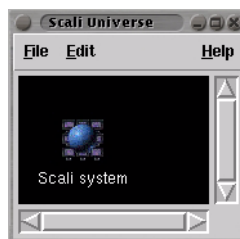


Figure 5-2: Initial unconfigured Scali System Management window

Chapter 5: Scali Universe

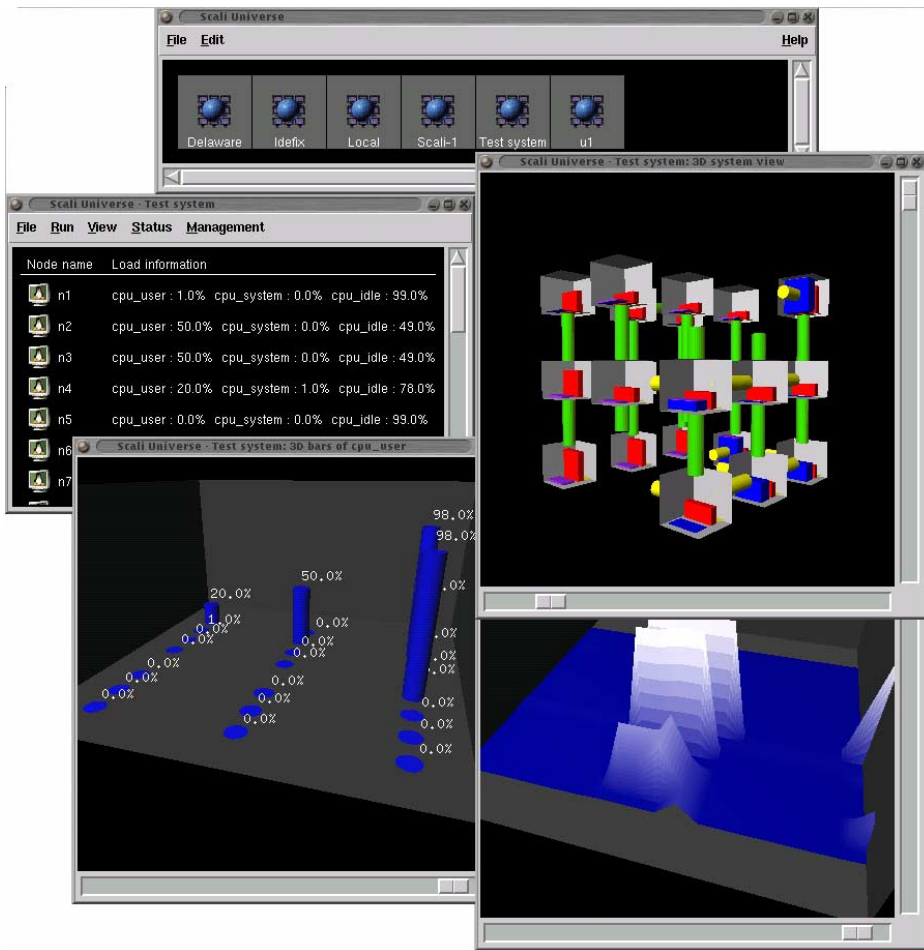


Figure 5-3: An example Scali Universe session with multiple monitoring views

5.2 Getting started with Scali Universe

Initially, only a default Scali System with system default values is available, These default values have very limited use, since they will only work if you run the desktop locally on the frontend of a Scali system. Hence you should configure Scali Universe to enable remote access to this and all other available Scali systems. Select **Edit→Configuration...** from the main menu to open the system configuration window.:

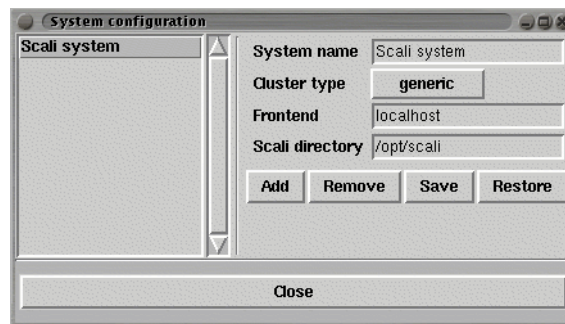


Figure 5-4: Configuration window showing system defaults.

In the System Configuration window select the “*Scali system*” and edit System name and Frontend to the correct values for your system. Usually default values for Cluster type and Scali directory are correct, and can be left unchanged. To make the changes take effect and save the newly edited configuration press Save. More Scali Systems may be added to your configuration like this: Press the Add button and you will be prompted for a new system name. Enter a unique system name, edit the Frontend name and press Save again..



Figure 5-5: Example System configuring

Removal of undesired systems is done by selecting a system and then pressing the Remove button. The Restore button will restore the contents of the configuration window from the last saved profile. Finally to finish the configuration, press Close in the System Configuration window and the System Management window will be updated to show a “cluster icon” and name for each of the configured systems.:



Figure 5-6: Scali System Management window with five Scali Systems

5.2.5 Logging in to a system

Like any other single networked computer you must also log into a Scali System before you can start using any of the tools in Scali Universe. To login to a Scali system you simply double-click on the corresponding system icon in the System Management window. First you will see small “countdown” graphic while the system is contacted. When connection is established you will be presented with a “login” dialogue box for this system. Enter your user name and password and press OK.

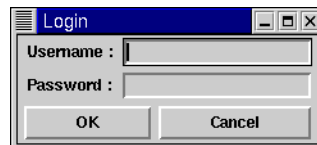


Figure 5-7: Login dialogue box

The username and password will then be encrypted and sent for verification locally on the frontend for this Scali system. Please note that it is therefore your access rights on the Scali system that counts, not your access rights on the workstation running the Scali Universe GUI. Depending on whether you log in as a regular user or as “root” the Scali Desktop will operate in either regular user mode or system administrator mode

The difference is that some tools, notably configuration and maintenance modules, are not available to regular users. Maintenance and configuration modules gives access to remote power control, interconnect configuration and software installation.

5.3 Using Scali Universe

When logged in to a Scali System, the MainWindow of Scali Desktop appears. The main window of the Scali Desktop shows all nodes in the network. The node symbols explained in table 5-1, are used to indicate the operating system and status of each node.

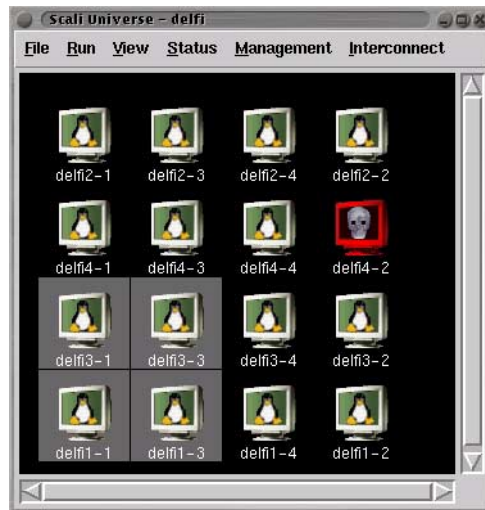


Figure 5-8: Scali Universe system main window with four selected nodes.

5.3.1 Node selection

One important use of the main window is node selection. This because most of the application launchers available from the Run menu requires that you have selected a set of nodes as targets for the application. Some launchers also has a “live link” to the main window which means that while having a launcher window open, you may change the node selection in the MainWindow and re-run the application on the new set of nodes.

Chapter 5: Scali Universe

You may select set of nodes by left-dragging the mouse in the window. Alternatively nodes may be toggled between selected and unselected state by *<Ctrl>+left-click*. Selected nodes will appear with a grey background in the window.





	An operational node running the Linux operating system		Node for which the status and operating system has not yet been determined
	An operational node running the Solaris operating system.		Unreachable node.

Table 5-1: System MainWindow node symbols

5.3.2 MainWindow menu overview

The contents of the main menu will vary slightly depending on whether you are logged into the system as “root” user or not. At the top level it is only the **Software** menu, which is used for software installation that disappears completely for normal users.

Menu Item	Description
File	The file menu only contains a Close button, which will terminate the session to this Scali system.
Run	The Run menu contains the different application launcher interfaces for the system, including queue system submission.
View	Opens the view menu from which different presentations for the nodes in the main window can be selected.
Status	The Status menu contains functions from the Scali Monitoring system - ScaMon, including alarms

Table 5-2: System MainWindow menu overview

Menu Item	Description
Management	The Management menu contains “hard” system management functions like power, console, shutdown
Interconnect	Only available on systems with high speed interconnects installed. Contains functions from the Scali interconnect management system - ScaConf.
Software	Contains functions from the Software installation and management system.

Table 5-2: System MainWindow menu overview

5.3.3 Different system views - the View menu

The view menu gives you three different alternatives for node presentations in the MainWindow. The default Large Icons view is shown in figure 5-8. Other available views are Small Icons which gives room for more nodes on big systems, and Detail which is a compound view combining some textual monitoring into with the graphical node status. Both can be seen figure 5-9.

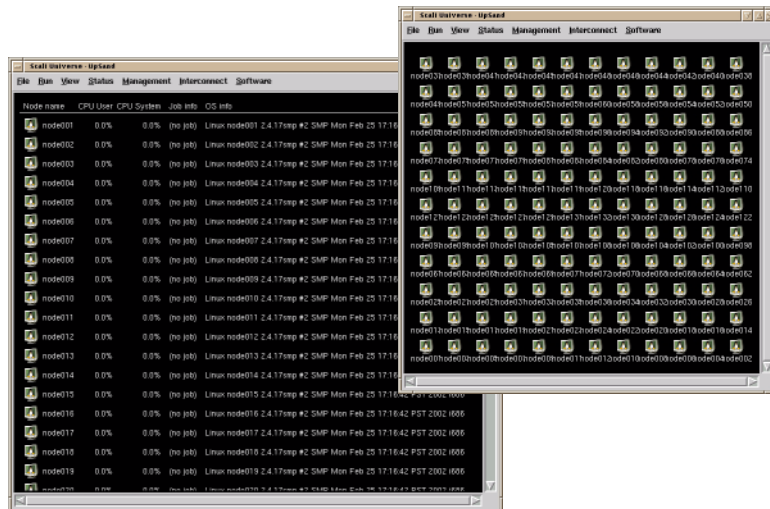


Figure 5-9: Main Window views: “Detail View”(left) and ”Small Icons”(right)

5.3.4 Running programs - the Run menu

The Run menu is used for execution of programs on multiple nodes in a Scali System. This may be MPI programs, shell commands or terminal emulators. Table 5-3 below gives an overview of the Run menu items. Note here that the **Submit Job...** item is only available to non-root users and if a queue system is installed. .

Run Menu Item	Description
MPI Program...	Opens the ScaMPI monitor window to control execution of ScaMPI programs on the system.
MPICH Program...	Opens the MPICH window to control the execution of MPICH jobs on the system.
Submit Job...	If a queue system is installed, this will open the job submission window of the queue system. Its use is described in detail in section 8.8.1 on page 124.
Parallel Shell...	Opens the parallel shell window to run shell commands on all selected nodes.
Node Terminal...	Opens terminal window sessions on all selected nodes
Frontend Terminal...	Opens a terminal session to the frontend of the system.

Table 5-3: Main window: Run menu items explained

5.3.4.1 Working with runsets

The ScaMPI monitor window, MPICH launcher and queue submission window all supports runsets. A runset is a collection of all information needed to run the program: program name, program (command line) options and in case of ScaMPI, MPI monitor options. The idea is that you can save time and effort by creating runsets for programs you use frequently. You will find the runset functionality under the File menu where available:

Menu Item	Description
Open Runset ...	Opens a runset replacing current contents of window
Save Runset ...	Saves current contents of window as a runset

Note: Node selections are not part of the runset. This means that when opening a runset you'll still have to select nodes to run on. It also means that you can use the same runset on a different set of nodes than it was created for.

5.3.4.2 Running ScaMPI MPI programs

From the Scali MPI monitor window, execution of MPI programs at selected nodes may be controlled. To invoke the Scali MPI monitor window, select MPI Program... from the Run menu.

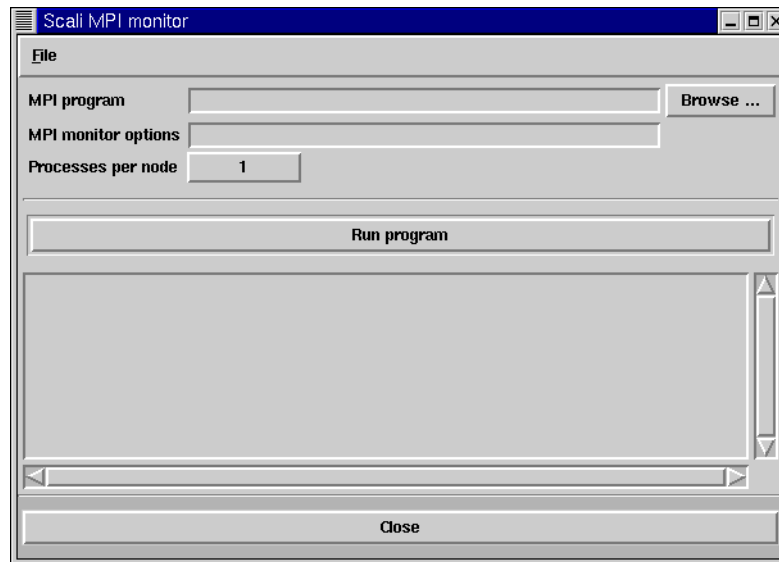


Figure 5-10: Scali MPI monitor window.

From this window a MPI program may be started or stopped. The following options are available:

- **MPI Program**
Specify MPI program to be started. Any options to the program is also specified here. The Browse... button may be used for selecting program.
- **MPI monitor options**
Specify options for **mpimon**, see ScaMPI User's Guide for more information.
- **Processes per node**
The number of instances of the program to be started on each node.

- **Run program**
Start the selected program by pressing this button. The program will be started on the nodes selected in the main desktop window. Output to “stdout” and “stderr” is printed in the centre window. While a program is running the button becomes the Stop Program button.
- **Stop program**
Only available when a program is running. By pressing this button the program will be aborted. All instances of the program at all nodes will be stopped. Only MPI programs started from the Scali MPI monitor window may be stopped.

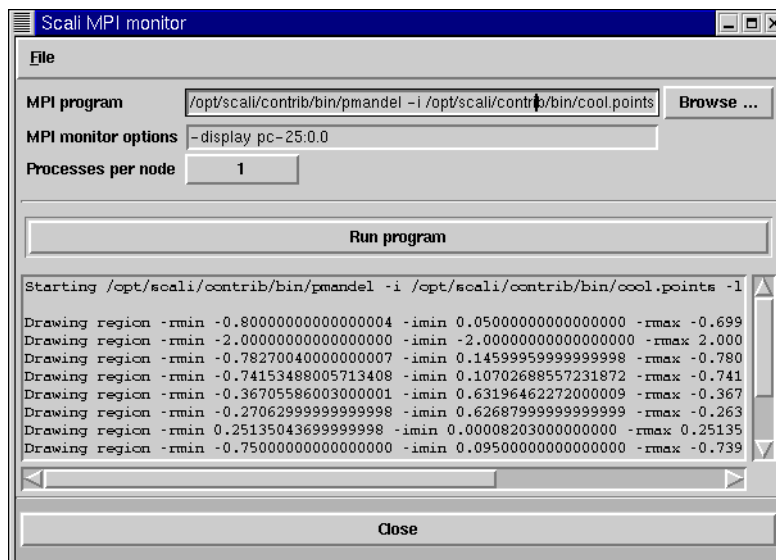


Figure 5-11: Scali MPI monitor window with output from an MPI program.

From the File menu in Scali MPI monitor you may open and save runsets.

5.3.4.3 Running MPICH programs

If the Scali MPICH distribution is installed (ScaFmpich), you will also get a graphical launcher window for MPICH applications. Select **MPICH program ...** from the Run menu to open the window. Except for the missing field for MPI monitor options, the MPICH launcher behaves exactly like the ScaMPI MPI monitor window described in the previous section. The MPICH launcher supports runsets.

5.3.4.4 Running Parallel Shell commands

The parallel shell window allows you to run shell commands on selected nodes in parallel. This is a graphical frontend to a subset of the ScaSh parallel shell utilities. (See “ScaSH - parallel shell tools” on page 131.) To open the parallel shell window select **Parallel shell ...** from the **Run** menu. You may then type in the command in the **Command to run** field, and press the **Run** command button to start it. Just like in the MPI launchers the **Run** command button doubles as a **Stop** command button while the command is running. Output from the commands will be presented in the window as shown in figure 5-12.

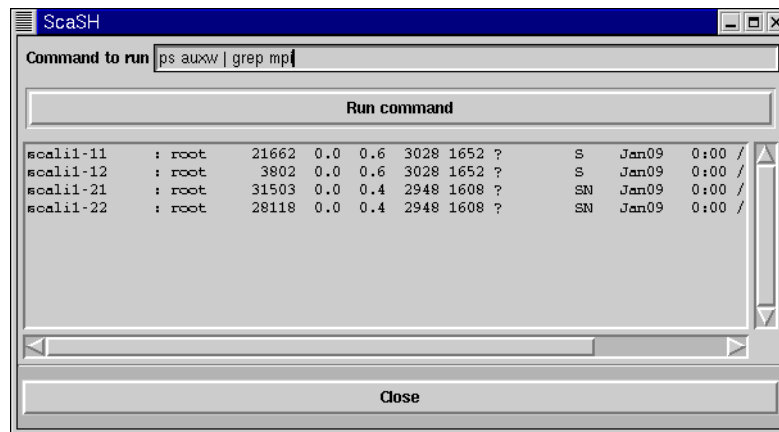


Figure 5-12: Example of a Parallel shell command

5.3.4.5 Node terminal sessions

By using the **Node Terminal ...** option from the **Run** menu, terminal emulators will be started with connections to all selected nodes. As you will soon find out this is a very practical feature for a number of reasons: Firstly the terminal emulator is built into the Scali Universe, so you do not need any other package on the system in order to run Xterm like terminal sessions. This is true even for the MS-Windows version of the product. Secondly the communication runs through the single encrypted socket connection (SSC) from Scali Universe to the system frontend. This gives you secure remote terminals on any node in the system at the click of a button, without struggling with ssh, Xterm and DISPLAY variables.

Chapter 5: Scali Universe

If more than one node was selected, you will get a small window named Master in addition to one terminal window for each selected node. This Master terminal window provides you with terminal broadcasting and command history..



Figure 5-13: The Master window for terminal broadcasting

When you move the mouse cursor into the mouse window and start typing, everything you type is replicated to all terminal windows. Using the up and down arrow keys allows you to scroll up and down in previous commands. Note that for security reasons the echo in the Master terminal window has been turned off. Typing "exit" into the Master window will close all sessions.

5.3.4.6 Frontend terminal session.

Similar to the node terminals the Frontend Terminal ...option enables a quick and easy way to get a terminal connection to the system frontend.

5.3.5 Management menu

The Management menu provides the tools for system management related tasks of the Scali system, including the use of remote power and console switches.

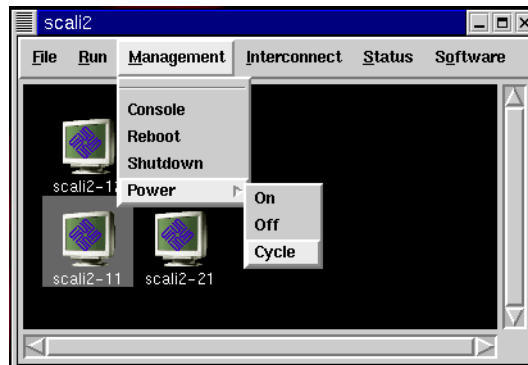


Figure 5-14: The Management menu

The items in the Management menu are described in table 5-4, note that only the Console option is available to ordinary users, all other items requires root privileges. Furthermore Console and Power functionality requires additional hardware and will be greyed out if no proper hardware is found. The Queue Enforcement option will only be visible if OpenPBS integration has been installed.

	Menu Item	Description
Universe XE	Console	Opens a console window on the selected nodes
	Reboot	Initiates software reboot at all selected node (confirmation is needed)
	Shutdown	Initiate software shutdown for all selected nodes (confirmation in needed)
Universe XE	Power	Opens the "Power" submenu. This contains options to perform a hardware power Off, power On or power Cycle on all selected nodes.
Universe XE	Queue enforcement	Opens the Queue enforcement submenu to set the queue enforcement status. Available options are Enabled or Disabled.

Table 5-4: Management menu options

5.3.6 Software installation - Software menu

When logged into the Scali system as root you will get access to the Software menu. This is really the Scali Universe graphical interface for the underlying ScaPkg software installation system (described in detail in appendix E). From this software menu you may install and maintain software packages across all nodes in a Scali System. Table 5-5 gives a summary of the Software menu options.

Menu Item	Description
Install...	Opens the software installation window

Table 5-5: Software menu options

5.3.6.1 Software Installation Window.

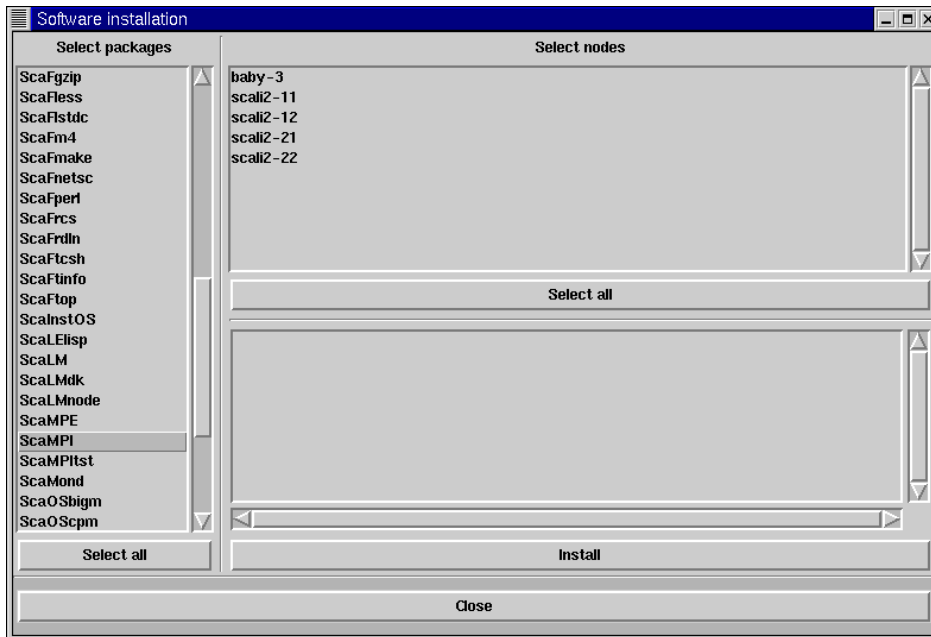


Figure 5-15: Software installation window

The software installations window provides the following options. The **Select packages** scrolled list at the left edge of the window lists possible packages to install on the System. You may select all packages with the **Select All** button. Alternatively packages may be toggled between selected and unselected state by *<Ctrl>+left-click* (hold the Ctrl key down while clicking with the left mouse button).

Similarly, the **Select nodes** list to the right specifies which nodes to participate in the software installation. When software packages and nodes have been selected you may start it all at ion with the **Install** button.

Chapter 6

System Monitoring

Aimed at all users of Scali Systems this chapter explains how to use and configure the Scali cluster monitoring system: ScaMon. Except for configuration ScaMon is almost exclusively used through the graphical environment in Scali Universe.

6.1 Overview

ScaMon provides flexible and powerful monitoring of clustered computer systems including user-definable alarms on selected monitoring events. Based on the industry standard SNMP a number of system and interconnect parameters may be monitored. The graphical monitoring plug-in for Scali Universe provides a number of presentation options ranging from simple “xload” style 2D history graphs to accelerated 3D OpenGL graphics.

6.1.1 Architecture.

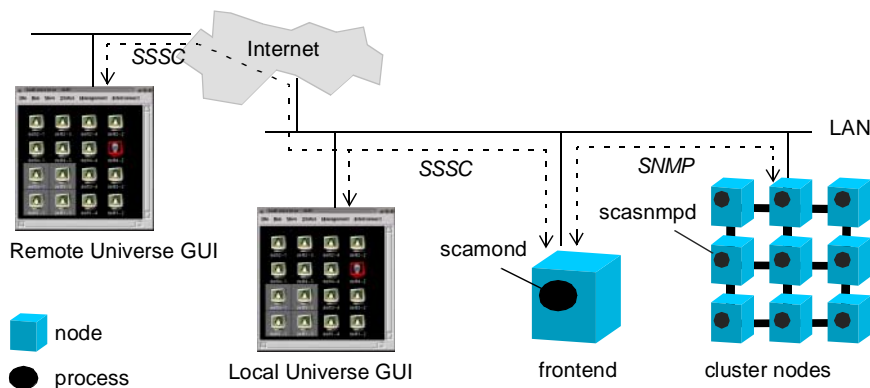


Figure 6-1: ScaMon components in context

The ScaMon system consists of several software components: The Scali SNMP daemon **scasnmppd**, must run on every node to be monitored. The Scali monitoring daemon **scamond**, must run on the frontend of the cluster. Scamond allows clients to subscribe to monitoring variables which will be extracted over SNMP from the nodes at regular intervals. Clients, local or remote, uses SSSC (Scali Secure Socket Communication) to communicate with **scamond** and can therefore safely monitor the system over the Internet.

6.2 Using monitoring from Scali Universe

6.2.1 The Status menu

The ScaMon plugin for Scali Universe adds the Status menu to the MainWindow. This is the main entry point to monitoring functionality. An overview of the Status menu is given in table 6-1

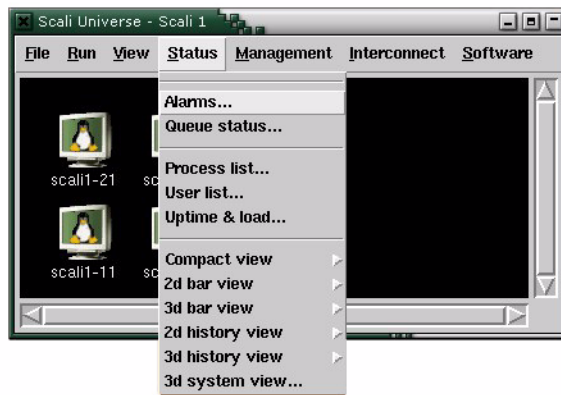


Figure 6-2: The Status Menu

Note that the Status menu is arranged as three logical groups. From the top you have access to monitoring subsystems like **Alarms...** or the monitoring functionality of other subsystems like **Queue Status...**. Then there is a group for commands which all will present some possibly processed output of commonly used shell commands in a window, like **Uptime & load...**. The last “view” group is all related to graphical presentations of monitoring variables, like the **2d bar view**.

Common to all the graphical monitoring functions is that you will first have to select a view (presentation) and then the monitoring variable to present. Monitoring variable selection is available from a submenu for each of the view selections. Every view has the same variable selection submenu which corresponds to the list of variables defined for the scali monitoring daemon **scamond**. This list is in turn customizable as we will explain in “6.7 Defining new monitoring variables”.

Menu Item	Description
Universe XE Alarms ...	Brings up the alarm control window, How to use alarms is described in section “6.4 Using Alarms”
Universe XE Queue Status ...	Brings up the queue status monitoring window, this is described in detail in section “8.8.2 Monitoring jobs from Scali Universe .”
Process list...	Shows processes on selected nodes.
User list...	Lists users on selected nodes.
Uptime & load...	Shows uptime and load for selected nodes (rup command).
Compact view	Brings up a compact bar graph presentation of the selected monitoring parameter for all nodes.
2d bar view	Brings up a standard 2D bar graph presentation of the selected monitoring parameter.
3d bar view	Brings up a 3D bar graph presentation of the selected monitoring parameter.
2d history view	Brings up a two dimensional (xload type) presentation of the selected monitoring parameter.
3d history view	Brings up a 3D history graph (carpet graph) of the selected monitoring parameter.
3D system view	Brings up a 3D compound view of the system.

Table 6-1: The Status Menu

6.3 Graphical monitoring

As explained earlier each of the monitoring variables made available from the monitoring daemon may be combined with any suitable graphical presentation. The different graphical presentations available in Scali Universe each have different strengths which will be highlighted in the following sections.

6.3.1 Common functionality

6.3.1.1 Pop-up menu

Common to all presentation views is a pop-up menu with options for the presentation. The pop-up menu is activated with a right-click in the window. Menu items are explained in table 6-2. Note that not all options are available with every view.

Menu Item	Description
Large Icons	Selects the default “Large Icons” view of the presentation, This often contains more textual information than the “Small Icons” view.
Small Icons	Selects the space saving “Small Icons” view of the presentation. Small icons allows for compact monitoring, but has less detailed information.
Log to file...	Enables you to select a file to which the monitoring data is logged. The logging interval is the same as the update interval specified in the monitoring options.
Close	Close the window

Table 6-2: Monitoring view window options menu

6.3.1.2 3D view camera controls

All the 3D presentation view windows have scroll bars right and bottom to control the camera angle for the presentation. The right scrollbar controls the elevation (up/down) while the bottom scrollbar controls rotation (left/right). In addition we have the mouse + key-bindings as explained below (zoom is only available as a mouse-binding).

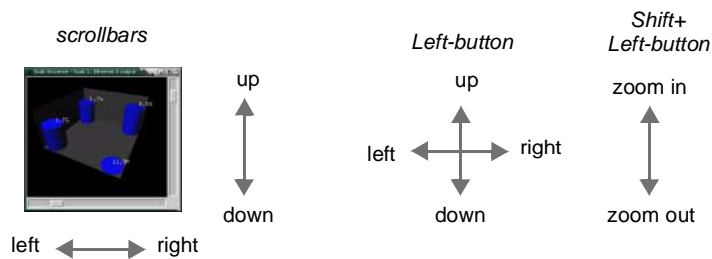


Figure 6-3: 3D view camera controls

6.3.2 Compact View

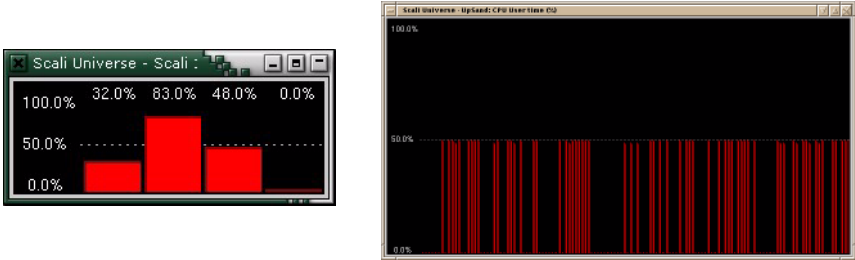


Figure 6-4: Compact view examples with 4 and 132 nodes.

Due to its flexible resizing behaviour, the compact view is great both for monitoring large systems and for making icon-sized monitors for your desktop. The compact view provides dynamic scaling and labelling which means labels are removed when the window is resized below a certain limit. Still, by moving the mouse pointer over one of the value-bars you will get exact readout in the form of a pop-up label with the node name and value of the monitoring variable.

6.3.3 Two dimensional 2D bar view

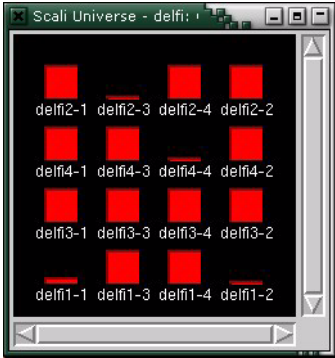


Figure 6-5: 2D Bar small-icons view of system load

The 2D bar view presents monitoring variables as classic bar-graphs with a node layout similar to the node placement in the MainWindow. Numeric values are presented above each bar for exact readout. This view also supports a “small-icons” view which is great on larger systems.

Chapter 6: System Monitoring

6.3.4 3D bar View

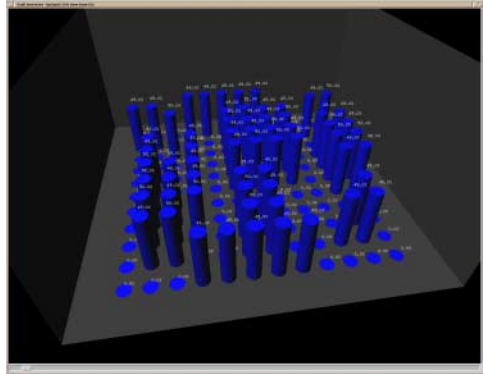


Figure 6-6: 3D bar view on a large system (132 nodes)

The 3D bar view is a translation of the 2D bar view into three dimensions. Node layout is similar to that of the MainWindow with a 3D bar in each node position. The numeric value of the measured variable is “floating” above the bar-representation. The 3D bar view supports all 3D view camera controls.

6.3.5 2D history view

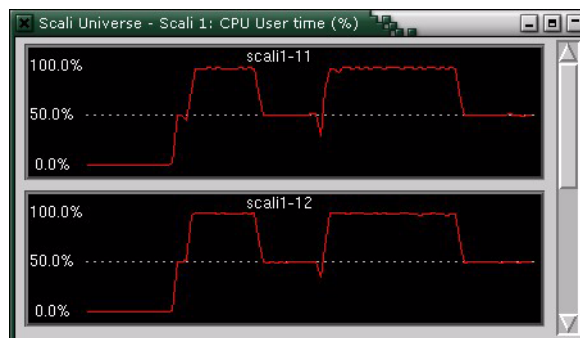


Figure 6-7: 2D history view showing two of four nodes

The 2D history view is very similar to the classic Xload program and enables you to easily spot deviating values within the history time frame with precise readout. Here one graph per node is arranged in a larger scrolled window. The 2D history view scales dynamically on the value axis.

6.3.6 3D History View

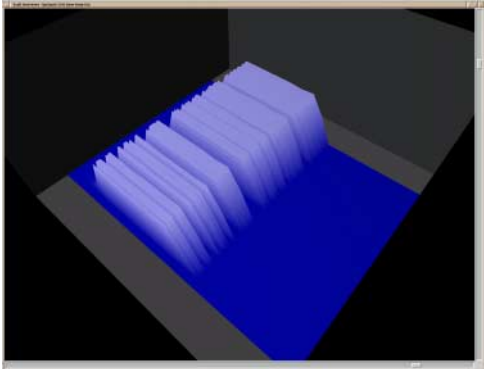


Figure 6-8: 3D History view

The 3D history diagram resembles a moving mountain range and enables you to easily spot deviating values anywhere in the system within the history time-span that is visualized. Increasing values of the monitoring variable is shown as increasing height. Nodes are laid out along one axis while time runs along the other. The 3D history view supports all 3D camera controls.

6.3.7 3D System View - compound view

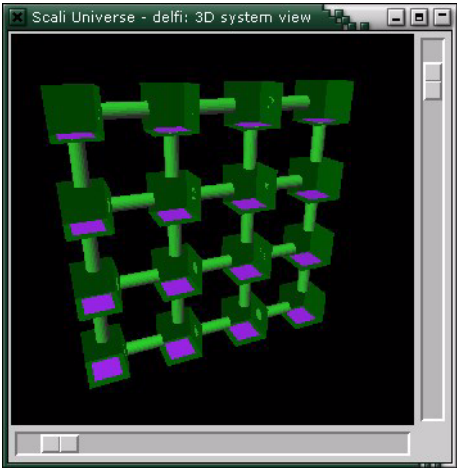


Figure 6-9: 3D System view on a 2D SCI system

Chapter 6: System Monitoring

Especially on SCI systems the 3D system view will provide you with a more useful information in a single compound view. The nodes can selectively be shown as plain colour coded boxes, or as a “platform” including a 3D bar graph for monitoring selected values. The layout of the nodes in 3D space tells you the topology of the system. If an SCI network is present this will be shown as thick “pipes” between the nodes. These pipes will change colour according to the state of the SCI link following the same colour code as described in “Link status graphics explained” on page 98. The layout of the nodes in 3D space tells you the topology of the system. The 3D System View supports all 3D camera controls

Universe XE 6.4 Using Alarms

Scali user definable alarms are part of the monitoring system available in professional versions of the software. You may define your own events to trigger an alarm based on any combination of comparative operations of any available monitoring variable. When the alarm is triggered you can select between a set of predefined actions to be performed. Combined with the possibility to define your own monitoring variables this makes for an extremely flexible and powerful solution.

6.4.1 The Alarm window.

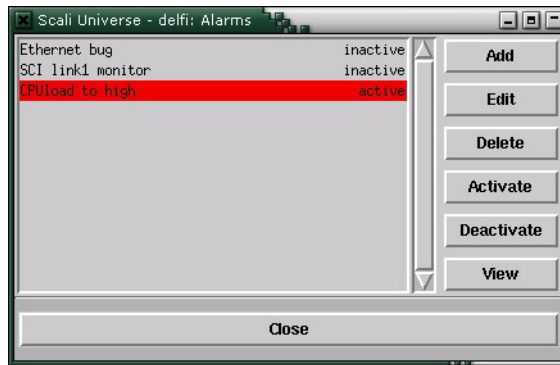


Figure 6-10: The Alarm window

You open the alarm main window by selecting **Alarms ...** from the Status menu. The Alarm window contains a scrolled list over current alarms with status information and a row of buttons to perform operations on selected alarms along the right edge. The function of the different buttons is pretty self-explanatory, but a summary can be found in table 6-3. Apart from **Add**, all other buttons requires that an alarm is selected

first in order to work properly. When an alarm is triggered it will change to red background colour in the alarm list. This is the case for alarm “CPUload too high” in figure 6-10.

Button	Description
Add	Brings up the alarm editor window to create a new alarm.
Edit	Brings up the alarm editor window to edit the selected alarm. Requires an existing alarm.
Delete	Deletes the selected alarm.
Activate	Activates (enables) the selected alarm
Deactivate	Deactivates (disables) the selected alarm
View	Brings up the alarm view window which shows a dynamically updated log of the selected alarm.

Table 6-3: Alarm Main window control summary

6.4.2 The Alarm Editor.

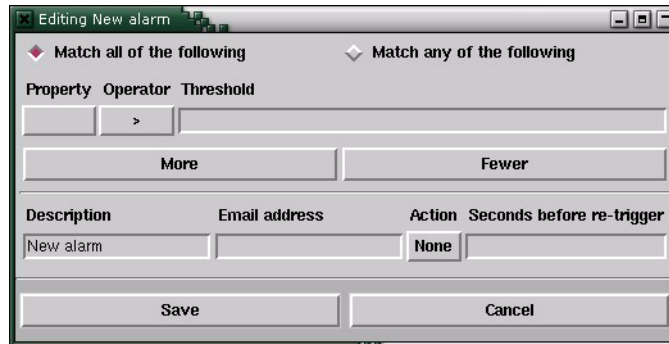


Figure 6-11: The alarm editor window

The alarm editor is used both for defining new alarms and editing existing ones. Hence it is activated by pressing either **Add** or **Edit** in the alarm main window. The model we’ve used to describe a condition that should set off an alarm is to combine a series of

Chapter 6: System Monitoring

boolean expressions using either AND or OR logical functions. The boolean expressions are constructed by comparing a monitoring variable to a reference value by means a logic operator (<, <=, >, >=, !=, ==).

As a response to the alarm being triggered, you can select between different actions to perform in addition to sending an E-mail. Beware the field called *Seconds before re-trigger*. This is default set to “-1” which means that the alarm should not be triggered again after the first time (a one-shot). You may receive *a lot* of alarms if this value is set to low.

Window item	Description
Match all of the following	Use logical AND between the monitoring variable comparisons below. Excludes Match any...
Match any of the following	Use logical OR between the monitoring variable comparisons below. Excludes Match all...
Property	Option menu to select a monitoring variable for comparison.
Operator	Option menu to select operator for comparison <, <=, >, >=, ==, or != (C syntax).
Threshold	Threshold value to compare against. Uses default unit of selected monitoring variable, e.g. percent for CPU load, rpm for fan speed ...
More	Adds one more monitor variable comparison.
Fewer	Removes one monitor variable comparison.
Description	Name of alarm.
Email address	Where to send the alarm notification
Action	What to do when the alarm is triggered: Either None, Reboot Machine or Shutdown Machine
Seconds before re-trigger	Defined the minimum interval (in seconds) after an alarm has been triggered till it can be triggered again. The default value is -1 which means no re-trigger.

Table 6-4: Alarm Editor control overview

6.4.3 Alarm log viewer

The View button in the alarm main window will open the log viewer window for the selected alarm. This will show a dynamically updated list of events that has triggered the alarm, listing exact time of day and actual parameter values. You may clear the window using the Reset button (no undo).

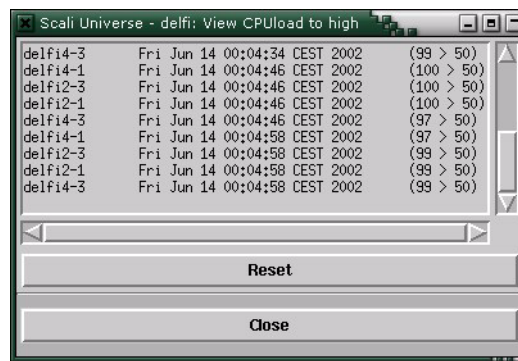


Figure 6-12: The alarm log viewer

6.4.4 Example: Defining a new alarm

As a simple example we will define the alarm called “CPUload too high” which will send us an E-mail if user CPU load passes 50%. To define a new alarm, press the Add button in the alarm main window. The alarm editor appears and we can start defining the values (fig 6-13).

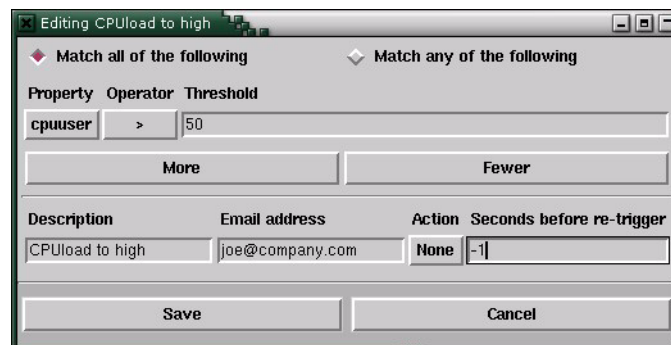


Figure 6-13: Adding a new alarm

Chapter 6: System Monitoring

First select “cpuuser” from the Property menu, then set the Operator to “>”, and the Threshold to “50” (%). Type the name of the alarm in the Description field and your E-mail address in the field next to it. Set Action to “none”, and Seconds to re-trigger to 10. Save your settings by pressing the Save button.

The alarm has now been defined and will show up in the Alarm main window. Note that a newly defined alarm will always be set to *inactive*. In order to activate it for use you must select it and press the Activate button. Now, to test the new alarm, running any reasonably demanding program should push CPU load beyond 50. If you keep the alarm main window open you can see how the alarm background changes to red as the alarm is triggered. Looking in your mailbox you could expect to find a mail similar to this:

```
Subject: Scali alarm reporter
Date: Wed, 12 Jun 2002 16:55:38 +0200
From: root <root@cmpq4-11.scali.com>
To: joe@company.com
```

DO NOT REPLY TO THIS MESSAGE

This is an automated message to inform you that the following alarm was triggered on Linux cluster cmpq4-11.scali.com.

The expression for this alarm is :
“{cpu_user} > {50 }”

The “CPUload too high” alarm was triggered on Wed Jun 12 16:55:38 CEST 2002 on the following nodes :

```
cmpq4-11 (62.0 > 50)
cmpq4-12 (62.0 > 50)
```

This alarm has no valid action.

6.5 The monitoring daemon: scamond

The Scali monitoring daemon: **scamond** provides filtered monitoring values to the clients. This means that two clients requires the same parameter only one SNMP request is sent. Hence both the network load and CPU load from the **scasnmpd** is reduced.

6.5.1 Manual start/stop

Although **scamond** will be installed and started by the SSP install script it may be started, stopped or restarted with the command below should it be necessary:


```
# /opt/scali/init.d/scasmpd [start|stop|restart|info]
```

6.5.2 Configuration file

The scamond configuration file defines a subset of monitoring variables that will be available for monitoring by other applications. It is located in

```
/opt/scali/etc/ScaMond.conf
```

Due to its “wide” format this configuration file is not very well suited for printing so if you’re interested we suggest that you view the file on-line using `less` (or `more`).

6.5.3 Default monitoring variables

The predefined monitoring variables which occur in the default “Status” menu represents only a handful (though probably the most interesting) system parameters to monitor. As we shall see in “6.7 Defining new monitoring variables” this is only a small subset of the available parameters and can be easily extended and customised. The default monitoring parameters are arranged in two major subgroups Performance and System as described in tables 6-5 and 6-6

Menu Item	size	maxval
CPU Idle Time (%)	percent	100
CPU System time (%)	percent	100
CPU User time (%)	percent	100
Context switches/s	human	1024
I/O in (blocks/s)	human	1024
I/O out (blocks/s)	human	1024
Interrupts/s	integer	200
Swap in (kB/s)	human	1024

Table 6-5: Default “Performance” monitoring variables

Menu Item	size	maxval
Swap out (kB/s)	human	1024
Ethernet 0 input (octets)	human	1024
Ethernet 0 output (octets)	human	1024

Table 6-5: Default “Performance” monitoring variables

Menu Item	size	maxval
Disk: Free	percent	100
Disk: Used	percent	100
Memory: Real Free (%)	percent	100
Memory: Real Free (kB)	human	134217728
Memory: Real Used (%)	percent	100
Memory: Real Used (kB)	human	134217728
Memory: Swap Free (%)	percent	100
Memory: Swap Free (kB)	human	134217728
Memory: Swap Used (%)	percent	100
Memory: Swap Used (kB)	human	134217728

Table 6-6: Default “System” monitoring variables

6.6 The SNMP daemon: `scasmpd`

Also part of the monitoring system, the Scali SNMP daemon: `scasmpd` is based upon the UCD SNMP package [18]. The daemon has been extended by Scali with monitoring and configuration variables (OIDs) for Dolphin PCI-SCI adapters and various other host specific variables.

The Scali SNMP daemon is configured to not interfere with existing SNMP daemons by using a non-standard SNMP port. The default port used by `scasmpd` is: 32016

6.6.1 Manual start/stop

Although **scasmpd** will be installed and started by the SSP install script it may be started, stopped or restarted with the command below should it be necessary:

```
# /opt/scali/init.d/scasmpd [start|stop|restart|info]
```

6.7 Defining new monitoring variables

The monitoring capabilities in the Scali Universe is easily extended to monitor other SNMP variables than the default set. It's necessary with some knowledge of SNMP to create the necessary configuration statements. You can here create a monitoring value based on a function off many SNMP OIDs, like sums, ratios, etc.

6.7.1 Obtaining the SNMP OIDs

First of all you need to know the SNMP Object ID you want to monitor. We recommend that you have the **snmpget** and **snmpwalk** command available (on Red Hat systems they're located in the `ucb-snmp-utils` or `net-snmp-utils` package). You should test that your SNMP variables are available in the SNMP daemon with the **snmpget** or **snmpwalk** command like this:

```
% snmpwalk -p 32016 localhost birka host.1
host.hrSystem.hrSystemUptime.0 = Timeticks: (310235448) 35 days,
21:45:54.48
host.hrSystem.hrSystemDate.0 = 2002-6-13,12:0:5.0,+2:0
host.hrSystem.hrSystemInitialLoadDevice.0 = 1536
host.hrSystem.hrSystemInitialLoadParameters.0 = "auto BOOT_IMAGE=linux2417
ro root=4806 BOOT_FILE=/boot/vmlinuz-2.4.17-2smp console=ttyS0."
host.hrSystem.hrSystemNumUsers.0 = Gauge32: 4
host.hrSystem.hrSystemProcesses.0 = Gauge32: 131
host.hrSystem.hrSystemMaxProcesses.0 = 0
```

Or if you prefer the numeric OID's.

```
% snmpwalk -On -p 32016 localhost birka host.1
.1.3.6.1.2.1.25.1.1.0 = Timeticks: (310235107) 35 days, 21:45:51.07
.1.3.6.1.2.1.25.1.2.0 = 2002-6-13,12:0:1.0,+2:0
.1.3.6.1.2.1.25.1.3.0 = 1536
.1.3.6.1.2.1.25.1.4.0 = "auto BOOT_IMAGE=linux2417 ro root=4806
BOOT_FILE=/boot/vmlinuz-2.4.17-2smp console=ttyS0."
.1.3.6.1.2.1.25.1.5.0 = Gauge32: 4
.1.3.6.1.2.1.25.1.6.0 = Gauge32: 133
.1.3.6.1.2.1.25.1.7.0 = 0
```

Chapter 6: System Monitoring

This example queries the Scali SNMP daemon (`scasnmppd`) which runs on a separate UDP port (32016) and on the community `birka`. You're likely to use a separate SNMP daemon that has a different port and community. The regular community tend to be "public". As a hint, many SNMP daemons only support SNMP version 1 and you'll need to give the `-v 1` option to `snmpget`.

Once you are able to read the SNMP variables (OIDs) you wish to monitor, make a note of the options to `snmpget`, you'll need them in later when editing the config file.

6.7.2 Editing the ScaMon configuration file

On the frontend node there is a file `/opt/scali/etc/ScaMond.conf` where all the necessary configuration is done. This file has a number of sections where as four are of interest and will require some editing.

6.7.2.1 Editing the class definition

The first section of interest is the class definition:

```
#class name type format cmdline
class scasnmpp internal {%s %s = %[^]} {snmpaget -p 32016 %s birka %s}
```

Where there is a `scasnmpp` class for the Scali SNMP daemon. The `snmpaget` is an internal `snmpget` that does a parallel query to many nodes, but it takes the same arguments as the command `snmpget`, see `man snmpcmd` (part of the `ucb-snmp-tools` package) for a full description. To add a class for a default SNMP daemon you need to add a line like this:

```
class stdsnmp internal {%s %s = %[^]} {snmpaget -v 1 -p 161 %s public %s}
```

The two `%s` are replaced with host name and OID respectively.

6.7.2.2 Editing hwgroup classes

The second section of interest is the `hwgroup` class, with is actually optional. If the SNMP OIDs you want to monitor only appear on a selection of you nodes (ie: you've a heterogeneous cluster) you can define them into a group like this:

```
#hwgroup name members
hwgroup myset {node23 node44}
```

Where we put the two nodes into a group.

6.7.2.3 Adding the OIDs

The third section in where you define the OIDs you need to retrieve:

```
#oid hwgroup name size class format oid
oid {} cpuidle integer scasmp %d
enterprises.ucdavis.systemStats.ssCpuIdle.0
oid {} cpusystem integer scasmp %d
enterprises.ucdavis.systemStats.ssCpuSystem.0
oid {} cpuuser integer scasmp %d
enterprises.ucdavis.systemStats.ssCpuUser.0
```

If the `hwgroup` is empty like here (`{}`) it just means that you will monitor on all nodes in `/etc/scali/etc/ScASH.scahosts`. The `name` is mapped to a variable in the next section. The `size` is almost always integer, but occasionally strings are used to measure up/down values. The `class` refers to a class that must be defined in the class section, as we talked about in 6.7.2.1. The `format` is usually `%d` for integer, and `{${^1}}` for strings.

Note: some OIDs returns the data-type with the value like "Gauge32: 42" or "Counter32: 42", see the `snmpwalk` example in 6.7.1. In this case the data-type *must* be a part of the format like this: `{Gauge32: %d}`.

Then finally enter the `oid`, either numerically, or symbolically if there is a MIB for you variables in `/opt/scali/share/snmp/mibs/`. Remember that you can always mix the two formats. It helps readability to use the prefixes `host` for `.1.3.6.1.2.1.25`, `system` for `.1.3.6.1.2.1.1`, and `enterprises` for `.1.3.6.1.4.1`. To add an `oid` entry for one of the OIDs listed in the `snmpwalk` example using symbolic OIDs add the following lines:

```
oid {} process_count integer scasmp {Gauge32: %d}
host.hrSystem.hrSystemProcesses.0
oid {} user_count integer scasmp {Gauge32: %d}
host.hrSystem.hrSystemNumUsers.0
```

Or alternately with numeric OIDs:

```
oid {} process_count integer scasmp {Gauge32: %d} .1.3.6.1.2.1.25.1.6.0
oid {} process_count integer scasmp {Gauge32: %d} host.1.6.0
```

All three `process_count` definitions are equal.

6.7.2.4 Adding the variable definition for Universe

In the final section you need to define the variable as it appears in Scali Universe. The definition takes the form:

```
#variable descr size maxvalue expression
variable {Performance "CPU Idle time (%)" } percent 100 {$cpuidle}
```

After the keyword `variable` you must enter a descriptive string which will show up in the monitoring variable menu in Scali Universe. If you want your newly defined variable to show up in one of the submenus you must define the entire menu path separated by white-space within the `descr` field. In the example above the "CPU Idle time (%)" label is added to the "Performance" submenu (the very one you see by default in the Scali Universe GUI). If you want to use space within the menu label, you must double quote the string like you've seen here.

The `size` field is used to tell about the scaling of the graph. "human", "percent" and "integer" all expect integer value from the expression, but percent will always scale the Y axis from 0 to 100. The `maxvalue` is really just advisory. In the `expression` we can give an expression involving many oids:

```
variable {Performance "Processes per user" } integer 10 {$process_count /
$user_count}
```

Here we just defined a "Processes per user" monitoring variable based on the two oid's we defined above.

Note: all mathematical operators `+*/` and the parenthesis `()` *must* be surrounded by space! Hence, an expression like the following will silently be ignored:
`{ $process_count/ $user_count }`

You may notice that in the alarm section you may now define alarms based on your new oids.

Note: After you change the `ScaMond.conf` file you must restart the `scamond` service for the changes to take effect. This must be done as root with the command:

```
# service scamond restart
```

Chapter 7 Interconnect configuration

Aimed primarily at system administrators this chapter describes organisation of and how to use the Scali configuration and management system for high speed interconnects: ScaConf.

7.1 Overview

Currently supporting SCI networks of different topologies ScaConf provides automatic configuration of SCI networks for maximum connectivity and utilization. With distribution/check of SCI node-IDs and routing as primary functions, ScaConf is strictly necessary in order to make any but the smallest SCI network operational. Advanced routing algorithms developed at Scali (see “Routing” on page 107), allows ScaConf to bypass failed nodes with the least possible negative impact on total system performance. The ScaConf system has been designed for automatic operation, but users with ‘root’ privileges still have access to management interfaces both through the Scali Universe GUI and the ASCII based scaconftool.

7.1.1 Architecture

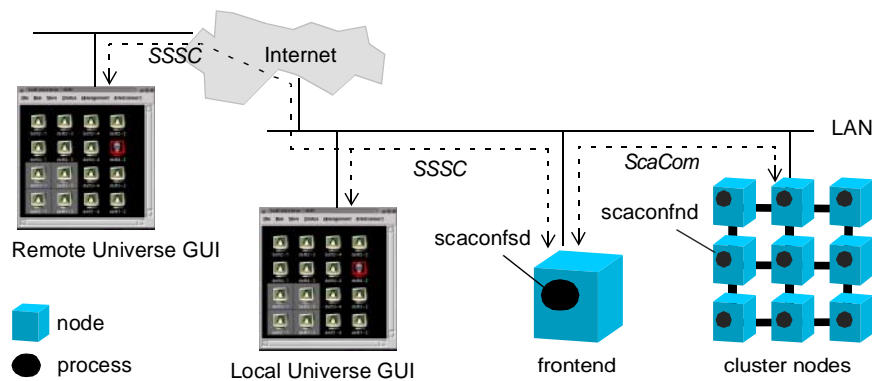


Figure 7-1: ScaConf components in context

The ScaConf system consists of several software components: The Scali interconnect configuration node daemon: **scaconfnd** provides an interface between the system and the underlying (SCI) interconnect and must run on every node to be managed. The

Chapter 7: Interconnect configuration

Scali interconnect configuration server daemon **scaconfsd** is the component that possesses the “intelligence” in the system. It must run on the frontend of the cluster. For communication between the node daemons and the server daemon, ScaConf uses the ScaCom communication backbone, this also implies that the Scali communication daemon, **scacomd** must also run on every cluster node.

Scali Universe Clients, local or remote, uses SSSC (Scali Secure Socket Communication) to communicate with **scaconfsd** and can therefore safely configure the cluster system over the Internet. In addition there is a text based interface to the server, called **scaconftool** which provides the most complete management interface to the configuration system.

7.2 SCI configuration from Scali Universe

When managing systems with SCI the ScaConf plugin for Scali Universe will add an “Interconnect” menu to the main window. The Interconnect menu enables you some control over the routing of the high speed SCI network of the system.

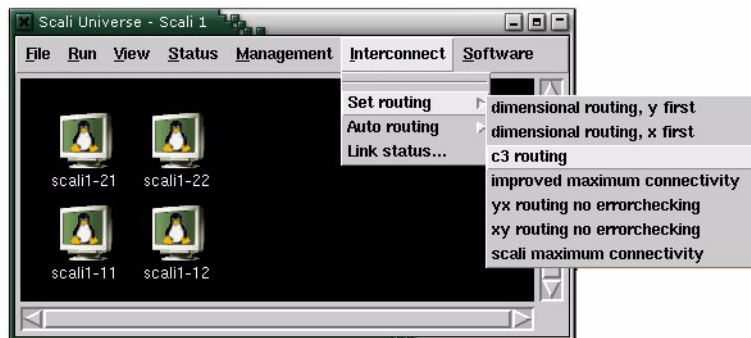


Figure 7-2: Interconnect menu added to the Main Window

7.2.1 The Interconnect menu

The Interconnect menu has the following options:

Menu Item	Description
Set routing	Opens the routingtype submenu which enables you to change the routing type for the system. The available routing types will depend on SCI topology (and license).Please refer to section "7.4 Routing".
Auto routing	Brings up a submenu which allows you to enable/disable automatic rerouting. The default value is "ON".
Link Status ...	Opens the link status window.

Table 7-1: The Interconnect menu items

7.2.2 Link Status Window - interconnect monitoring

The Link status window shows the status of all the SCI links in the network. This is really an SCI specific monitoring view placed in the Interconnect menu. Links are grouped per node and arranged in a 2D row-column fashion with a node placement similar to that of the "Large Icons" view in the MainWindow (figure 7-3):.

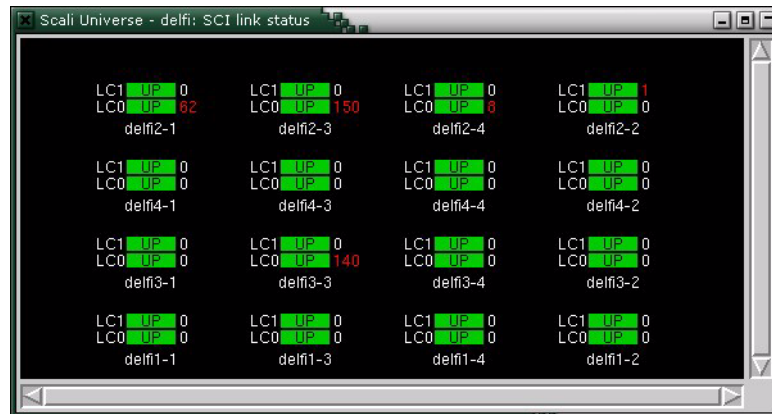


Figure 7-3: High speed SCI network link status window.

7.2.2.1 SCI link status graphics

Within the Links Status window, the SCI links for every node are displayed as a stack of virtual “LED indicators” as shown in figure 7-4. The most important information lies in the colour of the LED indicator. Green means “up and enabled” and represents the OK state for a link. You really want to see all green LEDs. If an SCI link goes down, ScaConf will disable it (and possibly some others) as a result of the new routing which is applied to avoid the problem. Disabled links show up as yellow LED indicators. Your system may be fully operational even with some disabled links, but you should try to resolve the cause of the problem as the redundancy of the system decreases with every failed component.

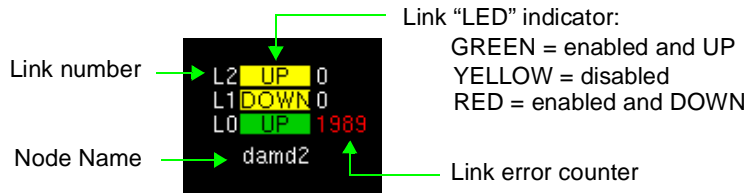


Figure 7-4: Link status graphics explained

Finally, a red LED indicates that the link is “down and enabled”. As a permanent state this represents a really bad error situation which has not (yet) been handled by ScaConf. Normally you’ll only see the red LEDs as a transitional state. The “Link error counter”, positioned to the right of each LED indicator also gives important information about the state of individual SCI links. High numbers of SCI link error may be a sign of a bad or loose SCI cable. For counts higher than 0, the number changes colour from white to red. You may reset the link error counters from the SCI link status window pop-up menu.

7.2.2.2 SCI link status window options.

The link status window provides a pop-up menu for setting options. Except for the Reset Counters option, these are standard options for any monitoring window. Right-click inside the window to bring up the menu.

Menu Item	Description
Large Icons	Select the default view with large icons and information as described above
Small Icons	Select compact view with only status “LED” indicators and node names, this allows for compact monitoring.
Reset Counters	Will reset the link error counters for all links, but only for this instance of the link status window.
Close	Close the window

Table 7-2: SCI link status window pop-up menu options

Note that resetting link error counters only affects this instance of the link status window. You will not interfere with other people monitoring the same system.

7.3 Command line interface - scaconftool

The **scaconftool** ASCII based command line interface is a fast and efficient way to access the ScaConf functionality. An on-line plain text manual for **scaconftool** may be found in the file `/opt/scali/doc/ScaConfC/scaconf.users.instruction.txt`. You may want to check out this for last-minute information which may not have made it into the manual.

7.3.1 Starting scaconftool

When installed, **scaconftool** is located in the `/sbin` directory of the Scali installation tree and can be started on the frontend by:

```
frontend% /opt/scali/sbin/scaconftool
```

To start the tool on any other node than the frontend, the hostname of the frontend must be included with the `-H` switch:

```
bash% /opt/scali/sbin/scaconftool -H <frontend>
```

If **scaconftool** fails to start or exits abruptly during run, chances are that it has lost the connection to `scaconfsd`. In that case, you need to check that the server is running, and if not, restart the server and start a new tool.

7.3.2 Using scaconftool

The following command line options are available with **scaconftool**:

Usage:

```
/opt/scali/sbin/scaconftool [Options]
```

[Options]:

```
-l d/e          enable (e) or disable (d) log from server.
-e "cmd1;...;cmdn;"  execute these commands in batch mode.
-h/?           help
-H <host>      set the server host.
-p <port>      set the server port (default is 32015).
-q            make scaconftool as quiet as possible
-s <number>    set the sleep constant, usually a number
               between 0 and 1 (default is 0.7).
-v            print version info.
```

When the tool has started, it will prompt you with `CONFTOOL>` to indicate that it is ready to accept commands. A reference to commands and their syntax can be found in the file `/opt/scali/doc/ScaConfC/scaconftool.txt`. Command line editing may be performed in “readline style”, where the arrow-key gives access to a history of previous commands.

7.3.2.1 User modes.

Depending on the privileges of the user, scaconftool will start in either user mode or administrator (admin) mode. The admin mode is available only to users with administrator rights (i.e. user id = 0). User identity is checked upon start and the mode for the configuration tool is set according to this information. Admin mode gives full access to ScaConf, while user mode only gives access to status information.

- **User mode:** For ordinary users there's only a restricted set of commands available: *getopt*, *list*, *log*, *reconnect*, *select*, *status*, *unselect*, *update*, *version*, *help* and *quit*.
- **Admin mode:** Admin mode is only available to root, and gives full access to all ScaConf operations like rerouting, starting/stopping daemons etc.

7.3.2.2 Command truncation

The commands and some of the keywords are also recognized if you truncate them, as long as enough of the command is present to be unambiguous. Node states, node names and name of routing algorithms may not be truncated. `lis no st a` is equal to typing `list nodeid status all`.

7.3.2.3 Node selection

Some of the scaconftool commands allows you to operate on a set of nodes. There are five different ways to select nodes. The *list* command will be used as an example to illustrate these possibilities:

1. *Explicit node names:* You may explicitly specify the node names on the command line separated by one or more blanks. Example: `list node1 node3 node14` will list information about node1, node3 and node14.
2. *All nodes:* The keyword 'all' is an alias for all nodes in the system. The command '`list all`' will list to all nodes known to the configuration server.
3. *By node state:* It is possible to select a set of all nodes that is in one particular state. Use the '`list status all`' command to see which state the nodes are in. `list NO_SCI_DEV` will list information on all nodes with status *NO_SCI_DEV*.
4. *By routing partition number:* You may select all nodes part of a particular routing partition. `list status #1` will show status for all nodes in partition 1. Partition number is prefixed with #.
5. *Internal list:* scaconftool maintains an internal list of nodes which is used no other node selection mechanisms are provided. The list is empty when the tool is started. Nodes may be added to the list with the `select` command and removed with the `unselect` command (ref. "D-2.7 ConfTool> select"). As an example you may use this to get a list of all nodes except those in state *OK*. First type `select all` followed by `unselect ok`. The `list` command without parameters will now list all nodes except those in state *OK*.

7.3.2.4 Disabling server log messages

The default setting for scaconftool is that log messages from server activity are shown directly in scaconftool with a timestamp like this:

```
ConfTool>
11:51:57 WARNING: Lost connection to node daemon on: scali1-21
11:52:05 NOTICE : Automatic reroute, routing type is: SCA_ROUTE_MAXCY
11:52:05 NOTICE : System rerouted with maxcy routing algorithm.
```

These messages generally helpful, but if you find them annoying they can be turned off by the `log` command. To disable the server log messages type: `log disable`. To enable server log messages again type: `log enable`. You may also enable or disable log messages from start with the `-l` command line option to scaconftool.

Chapter 7: Interconnect configuration

7.3.2.5 Batch mode

To run `scaconftool` in batch mode, use the `-e` option when starting the tool. The commands you want `scaconftool` to run must be enclosed by double quotes " " and separated by ';'. The following batch-job will restart the configuration daemon on nodes in state `NO_DAEMON`:

```
frontend% scaconftool -e "daemon scaconfnd restart NO_DAEMON"
```

`scaconftool` will first execute `daemon scaconfnd restart NO_DAEMON` command. This will launch a **scash** job to restart **scaconfnd** on all nodes in state `NO_DAEMON`.

7.3.2.6 On-line help

The `help` command provides an on-line reference to `scaconftool` commands. To get help on a specific command, just type `help <command>`, i.e. to get help on the `list` command type `help list`. The `help` command gets its input from a plain text file located in: `/opt/scali/doc/ScaConf/scaconftool.txt`. This file is formatted for on-line reading and can be viewed with any text file viewer like `more(1)` or `less(1)`.

7.3.3 Status check of nodes and daemons

Normally the first thing you want to do is to check the status of all the nodes in your system. To help `scaconftool` provides the `list` and `status` commands. Here is an example using `list` with the options `nodeid status all` which will show SCI node identification and status for all nodes:

```
ConfTool> list nodeid status all
Name      NodeId  Status
scali2-24 0x2400  OK
scali2-23 0x2300  OK
scali2-22 0x2200  OK
scali2-21 0x2100  OK
scali2-14 0x1400  OK
scali2-13 0x1300  OK
scali2-12 0x1200  OK
scali2-11 0x1100  UNREACHABLE
```

Have a look at the "Status" column, if some of the nodes are not OK state you should investigate this further. Table 7-3. contains a list of all possible node states with descriptions and some hints in how to resolve a possible problem. Try using these hints to get the node back to an OK state. It may be that some daemons are not running and needs restarting. on some of the nodes. The `fix` command described in the next section attempts to do this automatically.

7.3 Command line interface - scaconftool

..

Node state	Description	How to resolve
OK	SCI-driver is loaded, and communication with daemons on this node is ok	This is the state of a healthy node and need not be resolved.
NO_SCI_DEV	The SCI-driver on the node is not loaded	You can reload the driver with the <i>reload</i> command, or use the <i>fix</i> command. If this did not help, the state must be resolved outside scaconftool. Please check that the SCI card is properly fitted in the node. Also make sure that your hardware and/or kernel version is supported by the ScaSCI driver.
NO_DAEMON	The configuration server has lost contact with the node daemon scaconfnd on the node. This usually indicates that the daemon is not running or not connected to the server.	Make sure that the AUTO_RECONNECT and AUTO_REPAIR options are ON. Then you may try the <i>fix</i> command to resolve this state.
UNREACHABLE	The configuration server has lost (or never gained) contact with this node. This may indicate that the communication daemon (sacomd) on this node is not running or that the node itself is erroneous.	Make sure that the AUTO_RECONNECT and AUTO_REPAIR options are ON. If it is a communication failure, this will be resolved with the <i>fix <node></i> command. If the node is still <i>UNREACHABLE</i> after running the <i>fix</i> -command, the node most likely has problems beyond reach of scaconftool.
UNKNOWN	The configuration server has not been able to determine the state of this node.	Make sure that the AUTO_RECONNECT and AUTO_REPAIR options are ON, then you may try the <i>restart</i> command to resolve this state.

Table 7-3: ScaConf node states

7.3.4 The *fix* command

The *fix* command attempts to bring all specified nodes in a system back to an OK state by re-establishing connections and restarting daemons as necessary. The following algorithm is used:

- For all of the nodes that are in state UNREACHABLE:
 - * Attempt to reconnect nodes.
 - * For all nodes still in state UNREACHABLE:
 - * If the node answers to *ping*, restart the communication daemon.
 - * Wait for some time.
 - * Attempt to reconnect the nodes again.
- Then, for all of the nodes which are in state NO_DAEMON:
 - * Attempt to reconnect nodes.
 - * For all nodes still in state NO_DAEMON:
 - * Restart configuration daemon on the node.
 - * Wait for some time.
 - * Attempt to reconnect the nodes again.
- Then, for all of the nodes which are in state NO_SCI_DEV:
 - * Attempt to reconnect nodes.
 - * For all nodes still in state NO_SCI_DEV:
 - * Reload SCI-driver on the node.
 - * Wait for some time.
 - * Attempt to reconnect the node.

It may be necessary to run the *fix* command more than once, if the process of restarting daemons takes a long time due to heavy load on the system. When a daemon is started (this is done with the *scash* utility) *scaconftool* will wait for a given amount of time. By default this is (Number_of_nodes * WaitConst) seconds. WaitConst is 0.7 by default, causing a system with 8 nodes to wait for approx. 6 seconds before it will try to reconnect a node. WaitConst may be tuned when *scaconftool* is started with the *-s* option:

```
frontend> scaconftool -s0.5
```

On a machine with 8 nodes this will cause the tool to wait 4 seconds between restart of a daemon on a node and an attempt to reconnect the node.

7.3.5 Setting SCI nodeID manually

An SCI nodeID is the identification for a node in an SCI network. It is stored in NVRAM in the SCI hardware (card). The configuration server: *scaconfsd*, will set, check and correct SCI nodeID's on all nodes in the system unless this functionality has

been explicitly turned off by setting the `AUTO_NODE_ID` server option to `OFF`. When automatic nodeID checking and setting has been turned off, the `nodeid` command in `scaconftool` allows you to set SCI nodeIDs manually. Do not use this command if you do not know what you are doing. The `nodeid` command is only available in admin mode.

When run without parameters, the `nodeid` command will simply activate the check-and-correct cycle of the `scaconfsd`. This sets nodeIDs sequentially starting at `0x100` and increasing in steps of `0x100`, following the order in which nodes are listed in the configuration file `/opt/scali/etc/ScaConf.conf`. To set the nodeID to a specific value you must specify both node name and the new nodeID like this:

```
ConfTool> nodeid scali2-24 0x2400
```

Make sure this new nodeID is legal. The nodeID must be unique, in the range `0x100` to `0xff00`, and two least significant bytes in the nodeID must always be `00`.

7.3.6 Reloading SCI-driver

When installed, the SCI-driver is always loaded upon reboot of a node. If the SCI-driver somehow needs to be reloaded, this can be done with the `reload` command from within `scaconftool`. The `reload` command is only available to root. Reloading the SCI-driver on a node will clear the routing table on that node leaving the cluster in need of rerouting. This is done automatically unless the `AUTO_REROUTE` server option has been set explicitly to `OFF`.

7.3.7 Access to console on one node

The `console` command may be used to connect to the console on one node. For this command to work, a console switch must be available and the `ScaCons` package must be installed correctly. See `'man console'` for more information. The second argument to the console command is passed directly to the `/opt/scali/bin/console` program.

7.3.8 Hot restart of the configuration server

You may force the configuration server to perform a hot-restart with the `restart` command. A hot-restart means that the server breaks connections to all nodes, clears its configuration database and restarts its configuration thread - all without losing the connections to the clients. As the configuration file is read on startup only, this means that you can make changes to the configuration file take effect "on the fly", without the need for a full restart. A full restart takes significantly longer, and of course - breaks all connections.

7.3.9 Failing nodes

Used mainly for testing purposes, the `fail` command allows you to “soft-fail” a node, regardless of whether the node is alive or not. This will force the node into the UNREACHABLE state in the configuration server database, but it will not affect the node, apart from the fact that this node will be considered failed if the cluster is rerouted. The `fail` command is useful for generating and setting routing patterns that normally only will occur if the node actually has failed. A node failed with the `fail` command may be brought back to its correct state with the `reconnect` or `restart` commands. In order for the soft-failing of a node to last more than a few seconds, the `AUTO_REPAIR` option in the server must be turned off prior to failing the node.

7.3.10 Daemon control with `scaconftool`

The two daemons `scaconfnd` and `sacomd` may be started, stopped or restarted on all nodes from the tool. This is done with the `daemon` command. If you want to stop and start the communication daemon on all nodes, type this in the tool:

```
ConfTool>daemon scacomd restart all
```

If you want to stop and start the configuration daemon on all nodes in state `NO_DAEMON` (no configuration daemon), you type this in the tool:

```
ConfTool>daemon scaconfnd restart NO_DAEMON
```

The `daemon` command uses `scash` to access all nodes, and the `ScaSH` package must be installed on the machine running the tool in order for the `daemon` command to work from the tool. Starting and stopping daemons from the tool is only available in admin-mode.

7.3.11 Setting server options.

From `scaconftool` you may control configuration server options with the `setopt` and `getopt` commands. Please refer to the complete list of available options and their default values in Table 10-9 on page 173. To change an option, use the `setopt` command as shown below (disables automatic rerouting).

```
ConfTool>setopt AUTO_REROUTE OFF
```

Some server options are of type `ACTION`. This means that rather than setting a parameter they will cause the configuration server to perform an action. The action `setopt SAVE` will save your current server options settings to the file `/opt/scali/etc/ScaConf.opt`, from which they will be restored when the server is restarted. The `setopt RESTORE` action will set the value of all server options back to their factory default.

7.4 Routing

In order for the nodes to communicate over the SCI-network, there must be a routing path between the nodes. The route a packet will travel from a source node to a destination node is decided by a routing algorithm. ScaConf provides several different routing algorithms depending on topology and license type. You may check out the available routing types for your system in the `Interconnect ->Set routing` menu (see figure 7-2) in Scali Universe or the `info routealg` command from `scaconftool`.

Routing is set automatically by `scaconfsd` unless automatic rerouting is explicitly turned off. The machine may also be routed manually using the `Interconnect ->Set routing` menu, or via `scaconftool` with the `reroute` command. As a result of the routing, all unused or broken links will be disabled. Remember that to check the status of the SCI links and nodes you can either use the `Interconnect->Link Status ...` in Scali Universe, or the `list status link all` command from `scaconftool`.

There may arise situations where one or more nodes or one or more links are erroneous. Nodes are considered erroneous by ScaConf if they are in a different state than *OK*. SCI-links are considered erroneous if they are reported as *DOWN*. Routing is computed according to the state of nodes and SCI-ringlets. An SCI-ringlet is considered faulty if one of the links in the ring has failed. This means that controllers reporting their links as *DOWN* will cause the entire ring to be considered broken. You may check the status on links with the `status link` command in `scaconftool`. A ring is also considered erroneous if one or more of the nodes part of this ring is erroneous.

7.4.1 Ring topology

If the machine is configured with all nodes on one SCI ringlet, the routing algorithm is straightforward. All packets to all destinations are sent and received on the ringlet. Ring topology is detected by the line `topology TOP_RING` in configuration file `/opt/scali/etc/ScaConf.conf`. Use the `reroute` command with no arguments to route on a single ring from `scaconftool`.

7.4.2 2D Torus topology

Two dimensional torus topology is detected by the `line topology TOP_TORUS_2D` in configuration file `/opt/scali/etc/ScaConf.conf`. In a 2D torus each node is connected to two SCI ringlets, one in each dimension of the torus. Available routing types are:

- Dimensional routing: `xy`, `yx`, `raw xy` and `raw yx`
- Maxcy (Maximum connectivitY) routing
- C3 (Close Commutative Connection) routing

ClusterEdge

In short the difference is that if one or more rings are faulty, the maxcy and C3 routing algorithms will be able to connect all nodes with at least one working ring, while dimensional routing algorithms will not be able to connect nodes on the faulty rings.

7.4.2.1 Dimensional routing in 2D torus

In dimensional routing all routing in one dimension are completed, before packets are routed in the next dimension. Dimensional routing for 2D torus topologies is selected with the `reroute xy` or `reroute yx` commands. `xy` means that packets are routed in the x dimension first (x-dimension is equal to the horizontal dimension drawn for the `list configuration` command). This is also equal to the dimension connected through link 1 on the SCI adapter. `yx` means that packets are travelling in the vertical dimension first. Vertical refer to the printout from the `list configuration` command in `scaconftool`. The dimensional routing will not be able to connect any of the nodes on erroneous rings, and will only be able to connect nodes where both horizontal and vertical ring are working.

A “raw” version of the dimensional routing is also available. Use the `reroute <xy | yx> raw` command from `scaconftool`. With this algorithm one may force routing to be set as if all nodes and rings are ok. This does not make all rings and nodes ok, but it will set the routing tables on each node as if they where. If raw `xy` routing is set on a system where nodes or rings has failed, one will not get full connectivity. The raw `xy` routing is mostly useful for debugging purposes.

7.4.2.2 Maxcy routing algorithm for 2D torus

The maxcy-algorithm (MAXimum ConnectivITy) is a fault tolerant routing algorithm, capable of connecting the maximum number of nodes if one or more nodes or ringlet/link has failed. When all nodes are working, the maxcy-algorithm is equal to dimensional routing. Maxcy-routing is selected with the `reroute maxcy` command. Rings and nodes which are not *OK*, as far as the configuration server is concerned, will be considered failed when routing is computed, and these nodes and rings will be routed around.

When a node fails, this will make its two SCI-ringlets useless for SCI communication. This means that other nodes on these two failed rings will only have one operational SCI-ringlet left. When this happens, the maxcy-routing algorithm will still be able to connect all remaining nodes, as long as the node is part of at least one working SCI-ringlet. The routing might result in some disabled links. Operational nodes, where one or more rings are broken, will have their link disabled in the direction of the broken ring.

7.4.3 3D torus topology

Three dimensional torus topology is detected by the `topology TOP_TORUS_3D` line in the configuration file `/opt/scali/etc/ScaConf.conf`. In a 3D torus each node is connected to three SCI ringlets, one in each dimension of the torus. The following routing types are available in a 3D torus:

- Dimensional routing
- C3 (Close Commutative Connection) routing

7.4.3.1 Dimensional routing in a 3D torus

Dimensional routing is available for a 3D torus, but it is less fault tolerant than C3 routing. When using dimensional routing, packets are routed along link 0 first, then along link 1 and then along link 2. You may specify another order of dimensions as a parameter to the `reroute` command. The command

```
ConfTool> reroute dimensional order=210
```

will route link 2 first, then link 1 and link 0 last. If a node or a ring fails, the number of nodes in the main routing partition (partition number = 1, see 7.4.4) depends on the order of the dimensions in the routing algorithm. Other nodes, not part of the main partition will not have SCI connectivity to nodes in the main partition. These node may route other SCI-traffic, though.

ClusterEdge

7.4.3.2 C3 routing in a 3D torus

The C3 algorithm is a fault tolerant routing algorithm available for systems with a C3 routing license. The C3 algorithm is capable of maintaining full connectivity between the remaining nodes even if more than one node have failed.

7.4.4 Routing partitions

A routing partition (partition from here) is a collection of nodes which has a route to all other nodes in the same partition via the SCI network. Hence all nodes in the same partition can communicate via the SCI interconnect. Nodes not part of the same partition does not have a route between them, and can not communicate via the SCI-network. Partitions are reported as a positive integer called *partition number* for each node. If a negative number or 0 is reported, this has a special meaning:

- 0** The node is unreachable. (Power off or considered dead in some sense). No MPI programs or anything else may run on nodes with partition number 0.
- 1** Partition number for this node is not known (initial state). Before routing is set, all nodes will have this partition number.

A partition may contain from one to all nodes in the system. The routing algorithms computes which nodes that are part of what partition. If a node is OK but due to routing strategy it is not possible to connect this node to any other node on the SCI-network, it will receive an exclusive partition number. Exclusive partition numbers are those that are held by only one node. Nodes with exclusive partition number can operate separately. You may list the partitions with the `list partition all` command in `scaconftool`.

7.4.5 Testing the routing

Once routing is set, it may be tested with the `sciping` command in `scaconftool`. This command will send an sciping between all nodes listed. The `sciping` command is a combination of `scash` and the command `/opt/scali/bin/sciping`. The response is a list of possibly not responding nodes for each node in the list. Remember that in a system where one or more nodes or rings are down, there may be nodes that will not respond. This applies to faulty nodes or nodes where both SCI rings are unusable. All nodes with the same partition number should be able to reach each other with `sciping`, i.e `sciping #1`.

7.5 The configuration server daemon

The configuration server is available in the ScaConfSd package. The server must run on the frontend and will start automatically when installed and when the frontend is re-booted. The server can be stopped, started or restarted manually with the `/opt/scali/init.d/scaconfsd` script on the frontend:

```
# /opt/scali/init.d/scaconfsd [start|stop|restart]
```

When the server starts, it will read a configuration-file to get the name of all nodes and the configuration of the machine. Default configuration file is:

```
/opt/scali/ScaConf.conf.
```

Another file may be specified with the `-f` option. The `-f` option is only available if the server is started from `/opt/scali/sbin/scaconfsd` binary. The server maintains a database with the current state of the system. The database contains (among others):

- The interconnect configuration of the machine.
- For each node:
- Node name and SCI nodeID.
 - Routing tables.
 - Number of adapters and controllers in the system.

- Link status. This status can be either UP or DOWN indicating if the connection is ok or not.
- Status for each controller. A controller can be either enabled or disabled.
- Status for each node regarding the communication between configuration server and configuration daemon.
- Whether a SCI driver is available or not on each node.
- Routing partition number for each node.

When started, the server will poll every node for status information. If the `AUTO_REROUTE` server-option is enabled, it will reroute the cluster upon start and at every event that demands rerouting. Such events are node or link failure or change of SCI nodeID. For further information about server-options, read section 7.3.11.

7.6 The ScaConf node daemon

For the configuration server to collect information from the nodes, a configuration daemon must run on all nodes in the machine. The configuration daemon will be started upon reboot on each node when installed. It may also be started manually with the standard

```
# /opt/scali/init.d/scaconfnd [start|stop|restart]
```

script on each node or with the `daemon` command in `scaconftool`.

Chapter 7: Interconnect configuration

The Scali binary distribution of Portable Batch System, ScaOPBS allows jobs to be submitted to a batch queue directly from the Scali Universe graphical cluster management interface. ScaOPBS also provides an interface for ScaMPI jobs to be submitted to the batchqueue system.

8.1 Overview

The Portable Batch System, PBS, is a batch job and computer system resource management package. It will accept batch jobs, a shell script and control attributes, preserve and protect the job until it is run, run the job and deliver the output back to the submitter.

Portable Batch System, PBS was developed at NASA Ames Research Center, and later by Veridian System where it was split into a commercial version called PBS Pro and an open source version called OpenPBS. License and support for PBS Pro may be obtained from Veridian at <http://www.pbspro.com>. The full source and other resources for OpenPBS is found at <http://www.openpbs.org>. Additional information and patches for OpenPBS are available from "OpenPBS Public Home" at <http://www-unix.mcs.anl.gov/openpbs>.

ScaOPBS is the Scali binary packaging of Open PBS. It includes :

- `scasub` script for easy submission of ScaMPI/mpich applications
- automatic node/server/scheduler configuration with Scali Universe installation
- multiple patches to improve fault handling, reliability and scalability. Please check the file `/opt/scali/contrib/pbs/doc/RELEASE_NOTES` at the frontend node for a complete list of patches.

The Scali Universe graphical cluster management interface includes functionality to submit and monitor OpenPBS jobs.

PBS consists of four major components: commands, the job Server, the job executor and the job Scheduler.

Commands are used to submit, monitor, modify and delete jobs. Some commands are available for all users, others are available only for the administrator. Commands are conformant with POSIX 1003.2d Batch Environment Standard.

The **job server** (`pbs_server`) provides the basic batch services such as receiving/creating a batch job, modifying the job, protecting the job against system crashes and placing the job into execution. The `pbs_server` daemon is running at the frontend node.

The **job executor** (`pbs_mom`) is placing the job into execution when it receives a copy of the job from the server. It creates a new session as identical to the user login session as possible. It will return the job's output to the user. The `pbs_mom` daemon is running at all the execution nodes in the cluster.

The **job scheduler** (`pbs_sched`) is controlling the policy for which job is run where and when. The scheduler communicates with the various job executors to obtain information about the state of the system resources and the scheduler to learn about the availability of jobs to execute. The `pbs_sched` daemon is running at the frontend node.

8.2 ScaOPBS Installation

8.2.1 Installing OpenPBS during SSP installation.

ScaOPBS can be installed by the Scali Software Platform (SSP) installation. During the installation the user will get the following question:

```
You may install the free queue system OpenPBS (www.openpbs.org).
The ScaOPBS package is a binary distribution of OpenPBS with easy
installation and configuration of OpenPBS and job submission script
for ScaMPI and MPICH applications.

Q: Do you want to install OpenPBS queue system: [n] y

-- Requesting username ----- ? --
Testing of OpenPBS requires a non-root username, the username must be
defined at all nodes.
Use empty username (default) to skip the OpenPBS test.

Q: Please give username: [ ] <testuser>
```

8.3 OpenPBS configuration

OpenPBS does not allow root to submit jobs, hence to be able to test the installation a username is required. The user must be properly defined at all nodes in the cluster for the test to pass. If a username is provided, the installation script will also test OpenPBS at the end of the installation:

```
-- Testing OpenPBS ----- OK --
```

8.2.2 Installing ScaOPBS after SSP installation.

If ScaOPBS was not installed during the SSP installation, you may run an **upgrade** to install ScaOPBS at your cluster. Select

```
# /opt/scali/sbin/SSPinstall
```

After a while, the install script will ask:

```
-- Checking upgradeability ----- ?
```

```
--
```

```
It seems like we are able to upgrade your existing installation.
```

```
Q: Please confirm if you would like to upgrade: [y] y
```

Select yes to upgrade, after a while you will be presented with the questions from “Installing OpenPBS during SSP installation.” on page 114.

8.2.3 Running the ScaOPBS installation tests manually.

You may run the ScaOPBS installation tests manually by using the command:

```
# /opt/scali/libexec/scaopbs.config -t <testuser>
-- Testing OpenPBS installation ----- --
-- Testing OpenPBS with mpimon ----- OK --
-- Testing OpenPBS with mpich ----- OK --
```

Please provide a testuser name that is properly defined at all nodes.

8.3 OpenPBS configuration

8.3.1 Manual reconfiguration of ScaOPBS

The ScaOPBS configuration is based on information from the SSP packages ScaSH (`/opt/scali/etc/ScaSH.scahosts`) and SSP install (`/opt/scali/etc/frontend-name`).

Chapter 8: Batch system ScaOPBS

The default configuration will do for most systems, but if the configuration for these packages is changed or if you would like to manually override this configuration you may use the `scaopbs.config` script. This script has the following syntax:

```
# /opt/scali/libexec/scaopbs.config -h
Usage: scaopbs.config [ -f <frontend>] [ -n "<node1> <node2> ..." ][-tVh?]

-f <frontend>           Use <frontend as frontend>,
                        default : host specified in
                        /opt/scali/etc/frontend-name
-n "<node1> <node2> ..." Use specified nodes as node list,
                        default : hosts specified in
                        /opt/scali/etc/ScaSH.scahosts
-p "<property file>"     Specify node property file,
                        default : file with an entry for every node :
                        "<node> <nodename> <arch> [<mpi impl> ...]"
-s "<server config file>" Specify pbs server configuration file,
                        default : /opt/scali/libexec/pbs_server.conf
```

This script will configure OpenPBS for this node only. The frontend name is an important property at all nodes, hence the script must be run at all nodes with identical settings if the frontend name is changed. The remaining parameters are defined at the frontend only, hence the script is only required run at the frontend. Running the script with default options will reconfigure ScaOPBS as it is configured after installation.

The node property and server configuration files may be modified, the recommended method of modifying these files are to create a backup of the current configuration, modify these backup files and reconfigure the current setting using the `scaopbs.config` script with the modified files as input.

How to create backup files and the node property and server configuration files are described in the following sections.

8.3.2 Backup of local configuration files

Local configuration files (node property file and server configuration) may be backed up by running the command:

```
#!/opt/scali/libexec/scaopbs.config -b "<backup dir>"
```

where `<backup dir>` is the directory where to create the backup files.

8.3.3 The node property file

The default node property file is named `/var/spool/PBS/server_priv/nodes`. It is only available at the frontend and has the following format:

```
<node> <nodename> <arch> [<mpi impl> ...]
```

where:

- `<node>` and `<nodename>` are the name of the node.
- `<arch>` is the architecture for the system: i.e. `Linux2.i86pc`, `Linux2.ia64`, `Linux2.alpha`, or `SunOS5.sparc-u`
- `<mpi impl>` is `ScaMPI` or `MPICH`.

Example node configuration file for a 4 node system with ScaMPI and MPICH installed:

```
# cat nodes
# ScaOPBS node file, node name followed by properties
node-1 node-1 Linux2.i86pc ScaMPI MPICH
node-2 node-2 Linux2.i86pc ScaMPI MPICH
node-3 node-3 Linux2.i86pc ScaMPI MPICH
node-4 node-4 Linux2.i86pc ScaMPI MPICH
```

Please see section 3.2.2 at page 21 in the “Portable Batch System Administrator Guide” (available at the frontend as `/opt/scali/contrib/pbs/doc/pbs_admin_guide.pdf`) for further description of the node configuration file.

8.3.4 Server configuration - qmgr

The server is configured at runtime by the application `/opt/scali/contrib/pbs/bin/qmgr`. Server configuration includes setting the server attributes and establishing queues and their attributes.

8.3.4.1 Listing the current server configuration.

To list the current server configuration, type:

```
# /opt/scali/contrib/pbs/bin/qmgr
Max open servers: 4
Qmgr: print server
```

The example output from `qmgr` below shows a the default server configuration for ScaOPBS with a single queue called `scali_exec` enabled and running:

Chapter 8: Batch system ScaOPBS

```
#
# Create and define queue scali_exec
#
create queue scali_exec
set queue scali_exec queue_type = Execution
set queue scali_exec enabled = True
set queue scali_exec started = True
#
# Set server attributes.
#
set server scheduling = True
set server default_queue = scali_exec
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server scheduler_iteration = 600
```

The example above shows the default server configuration for ScaOPBS with a single queue called `scali_exec` enabled and running.

To modify the server configuration, change the server configuration file created by `scaopbs.config -b <backup dir>` and then reconfigure ScaOPBS using `scaopbs.config -s <server config file>`.

Please see section 3.5 at page 26 in the “Portable Batch System Administrator Guide” (available at the frontend as `/opt/scali/contrib/pbs/doc/pbs_admin_guide.pdf`) for further description of the server configuration.

8.3.4.2 Adding a queue

`qmgr` is used for adding extra queues. In this example a queue called `fluent` is added. The queue is configured to run a single job at a time to handle software license restrictions:

```
# /opt/scali/contrib/pbs/bin/qmgr
create queue fluent
set queue fluent queue_type = Execution
set queue fluent max_running = 1
set queue fluent enabled = True
set queue fluent started = True
```

8.3.5 Removing all ScaOPBS configuration files.

All ScaOPBS configuration files may be removed by running the script `/opt/scali/libexec/scaopbs.rmconfig`

8.3.6 ScaOPBS file locations

The ScaOPBS package includes all files for Open PBS for the frontend running the PBS server and scheduler and for the nodes running `pbs_mom` only.

ScaOPBS files are installed in

```
/opt/scali/contrib/pbs
```

One exception is the script

```
/opt/scali/bin/scasub
```

`scasub` is described in “`scasub`” on page 122.

ScaOPBS configuration and spool files are located in

```
/var/spool/PBS
```

All configuration and spool files are removed when running

```
# /opt/scali/libexec/scaopbs.rmconfig
```

8.3.7 Using aliases for host names

Note for installation on nodes where hostname aliases are used:

If hostname aliases are used in any of the SSP configuration files mentioned in “Manual reconfiguration of ScaOPBS” on page 115, the aliases must be included in the `/etc/hosts` file, i.e. `hostname -a` must report the hostname aliases. Otherwise PBS daemons will not be started at the nodes and OpenPBS will not work properly.

8.3.8 PBS enforcement

ScaOPBS offers an option to enforce use of PBS to access the nodes in the system. In order to ensure that jobs use only nodes that PBS has assigned any kind of remote login/execution on nodes not assigned must be blocked. The solution used is based on work from Willy Weisz at VCPC and makes use of the following facilities:

- prologue/epilogue feature of OpenPBS (version 2.3.15)
- PAM (Pluggable Authentication Modules)

```
<http://www.us.kernel.org/pub/linux/libs/pam/>.
```

Prerequisites:

The following files in the directory `/etc/pam.d` of any available node have to be amended:

- login
- rsh
- sshd (if ssh is used to login to the target node)
- rexec
- any other name of a remote login/execution service on target nodes

The following line must be present in all of these files (Default for Redhat 7.x):

Chapter 8: Batch system ScaOPBS

```
session required /lib/security/pam_limits.so
```

Secondly the root user must have access without password between all nodes in the system.

Running

```
# /opt/scali/libexec/scaopbs.config -e  
will enforce pbs usage. Similarly  
# /opt/scali/libexec/scaopbs.config -d  
will disable this option.
```

The user will be rejected login to nodes in the cluster when PBS enforcement is enabled:

```
% rsh scalil-12  
Too many logins for '<user>'  
Permission denied  
rlogin: connection closed.
```

8.3.9 Node definition using Virtual Processors

Default setup of node definition is that all nodes are "Exclusive Nodes", i.e. only one job will be allowed to run on a node at a time. Multiple MPI processes may be started on a node by using the `-npn` option to `scasub`.

Node specification may be changed to use "Virtual Processors" by:

- unconfiguring of ScaOPBS and making backup of current configuration

```
# /opt/scali/libexec/scaopbs.rmconfig -b <backup dir>
```
- modifying the file `<backup dir>/nodes` setting `np=<no. of CPU's in SMP>` for each node.
- reconfiguring ScaOPBS for use of the modified files:

```
# /opt/scali/libexec/scaopbs.config -p <backup dir>/nodes -s \  
<backup dir>/pbs_server.conf
```

OpenPBS will then allocate one job pr. CPU instead of one job pr. node. However to make a single job run multiple processes pr. node you still have to use the `-npn` `scasub` option. Future versions of ScaOPBS will include an automated option to `scaopbs.config` for configuring of ScaOPBS to use Virtual Processors. Please see section 3.2.2 at page 21 in the "Portable Batch System Administrator Guide" (available at the frontend as `/opt/scali/contrib/pbs/doc/pbs_admin_guide.pdf`) for further description of virtual processors.

8.3.10 xpbs and xpbsmon

The standard graphical tools `xpbs` and `xpbsmon` are not included in ScaOPBS distribution, if you want to install these tools please follow the instructions in the file `/opt/scali/contrib/pbs/doc/README` at the frontend node.

8.4 Example of ScaOPBS usage

The following section gives an example of submitting a small ScaMPI and MPICH program and checking the status of the execution.

Log in as a standard user (root is not allowed to submit jobs to PBS).

To submit a ScaMPI job type:

```
% /opt/scali/bin/scasub -mpimon -np 2 /opt/scali/examples/bin/producer
```

Two nodes will be allocated by OpenPBS for the job. The start-up script created by `scasub` will be initiated by OpenPBS at the first listed node in the list of allocated nodes for the job. The start-up script will determine the list of nodes allocated for the job from a set of environment variables and will start the ScaMPI application at these nodes using `mpimon`

To check status use:

```
% /opt/scali/contrib/pbs/bin/qstat
```

Two output files should be generated in current directory:

```
producer.o1 - stdout from application, should list TEST COMPLETE
producer.e1 - stderr from application, should be empty
```

Similarly, to submit a MPICH job use:

```
%/opt/scali/bin/scasub -mpich -np 2 /opt/scali/contrib/mpich/examples/cpi
```

To show nodes available for pbs, use the command

```
% /opt/scali/contrib/pbs/bin/pbsnodes -a
```

ScaOPBS will configure pbs with a single queue. To show queue status, issue

```
% /opt/scali/contrib/pbs/bin/qstat -Q
```

8.5 Open PBS commands

PBS commands are available in the directory `/opt/scali/contrib/pbs/bin`. man pages are available for most commands. The most frequent used commands are

```
qsub      - submit pbs job
qstat     - show status of pbs batch jobs
pbsnodes  - pbs node manipulation
qdel     - delete pbs batch job
qmgr     - pbs batch system manager
```

Please see man pages for each command for detailed description and options.

8.6 scasub

The `scasub` script is a wrapper for the OpenPBS `qsub` program for easy submission of ScaMPI and MPICH parallel jobs. Unlike other OpenPBS applications and scripts `scasub` is located in `/opt/scali/bin/scasub`. The `scasub` script actually builds a job execution script and submits this script to OpenPBS using `qsub`. `scasub` has the following options:

```
scasub <params> <programe> [program options...]
-np <np>      : Total number of processes, default
               1 for non-mpi programs
               2 for mpi programs
-npn <nnp>    : Number of processes pr. node, default 1
-nodes <nodes>: submit job at given nodes, comma separated
-mpimon      : Submit parallel job using ScaMPI mpimon
-mpich       : Submit parallel job using MPICH mpirun
-v           : Verbose
-qsparams <"parameters"> : Specify native queue system parameters
-mpiparams <"parameters"> : Specify parameters to mpimon/mpirun
-i <file>    : Use <file> as input for stdin
-a          : Start time for job ([[[CC]YY]MM]DD]hhmm)
-l <"resource_list"> : Specify res. for job (man pbs_resources)
-maxtime <mt> : Max time for mpi job in minutes (ScaMPI only)
-m <addr>    : e-mail <addr> when job completed
-N <job name> : Specify name for job
-s <system>  : submit job to given system/resources
-ns         : submit job to any system, independent of arch.

-r <minutes> : reserve nodes for <minutes>
               (nodes listed in file 'reserved_nodes.$PBS_JOBID')
-env <env>   : export list of environment variables, format :
               var1,var2,...
```

8.7 Starting MPI jobs from within scripts submitted to OpenPBS

```
-o <filename> : specify filename for stdout
-e <filename> : specify filename for stderr
-q           : Quiet, no echo of mpimon/mpirun command
-debug      : Debug, print debug output
-Q <queue>  : Destination queue for job
```

8.7 Starting MPI jobs from within scripts submitted to OpenPBS

Sometimes it is not convenient to use the scasub script for starting ScaMPI applications directly, e.g. when the application has its own start-up script. The user may then submit the application start-up script using scasub and specifying the number of processes by the `-np` option (and possibly the number of processes pr. node with the `-npr` option). When the application is started by the queue system, the script is started at the first node allocated by the scheduler. This script may then determine the nodes allocated for the job from the file specified by the environment variable `PBS_NODEFILE`. This file may also be used to determine the number of processes allocated. The environment variable `PBS_NPN` may be used to get the specified number of processes pr. node.

Example start-up script:

```
% cat mpiexample
#!/bin/sh
# Get number of nodes allocated
NO_OF_NODES=`cat $PBS_NODEFILE | egrep -v '^#\|'^$' | wc -l | awk '{print $1}'`
# Multiply with number of processes pr. node to get total number of processes
NO_OF_NODES=`expr $NO_OF_NODES \* $PBS_NPN`
/opt/scali/bin/mpirun -machinefile $PBS_NODEFILE -np $NO_OF_NODES -npr
$PBS_NPN /opt/scali/examples/bin/producer
```

Example submission:

```
% /opt/scali/bin/scasub -np 4 -npr 2 ./mpiexample
```

This example will start a total of 4 processes, two processes pr. node.

Universe XE 8.8 Using ScaOPBS from Scali Universe

Scali Universe provides a graphical user interface towards OpenPBS.

8.8.1 Submitting jobs using Scali Universe

In user mode Scali Universe provides options to submit jobs to OpenPBS. This option is only available when Scali Universe is started in user mode as OpenPBS does not permit root to submit jobs. To submit a job select Run -> Submit Job...

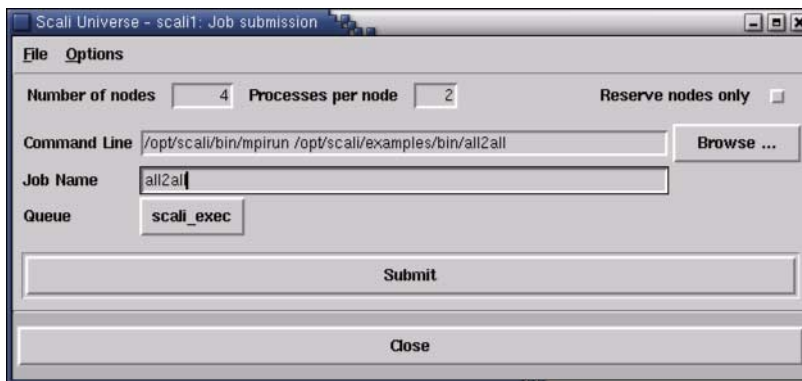


Figure 8-1: Example of submitting a ScaMPI job from Scali Universe to 4 nodes with 2 processes at each node.

The **Number of nodes** field specifies the total number of nodes to allocate for the job.

The **Processes per node** field specifies the number of processes at each node, ie. total number of processes = Number of nodes * Processes per node

The **Command Line** field specifies the actual command to be started by PBS. This command will be started at the first node allocated for the job. The ScaMPI mpirun (and similarly MPICH mpirun) script will pick up the node specification from the PBS_NODESFILe and PBS_NPN environment variables and distribute the processes to the allocated nodes. Regular commands will only be started at the first allocated node.

The **Job Name** field describes the name associated to the job, this name will show up in the queue status window (figure 8-5). If no job name is given, the basename of the Command Line will be used, eg. **mpimom** for the command line specified in figure 8-1.

8.8 Using ScaOPBS from Scali Universe

The Queue field is used for selecting the queue where the job will be submitted.

Reserve nodes only will allocate the specified number of nodes for the user, but will not start any application. When selecting Reserve nodes only the Minutes to reserve field appears where the duration of the reservation must be specified. The name of the reserved nodes may be obtained from the Job details window (see. figure 8-6) or from the file `reserved_nodes.$PBS_JOBID` at the users home directory.

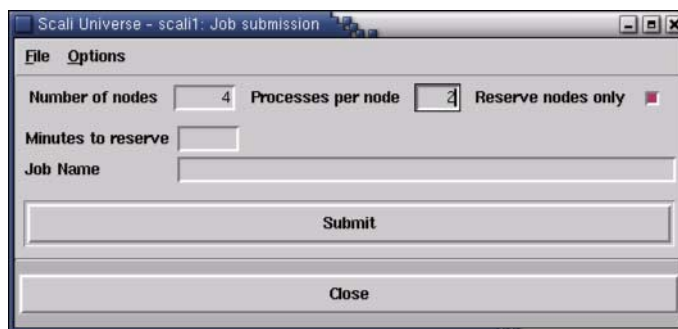


Figure 8-2: Submit window with **Reserve nodes only** selected.

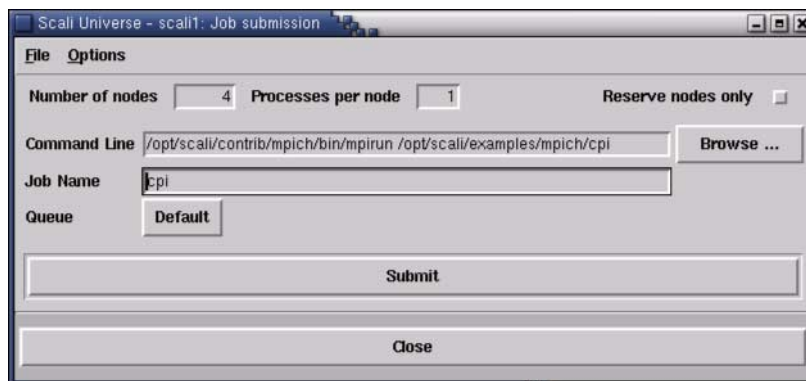


Figure 8-3: Example of submitting a MPICH job from Scali Universe to 4 nodes, one process pr. node.

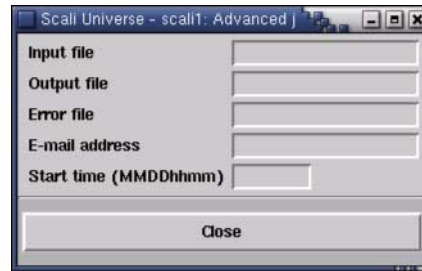


Figure 8-4: Options, Advanced... window for Job Submission

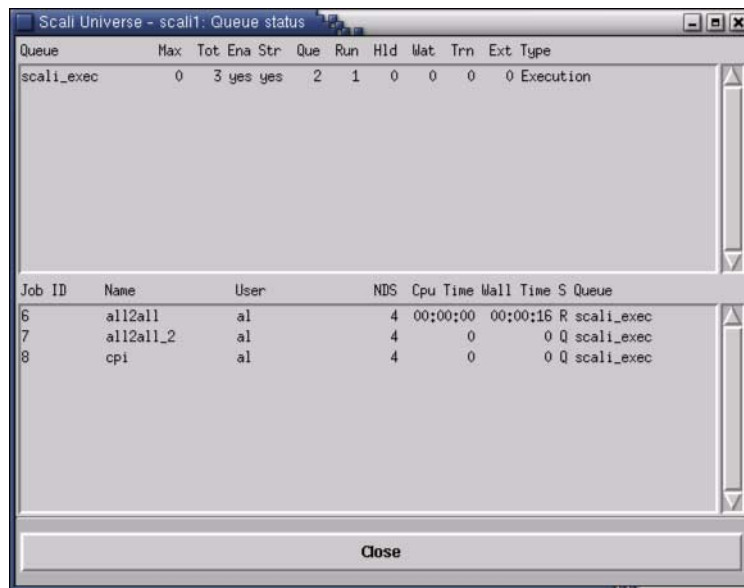
The Options -> Advanced... menu is used for additional parameters for jobs. All these entries are optional.

- **Input file** is the filename used for stdin to the job.
- **Output file** is the filename for stdout from the job.
- **Error file** is the filename for stderr from the job.
- **E-mail address** is the e-mail address used for job completion notification. Note that sendmail must be operational at frontend for this to work.
- **Start time** specifies when the job will be started.

8.8 Using ScaOPBS from Scali Universe

8.8.2 Monitoring jobs from Scali Universe .

The Scali Universe graphical user interface lets you monitor the queue status for OpenPBS. Select Status -> Queue Status to open queue monitoring window.



Queue	Max	Tot	Ena	Str	Que	Run	Hld	Wat	Trn	Ext	Type
scali_exec	0	3	yes	yes	2	1	0	0	0	0	Execution

Job ID	Name	User	NDS	Cpu Time	Wall Time	S	Queue
6	all2all	al	4	00:00:00	00:00:16	R	scali_exec
7	all2all_2	al	4	0	0	Q	scali_exec
8	cpi	al	4	0	0	Q	scali_exec

Close

Figure 8-5: Example Queue Status display in Scali Universe

When you select a running job in the Queue Status window, the nodes used by a running job will be marked in the corresponding “Main Window” of Scali Universe. Use the left mouse button to select job. If you select a job and press the right mouse button you may select actions to perform on the job from a pop-up menu:

- details - detail view of job attributes, see example below
- delete - delete job, only allowed by root or job owner
- hold - put the job on hold, only allowed by root or job owner
- release - release the job, only allowed by root or job owner
- order - order job, only allowed by root or job owner

Chapter 8: Batch system ScaOPBS

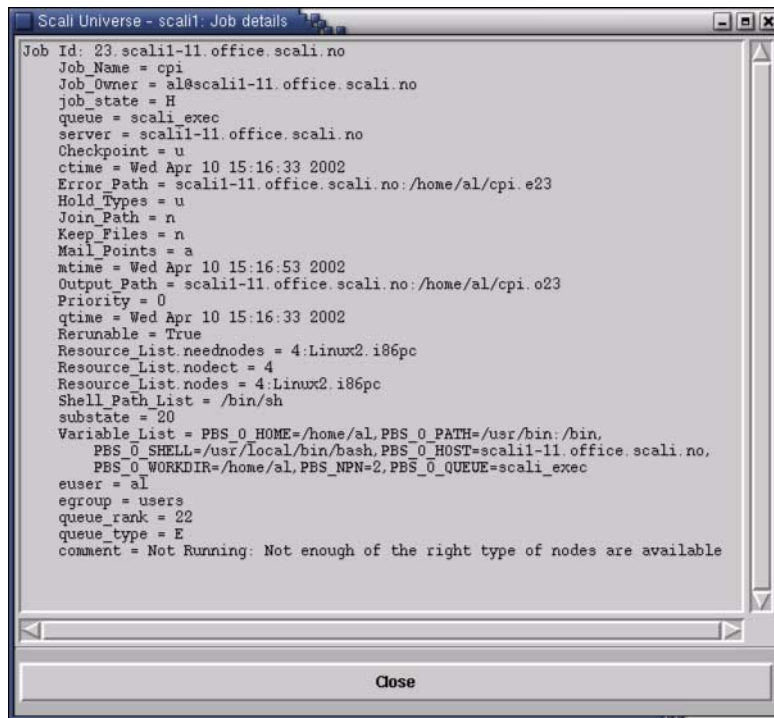


Figure 8-6: Example Queue detail window in ScaLI Universe

8.8.3 Enable and disable OpenPBS queue enforcement.

In administrator mode it is possible to enable and disable OpenPBS queue enforcement. Select Management->Queue Enforcement -> Enable/Disable. For further information about OpenPBS queue enforcement please see “PBS enforcement” on page 119.

8.9 Source code

The full OpenPBS source code can be downloaded from <http://www.openpbs.org>. To obtain the make system and patch files for ScaOPBS, please e-mail support@scali.com. These will also be included in future releases of the ScaOPBS package.

8.10 Software License

OpenPBS (Portable Batch System) v2.3 Software License

Copyright (c) 1999-2000 Veridian Information Solutions, Inc. All rights reserved.

For a license to use or redistribute the OpenPBS software under conditions other than those described below, or to purchase support for this software, please contact Veridian Systems, PBS Products Department ("Licensor") at:

www.OpenPBS.org +1 650 967-4675 sales@OpenPBS.org
877 902-4PBS (US toll-free)

This license covers use of the OpenPBS v2.3 software (the "Software") at your site or location, and, for certain users, redistribution of the Software to other sites and locations. Use and redistribution of OpenPBS v2.3 in source and binary forms, with or without modification, are permitted provided that all of the following conditions are met. After December 31, 2001, only conditions 3-6 must be met:

1. Commercial and/or non-commercial use of the Software is permitted provided a current software registration is on file at www.OpenPBS.org. If use of this software contributes to a publication, product, or service, proper attribution must be given; see www.OpenPBS.org/credit.html
2. Redistribution in any form is only permitted for non-commercial, non-profit purposes. There can be no charge for the Software or any software incorporating the Software. Further, there can be no expectation of revenue generated as a consequence of redistributing the Software.
3. Any Redistribution of source code must retain the above copyright notice and the acknowledgment contained in paragraph 6, this list of conditions and the disclaimer contained in paragraph 7.
4. Any Redistribution in binary form must reproduce the above copyright notice and the acknowledgment contained in paragraph 6, this list of conditions and the disclaimer contained in paragraph 7 in the documentation and/or other materials provided with the distribution.

Chapter 8: Batch system ScaOPBS

5. Redistributions in any form must be accompanied by information on how to obtain complete source code for the OpenPBS software and any modifications and/or additions to the OpenPBS software. The source code must either be included in the distribution or be available for no more than the cost of distribution plus a nominal fee, and all modifications and additions to the Software must be freely redistributable by any party (including Licensor) without restriction.

6. All advertising materials mentioning features or use of the Software must display the following acknowledgment:

"This product includes software developed by NASA Ames Research Center, Lawrence Livermore National Laboratory, and Veridian Information Solutions, Inc. Visit www.OpenPBS.org for OpenPBS software support, products, and information."

7. DISCLAIMER OF WARRANTY

THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT ARE EXPRESSLY DISCLAIMED.

IN NO EVENT SHALL VERIDIAN CORPORATION, ITS AFFILIATED COMPANIES, OR THE U.S. GOVERNMENT OR ANY OF ITS AGENCIES BE LIABLE FOR ANY DIRECT OR INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND IN ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This license will be governed by the laws of the Commonwealth of Virginia, without reference to its choice of law rules.

Chapter 9 ScaSH - parallel shell tools

The ScaSH parallel shell tool suite is a collection of scripts enabling the execution of commands in parallel on a large number of target nodes:

- **scash** : Execute command on nodes in a Scali cluster.
- **scahosts** : Checks nodes for availability.
- **scacp** : Copies file(s) locally on nodes in a Scali cluster.
- **scarcp** : Copies file(s) from/to local machine to/from nodes in a Scali cluster.
- **scaps** : Prints processes on nodes in a Scali cluster.
- **scakill** : Kill processes on nodes in a Scali cluster.
- **scarup** : Prints status on nodes in a Scali cluster.
- **scawho** : Prints user names and number of processes on nodes in a Scali cluster.

The file `/opt/scali/etc/scaSH.conf` contains configuration parameters to the ScaSH suite of tools. It will be sourced by the tools and must adhere to bourne shell script syntax. Table 9-1 on page 137 describes the options in the configuration file and lists their default values. All programs in the ScaSH suite will use the nodes reported by the **scahosts** program unless nodenames are specified on the command line.

9.1 scash - the parallel shell command

The **scash** utility located in `/opt/scali/bin` is a UNIX script that can run the same shell command on a set of Scali system nodes. The target nodes may be specified in a configuration file or on the command line (see description of **scahosts** program in 9.2). The command options are listed below.

Usage:

```
scash [options] ([-n "nodenames"] <command>)|(-c "command" [<nodenames>])
```

[options]:

- | | |
|----|--|
| -? | Print help text |
| -h | Print help text |
| -v | Print machine name and command before each output block |
| -V | Print version |
| -p | Print machine name before each line in each output block |
| -d | Dryrun: only print command, do not execute |
| -b | Run in background |
| -w | Wait until command has finished on all hosts |

Chapter 9: ScaSH - parallel shell tools

- a** Use 'at now' to execute command
 - s** Which remote execution/copy command to use
 - r key** Replace key with machinename if key is found in command
 - l <n>** Limit on number of parallel processes to run in background mode
 - F <n>** Fanout value. Default is 8. Use 0 or 1 to force no fanout.
 - n "n1 n2 .."** Nodenames separated by space characters
 - f <file>** Use node names from file containing nodenames separated by newlines
 - x "n1 p2 ..."** List of nodenames to be excluded
 - z** Do NOT check access to nodes (the default is to ping and test remote shell access)
 - c <cmd>** The shell command to be run
- <Node names>:
The default if no nodes are specified is all nodes in the user's local **scahosts** file (**/home/user/scahosts**).
If that file does not exist, the global **scahosts** file (**/opt/scali/etc/ScaSH.scahosts**) is used. Whenever you explicitly specify nodes the contents in the config files is ignored.
- <command>:
Any command you want!

9.2 scahosts - nodes availability check

The **scahosts** program located in `/opt/scali/bin`, will check a number of hosts for availability. An available host is one that answers to a ping request and is open for remote shell access. The program will print the name of the available hosts. Which hosts the program will check may be specified in several ways. One may specify hosts on the command line when running the program, either with the **-f** option which gives the path to a file containing host names or with a list of hosts at the end of the command. If neither the **-f** option nor the hostlist is present, **scahosts** will look for hosts in the file `$HOME/.scahosts`. If this file is not available, it will look for hosts in the file `/opt/scali/etc/ScaSH.scahosts`. The file containing hostnames should have one name per line. The option `RSH_TESTACCESS` in the configuration file `scaSH.conf` (see table 9-1) may be set to disable checking for remote shell access to validate a node.

Usage:

```
scahosts [options] [host ...]
```

[options]:

- f <file>** Use node names from file containing nodenames separated by newlines
- v** Print verbose information
- z** Do NOT check access to nodes
- x "n1 p2 ..."** List of nodenames to be excluded
- V** Print version
- ?/-h** Print help message.

9.3 scscp - local file copy

The **scscp** program copies file(s) locally on nodes in a Scali cluster

Usage:

```
scscp [options] <from> [<from>] <to>
```

[options]:

- ?** Print help text
- h** Print help text
- R** Copy recursively
- v** Print machine name and command before each output block
- V** Print version
- p** Print machine name before each line in each output block
- d** Dryrun: only print command, do not execute
- b** Run in background
- w** Wait until command has finished on all hosts
- n "n1 n2 .."** Nodenames separated by space characters
- f <file>** Use node names from file containing nodenames separated by newlines
- x "n1 p2 ..."** List of nodenames to be excluded
- z** Do NOT check access to nodes (the default is to ping and test remote shell access)

9.4 scarcp - remote file copy

The program **scarcp** copies file(s) between local machine and nodes in a Scali cluster. One may specify the command to be used when copying files with the **-c** option.

Usage:

```
scarcp [options] <from> [<to>]
```

[options]:

- ? Print this text
- h Print this text
- g Copy from remote nodes to local node (default is local to remote)
(use in combination with -r option to avoid overwriting file)
Note that use of the -g option will force fanout to be 0 (zero)
- R Copy recursively
- v Print machine name and command before each output block
- V Print version
- p Print machine name before each line in each output block
- d Dryrun: only print command, do not execute
- b Run in background
- w Wait until command has finished on all hosts
- r **key** Replace key with machinename if key is found in command
- s <cmd> Which remote execution/copy command to use
- F <n> Fanout value. Default is 8. Use 0 or 1 to force no fanout.
- n "n1 n2 ..." Nodenames separated by space characters
- f <file> Use node names from file containing nodenames separated by
newlines
- x "n1 p2 ..." List of nodenames to be excluded
- z Do NOT check access to nodes (the default is to ping and test remote
shell access)

9.5 scaps - process information

The command **scaps** prints processes on nodes in a Scali cluster

Usage:

```
scaps [options]
```

[options]:

- ? Print help text
- h Print help text
- u **user** Only list those processes that match the given username
- s **string** Only list those processes that contains the given string
- v Print machine name and command before each output block
- V Print version
- p Print machine name before each line in each output block
- d Dryrun: only print command, do not execute
- b Run in background
- w Wait until command has finished on all hosts
- n "n1 n2 ..." Nodenames separated by space characters

- f <file> Use node names from file containing nodenames separated by newlines
- x "n1 p2 ..." List of nodenames to be excluded
- z Do NOT check access to nodes (the default is to ping and test remote shell access)

9.6 scakill - parallel kill command

The **scakill** program allow you to kill or send a specific signal to processes running on nodes in a Scali cluster.

Usage:

```
scakill [options]
```

[options]:

- ? Print help text
- h Print help text
- v Print machine name and command before each output block
- V Print version
- p Print machine name before each line in each output block
- d Dryrun: only print command, do not execute
- b Run in background
- w Wait until command has finished on all hosts
- n "n1 n2 ..." Nodenames separated by space characters
- f <file> Use node names from file containing nodenames separated by newlines
- x "n1 p2 ..." List of nodenames to be excluded
- z Do NOT check access to nodes (the default is to ping and test rsh access)

One or more of the options following *must* be specified:

- i **startpid-stopid** Kill those processes that lie within the given process id range
- u <user> Kill those processes that match the given username
- s <txt> Kill those processes that match the given string
- l <level/name> Kill level or name (e.g. 9, HUP, ...)
- <level> Kill level (only numeric value allowed)

:

9.7 scarup - node status

The **scarup** command prints up-time/load information about nodes in a cluster.

Usage:

```
scarup [options]
```

[options]:

- ? Print help text
- h Print help text
- v Print machine name and command before each output block
- V Print version
- p Print machine name before each line in each output block
- d Dryrun: only print command, do not execute
- b Run in background
- w Wait until command has finished on all hosts
- n "n1 n2 ..." Nodenames separated by space characters
- f <file> Use node names from file containing nodenames separated by newlines
- x "n1 p2 ..." List of nodenames to be excluded
- z Do NOT check access to nodes (the default is to ping and test rsh access)

9.8 scawho - user information

The **scawho** program prints usernames and number of processes on nodes in a Scali cluster.

Usage:

```
scawho [options]
```

[options]:

- ? Print help text
- h Print help text
- v Print machine name and command before each output block
- V Print version
- p Print machine name before each line in each output block
- d Dryrun: only print command, do not execute
- b Run in background
- w Wait until command has finished on all hosts
- n "n1 n2 ..." Nodenames separated by space characters
- f <file> Use node names from file containing nodenames separated by newlines
- x "n1 p2 ..." List of nodenames to be excluded
- z Do NOT check access to nodes (the default is to ping and test remote shell access)

9.9 ScaSH configuration file

The file `/opt/scali/etc/ScaSH.conf` contains configuration parameters for the ScaSH parallel shell tools suite. It will be sourced by the tools and must adhere to bourne shell script syntax. Table 9-1 describes the options in the configuration file and lists their default values.

Option	Default value	Description
RSH_CMD	rsh -n	Which program should be used to execute a command remotely. The program specified must support the following calling convention: <code>\$RSH_CMD <host> <command></code>

Table 9-1: Options in the file ScaSH.conf

Chapter 9: ScaSH - parallel shell tools

Option	Default value	Description
RCP_CMD	rcp -p	Which program should be used to copy files from/to a remote node. The program specified must support the following calling convention: \$RCP_CMD <host>:<fromfile> <tofile> or \$RCP_CMD <fromfile> <host>:<tofile>
LCP_CMD	cp -p	Which program should be used to copy files on the local node. The program specified must support the following calling convention: \$LCP_CMD <fromfile> <tofile>
PARLIMIT	expr `ulimit -n` / 3`	When using the background option with one of the ScaSH tools, this parameter will control the number of commands that are actually run in parallel. ScaSH tools will wait until the below number of processes are finished before issuing the next batch.
RATESLEEP	2	Heavy use of rsh/rcp can lead the inet daemon to enter a non responsive state. This is related to the fact that rsh uses socket ports below the 1024 limit and exhausts the number of free ports OR that the number of successive connections to the shell service are too many to be handled. Both problems arise because the sockets enter a TIME_WAIT state which uses ~120 seconds to terminate. To limit the number of successive rsh accesses to each host use RATESLEEP to adjust interval between each access to that node. (If you are using ssh or other remote execution programs which do not have the above mentioned problem set the value of RATESLEEP to 0).
RSH_TESTACCESS	true	Define if the <i>scahosts</i> program should test the nodes for remote shell access to verify access to nodes. The default is true.
FANOUT	8	Define default FANOUT factor to be used when not explicitly stating the FANOUT factor on the commandline. FANOUT is used for limiting the number of connections from one host when running a <i>scash</i> command. When the number of nodes used as argument for the <i>scash</i> command exceeds the FANOUT factor, the hosts will be divided into groups where FANOUT gives the number of groups and the <i>scash</i> command will be run as a <i>scash</i> command in each group in parallel. Hence, the number of connections from one host will be limited to FANOUT number of connections.

Table 9-1: Options in the file ScaSH.conf

This chapter is the place to start if you are experiencing problems either using or installing a Scali system. Some of the most common problems and their solutions are described here. Remember that the FAQ (Frequently Asked Questions) lists at Scali's web site may contain information which is more up to date than the one contained in this document.

If your problems persists please refer to "Technical Support" on page 149 which explains how to get technical support.

10.1 Hardware installation problems

Before you investigate your problems further, make sure the hardware installation is in accordance with the guidelines in Chapter 3.

10.2 Software installation problems

Before you investigate your problems, check that your software installation is according to the guidelines in Chapter 4. If you experience problems during installation, always attach a copy of the output from the install script if you contact support@scali.com¹.

1. Wulffit customers should contact wulffit-support@dolphinics.com

Chapter 10: Troubleshooting

Problem	Solution
SCI test fails during SSP install.	<ol style="list-style-type: none"> 1. Check that the cabling is according to the file <code>/opt/scali/etc/ScaConf.conf</code>. If you find that there is a mismatch, correct the cabling, or the file, (whichever is in error), and restart <code>scaconfsd (/opt/scali/init.d/scaconfsd restart)</code>. Then you can rerun the SCI test from the install script. 2. Check that the kernel is supported. If kernel is not supported, run <code>ScaSCIadap</code> 3. Check that all links and nodes are OK: Start <code>/opt/scali/sbin/scaconftool</code>. Issue command <code>'list no st links all'</code> and check that all nodes has status OK and that all links are UP and ENABLED. If there are any nodes that are UNREACHABLE or links that are DOWN, fix this problem before retrying the SCI test from the install script. 4. Look for lines containing "ssci:" in <code>/var/log/messages</code> on all nodes. This is log/error messages created by the ScaSCI driver and they might give a clue on why the SCI interconnect does not have full connectivity.
When installing SSP or running <code>scapkg</code> , I get the error message: 'all ports in use'.	The solution is to reduce the number of nodes to install on, and run several rounds of <code>scapackge</code> until all nodes are covered. This may be a problem on large systems. New versions of <code>scash</code> is supposed to handle this, but if you run into this situation during <code>SSPinstall</code> , you can stop the <code>SSPinstall</code> script and try to install packages with <code>scapkg</code> : Run <code>/opt/scali/sbin/scapkg</code> on half of the nodes first, then on the rest. If you still get 'all ports in use' try running on a smaller set of nodes. When <code>scapkg</code> has been run with all nodes you may resume with the installation process.
Access test to nodes failed for one or more nodes.	Read the file <code>/opt/scali/doc/OS</code> and make sure <code>rsh/ssh</code> configuration on the nodes in question are according to the prerequisites stated there.
Installation of ScaSCI driver rpm complains about 'unresolved symbols'.	SSP only support certified RedHat base or update linux kernels. If you, on your own risk, try to install SSP with a different kernel, you will need to run <code>ScaSCIadap</code> to adapt the <code>ScaSCI</code> driver to the kernel.

Table 10-1: Software installation troubleshooting

10.3 License problems

Troubleshooting licensing problems is described in the appendix "Scali Software Licensing" on page 157.

10.4 MPI problems

Common problems encountered running MPI programs with Scali's MPI implementation ScaMPI, is handled in chapter 4.4 "When things don't work - troubleshooting" in the "Scali Library User's Guide" accompanying the product. You should also take a look at the on-line ScaMPI FAQ in the support section at Scali's web-site: <http://www.scali.com/support>.

10.5 SCI Interconnect problems

The LEDs on the SCI-card can give hints about the condition of the SCI interconnect, refer to Table 10-2: This table is only valid for D33x cards.

Colour	Behaviour	Interpretation
Yellow	Steady	Power on D33x card, but no cable installed correctly, remote computer not powered or erroneous cable.
Green	Steady	Cable installed OK.
Yellow	Short period	Self test after power on, after installation of cable or after software reset.
Green	Blinking	Cable installed and OK and SCI traffic on B-Link.

Table 10-2: LED behaviour on D33x cards

Scali uses its own highly optimized driver **ScaSCI** when utilising the ultra high speed SCI interconnect in a Scali system. The LEDs on the SCI card will not be steady green until this software is loaded and all cables are inserted correctly.

10.5.1 SCI error messages

The SCI status error messages reported by `/opt/scali/bin/mpimon` are explained by the program `/opt/scali/bin/scierrmsg`. Taking as input a SCI status error message number or a symbolic name, **scierrmsg** (see C-1.3) gives a description of the cause of the error. Some common SCI error messages are explained here:

Chapter 10: Troubleshooting

SCI error type	Description
ICMS_NO_RESPONSE	Reporting of this message type when attempting communication with an existing node is an indication that something is wrong because the remote node did not respond. Either the physical connection is broken, there is no route back from the destination or the node is down or unable to respond.
ICMS_CONNECTION_REFUSED	This message type is returned when the connection request was refused by the memory owner either because the resource (remote memory) does not exist or it is not currently offered to new connections.
ICMS_OUTBOUND_MAP_FULL	This error type indicates that not enough SCI map resources are available to satisfy the application need. (That is the local SCI adapter address translation table (ATT) is full and there is no room for more remote memory mappings). This is unrelated to the state of the interconnect (except of course that processes may hang and block resources because of some interconnect problem). See “wrong jumper settings” in Table 10-4.
ICMS_FAILURE	This error type indicates that an operation failed due to an implementation detail which does not map to the abstract level. This may indicate a server memory alloc situation or to many open SCI driver connections.
ICMS_OUT_OF_MEMORY	This error type indicates that it was not possible to allocate the requested amount of pinned memory. Add more memory to the system or change configuration options to allow a more aggressive SCI memory allocation strategy.

Table 10-3: Some SCI error types

10.5.2 SCI Troubleshooting

Problem	Description	Solution
Interrupt conflict	During ScaSCI install the SCI hardware has an interrupt conflict with other hardware cards.	Please consult the operating system install documentation about how to solve the problem.
Wrong jumper settings. NB! This only applies to D31X type of cards.	After installation of the ScaSCI software package, run: <code>/opt/scali/sbin/scinfo -m</code> to check that jumper settings are correct. Used/available ATT entries should typically be: RMAP_IO of 1/256 and RMAP_GATHERING of 1/256 when ATT size is 512k (default), i.e., the SCI driver is able to map a maximum of 256*512k=128M remote memory.	Check the ScaSCI release notes if your values are different and correct the SCI board jumper settings or the driver configuration values.
No adapters found	Use the <code>/opt/scali/sbin/scicards</code> script to find the number of installed SCI adapters on the node. If the answer does not match your hardware installation of SCI adapters or you get the error message: <i>SciInitialize failed because no adapters found</i> , check if the SCI hardware is recognised by the OS.	Are your motherboards supported? Check the ScaSCI release notes. Check if the SCI hardware are recognised by the OS: Solaris: <pre>% prtconf -pv grep 'PCI: 11c8' model:'PCI:11c8,658-class:Bridge dev.' model:'PCI: 11c8,0 - class: Bridge device' model:'PCI: 11c8,658 - class:Bridge dev.'</pre> Linux: <pre>cat /proc/pci grep 'Vendor id=11c8' Vendor id=11c8. Device id=658.</pre>
LEDs on SCI cards are red/yellow ^a . (1)	One or more of the LEDs on the SCI card are red/yellow ^a and the installation process is completed.	Check that cabling is according to the guidelines in section 3.4.3. Check that all cables run from IN connector to OUT connector, and that all links are connected to the same link number on a neighboring node. (Link 0 on node A is connected to link 0 on node B etc) . Correct the cabling if erroneous. Otherwise take a look at the next issue.
Red/yellow ^a LEDs on SCI cards. (2)	One or more of the LEDs on the SCI card is red/yellow ^a , and the cabling is checked and found ok.	Check that all the cables are properly inserted into the connector, and tighten all screws.

Table 10-4: ScaSCI troubleshooting

Chapter 10: Troubleshooting

Problem	Description	Solution
Red/yellow ^a LEDs on SCI card (3)	Checking cabling and tightening screws didn't help.	Check that the SCI driver is loaded on all nodes. You may check this with the <i>list status all</i> command in scaconftool , see section D-2.9
OS does not recognize the SCI card	Some PCI slots do not fully support the specification. Interference with AGP or other hardware devices may happen.	Try another PCI slot for the SCI card. If changing PCI slot for the SCI cards does not help, please refer to Dolphin. http://www.dolphinics.com .
sciping reports "not responding" nodes	The SCI utility <code>/opt/scali/bin/sciping</code> or the sciping command in the ScaConf utility scaconftool reports not responding nodes.	Reroute the cluster with scaconftool or ScaDesktop. Make sure sciping is run only between nodes with same routing partition. See section 7.4.4.
ScaSCI log message RMAP_OVERFLOW	The ScaSCI log contains RMAP_OVERFLOW message of lost pages: Unix: WARNING: kernel map: rmap overflow, lost [39250, 39254] Unix: WARNING: kernel map: rmap overflow, lost [39238, 39242] ... error messages repeated multiple times in the log.	The problem is the MEM_BLOCK_SIZE setting in the ScaSCI.response file that is reflected to the mem_block_size variable setting in the ssci.conf file on all nodes at ScaSCI package installation time. This variable regulates the size of each page aligned block of memory allocated from the OS (e.g. "granularity" of allocation).

Table 10-4: ScaSCI troubleshooting

- a. Red applies to old type of SCI cards, i.e. D31x . Yellow applies to new type of SCI cards, D33x

10.5.3 SCI link errors.

For systems with an explicit support license, a special test programs for the SCI network is available in the ScaDiagSC package. ScaDiagSC is intended for use at troubleshooting problems related to SCI hardware, and consists of three programs; **sci_hwid**, **sci_hwtop** and **sci_hwdiag**. They are all ment to be used when the system is operational, and may be used to find failing cards and cables that are not detected by hardware or topology miss-configurations.

10.5.3.1 sci_hwid

The **sci_hwid** program checks that every link controller on the nodes have a unique SCI IDs. If every ID is in fact unique, 0 will be returned to the calling environment. If not 1 will be returned and an error message will be printed.

The `-l <lc list>` input parameter determines which link controllers to check Default is to check link controller 0. In a 2D torus there are two link controllers. To check them both use `/opt/scali/sbin/sci_hwid -l "0 1"`. If the program discovers any non unique SCI IDs, please contact your SCI hardware vendor to sort this out.

10.5.3.2 sci_hwtop

The **sci_hwtop** program performs a series of link controller resets to figure out how a set of nodes are connected (topology). **sci_hwdiag** will get the node names from **/opt/scali/bin/scahosts -z**.

You may run **sci_hwtop** in one of several modes, **--print**, **--write**, **--connections**, **--topology**, **--rings**, and **--positions**. Default is **--print** mode. The **--print** and **--write** modes will find the connection list and topology used in the **/opt/scali/etc/ScaConf.conf** file and print them to screen or write it directly to the file. Only unbroken rings will be printed in the connection list. The topology will be one TOP_RING(_CR), TOP_TORUS_[23]D or TOP_FREE_?D. The ring and torus topology are the ones supported. Still, no error messages will be printed if the cluster is not connected in a non-supported topology (ie. TOP_FREE_?D). The topology will be TOP_FREE_?D if some nodes or link controllers are down.

The **--connection** mode checks if the nodes are connected according to the connection list in the **/opt/scali/etc/ScaConf.conf** file. If the connection list is correct, 0 will be returned. If not, 1 will be returned and error messages indicating what might be wrong will be printed. The same goes for the **--topology** mode. 0 will be returned if the cluster is connected in a supported topology, that is a ring or torus topology. If not, 1 will be returned and error messages will be printed.

The **--rings** mode prints the rings in a more reader-friendly manner, with one ring per line, and the **--positions** will print the nodes positions in a ring, 2D or 3D topology with respect to an origin node.

There are several options to change the behaviour of **sci_hwtop**. With **-f <ScaConf.conf file>**, **sci_hwtop** will get the rings from this file rather than perform hardware probe with link controller resets. It makes most sense to use this in combination with the **--rings** or **--positions** modes. You may decide which node to be the origin in the cluster with the **-1 <node>** option. This node will be listed first in the connection list, and will have position 0, 0 (2D).

If you have a cluster where the nodes have different numbers of link controllers and/or the rings are not connected through the same link controller, you need to use the **-r** flag. This will create a connection list of all rings (unbroken) in the cluster. Isolated parts which is not somehow connected to the starting node will not be included. The topology will be TOP_FREE. You may use this flag with the **--print**, **--write** and **--rings** modes. It will not make sense to run any diagnostic mode (**--connections** and **--topology**) with the **-r** flag.

10.5.3.3 sci_hwdiag

The **sci_hwdiag** program runs an MPI program systematically on parts of a Scali cluster to test the interconnect. On older SCI cards (pre PSB66, i.e D31x SCI cards), the b-link retry must be set to zero. The b-link retry mechanism is controlled with the `blink_retry_count` parameter in the file `/opt/scali/kernel/ssci.conf`. To disable this SCI hardware feature, please use `/opt/scali/libexec/b_link.sh off` to turn it off. To turn it back on use `/opt/scali/libexec/b_link.sh on`. This will turn b-link back to the value before last time `b_link.sh off` was run (if you run `b_link.sh off` twice, `b_link.sh on` will have no effect since it sets `blink_retry_count` to zero. Default value for b-link retry is 64). Make sure routing is OK after turning b-link on/off.

The way of the program:

Before the test application is started, the cluster is set in “diagnostics mode”. This involves turning of all automatic modes in ScaConf server daemon (**scaconfsd**) and it may also imply reconfiguration of the ScaSCI driver. Default ScaSCI driver configuration for diagnostic mode is **max_trans_size=64**, **lc_ec_period=0** and **scilink_freq=200**. These parameters will be reconfigured with **scireconf**. For the reconfiguration to take effect, the ScaSCI driver is reloaded on all nodes before the MPI test application starts. When the tests are finished, these parameters will be set back to their original values. If the value originally was not the same on all nodes, the parameter will be commented out from the file `/opt/scali/kernel/ssci.conf`, and default value will be used when the ScaSCI driver is reloaded.

With the **-S** flag, you may add your own parameters to be reconfigured, or override the default. Do not use this flag if you don't know what you are doing. The flag is followed by a string with format “<parameter name> <value>”. The parameter you specify with the **-S** flag will be set when the diagnostic mode is applied.

When the MPI test application is run the cluster will be divided into rings and rectangles (this goes for 2D torus topology). First run is for all horizontal rings, second run is for all vertical rings, third run the machine is divided into 2x2 squares of neighbour nodes. In 3D torus topology the cluster will be divided into `lc0_rings`, `lc1_rings` and `lc2_rings`. No rectangles or cubes are tested.

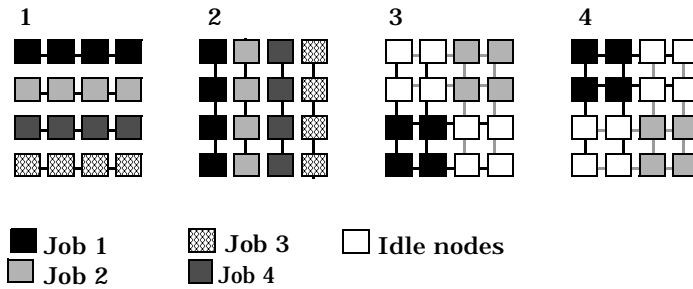


Figure 10-1: Vertical, horizontal and rectangle groups of nodes running the same test application job in a 4x4 2D torus. The wraparound connections of the tours are not shown.

The test application will be run on all `lc0_rings` (vertical) in parallel as shown in part 1 of Figure 10-1.; then on all `lc1_rings` (horizontal) in parallel as shown in part 2 of Figure 10-1: When the ring by ring tests are completed, the cluster will be divided into 2x2 rectangles of nodes as shown in part 3 and 4 of the figure. The rectangles will be split in several groups, such that no two nodes not in the same rectangle are on the same `lc0` or `lc1` ring. There will be a staircase-like pattern of the rectangles which are run in parallel. Larger clusters will have more groups, a 6x6 cluster will have 3 groups with 3 rectangles in each.

Nodes which are down will be ignored, and rings in which these are members will not be tested. No rectangles will be tested if some nodes are down. You may state which nodes that are not to be tested with the `-x <nodes>` option. To ensure that communication between nodes on a particular ring stays on that ring, on 2D systems dimensional routing will be used (One run with `xy` and one testrun with `yx` routing). This in turn means that all SCI controllers and links should be up and enabled. You need to run `sci_hwdiag` as root.

`sci_hwdiag` will detect how many processors the nodes have, and start the MPI test application with the correct parameters. You may manually overrun this with the `-1` or `-2` flag, to run with either 1 or 2 process(es) per node.

With the `-c` flag you may specify your favourite MPI application to use when running the tests. The flag should be followed by a string enclosed in “ ”, which should contain what you usually put between the command `/opt/scali/bin/mpimon` and `'--'` before specification of the nodes. No specification of the test application is equal to running with:

```
-c '/opt/scali/contrib/PMB2/build/Linux2.i86pc.gnu/PMB-MPI1 -multi 0'
```

Chapter 10: Troubleshooting

With the “-c **bidirect**” parameter, permuted bidirect (from **/opt/scali/examples/bin**) will be used as test application. This was default behaviour for **sci_hwdiag** prior to version 1.2.0. Rings longer than 4 nodes will be divided into shorter parts to limit the number of permutation in the **run_permuted_bidirect** program. The following flags are parameters to vary the behaviour of the **run_permuted_bidirect** program:

```
-u    Unidirectional mode only
-b    Bidirectional mode only
-q    Permuted bidirect quick mode,
      work reduced with a factor of 10
-Q    Permuted bidirect very quick mode,
      work reduced with a factor of 100
-d    Dryrun, nothing really done
```

These flags have no effect without the -c 'bidirect' flag. With the “-c **PMB**” flag, the Pallas benchmark is used as test application. This is default if no other test application is specified with the -c option.

By default all output from the test application will be written to **/dev/null**. If you wish to save the output from the test runs you may use the -o flag. With no parameter the output from the different runs will be written to files in the current directory. The file name will contain the name of the nodes it was run on. With a parameter to the -o option you may give the path to a directory with write access where you want the files to be put.

Between each test run, **sci_hwdiag** will print the value of non zero counters in the SCI-driver. Link errors and Times handler (...) will be listed in a summary at the end. Other non-zero counters will only be listed during the run. Please be aware that the ScaSCI software detects and handles all these incidents, hence data transfer is always safe. For more information on how to interpret the results, please take a look at **/opt/scali/doc/ScaDiagSC/GUIDE**.

Various types of errors may show up during the run. You should pay special attention to large amount of link errors;

```
"node1-1    : LC[0] link errors:                27"
```

A bad cable in the cluster may show up as link errors when running either the `lc0_rings` or the `lc1_rings`. The bad cable may be in connection with the node on which the error occurred, but not necessarily. Due to the nature of the SCI ring, errors are not always discovered where they are generated, but may propagate to several nodes in the ring.

If you get errors when running rectangles but none when running `lc0_rings` nor `lc1_rings`, this may indicate a bad SCI board. Errors evenly distributed all over the cluster may be a sign of some other problem not related to bad SCI boards or cables. Not all counters reported are due to errors. The message

```
"node1-1 : Times handler called and interrupt not claimed: 90302"
```

may indicate that the SCI card share interrupt with another device, typically your NIC. This is not an error, but may in certain circumstances affect system performance.

10.6 Technical Support

Scali is proud to provide the highest level of technical support. Normal first-response time for a support request is less than 8 working hours. You will then be appointed a personal contact and a serial number for your particular problem.

10.6.1 Support Contracts

The duration and level of your support contract or warranty should be obvious from the invoice that came with the product. If in doubt just contact Scali¹. Extended support contracts or pay-per-request are also available. When requesting support please state your *name* and *company*.

10.6.2 How to report the problem

It will save both your time and ours if you do the following before requesting support

- Read the documentation again: Quite often the problems can be resolved from the relevant chapters in the manuals or FAQs.
- Collect version information: This includes version information for *all* involved software; Scali SW, compilers operating system. Versions of HW: SCI-cards, computer architecture and chipset versions (BX, LX, NX ...).
- Isolate your problem to a small test case - if applicable. The shorter the better.
- Record the sequence of events causing the problem.

1. Wulffit customers should contact their reseller or Dolphin Interconnect Solutions.

Chapter 10: Troubleshooting

You can reach Scali's technical support using either:

E-mail:	support@scali.com (Preferred)
Phone:	(+47) 2262 8950
Fax:	(+47) 2262 8951

Comments regarding beta software may be sent to:

E-mail:	beta-support@scali.com
----------------	-------------------------------

Support for Wulfkits are handled by Dolphin Interconnect Solutions. Please contact **wulfkit-support@dolphinics.com**.

10.6.3 Other Feedback

Scali welcomes any suggestions, improvements, feedback or bug reports concerning this Scali System Guide or the hardware and software described herein. Such comments should also be sent to **support@scali.com**.

10.6.4 Keeping informed

Apart from regular visits to the Scali web site Scali also recommends subscription to the two mailing lists maintained by Scali. These are **scali-announce** which is where Scali posts the latest news about Scali products, releases, patches and so on, and **scali-user** which is a discussion group for Scali users. Instructions on how to join the appropriate mailing list can be found on **<http://www.scali.com/support>**.

A-1 General information

A-1.1 Scali software platform CD-ROM

The Scali Software Platform (SSP) CD-ROM contains software packages for all Scali products needed to get a Scali system installed and operative. Below is a description of the top level file system layout on the SSP CD-ROM.

- `README` : Start by reading this!
- `NEWS` : The SSP release notes
- `/doc` : documentation.
- `install` : The Scali software installation program.
- `uninstall` : The Scali software removal program.
- `/pkg` : All the software distribution files.
- `/bin` : binaries used during installation.

A-1.1.1 Installation for the impatient

Before starting the installation program you should check out the online documentation for hardware configuration in `/doc/HW`, and operating system prerequisites in `/doc/HW` for the latest information which may not have made it into the manual. Hardware installation is described in detail in chapter 3. Operating system configuration and Scali software installation is described in chapter 4. When you are ready to start the installation type the following (as root):

```
# ./install
```

The installation program will guide you through the installation process which will install SSP components according to your licenses on the whole cluster. The installation program will detect if this an upgrade and re-use configuration settings as appropriate. Note that the SSP installation program provides a number of useful command line switches (see “SSP install program options” on page 40). One example is the ability to re-run parts of the installation, like the test section with `-t`:

```
# ./install -t
```

Appendix A: Scali software details

A-1.1.2 Uninstall

If you wish to remove the SSP at a later stage you can run the uninstall program (as root):

```
# ./uninstall
```

This will remove all traces of the SSP on all nodes.

A-1.2 Scali software directory structure

All Scali software is installed under the directory `/opt/scali`. The top level directory structure is as follows:

- `/bin` : all normal executables to be run by users.
- `/sbin` : all daemons and executables to be run by administrators.
- `/libexec` : executables which are hidden from normal invocation (e.g. used by applications under `/bin`).
- `/lib` : libraries used by our applications and by end users.
- `/include` : include files for libraries under `lib`.
- `/doc` : all documentation.
- `/etc` : configuration files for Scali software.
- `/examples` : example MPI applications and source.
- `/contrib` : 3rd party software adapted to Scali software.
- `/init.d` : boot/startup scripts. Soft links are made from system startup catalogues (`rc.d` and `init.d` under `/etc`) by install script.
- `/kernel` : kernel related files.
- `/man` : manual pages
- `/plugins` : plugins for the Scali Universe GUI
- `/share` : share directory used by SNMP clients/agents

A-2 Scali package file naming convention

Every software unit is distributed as a single package file:

```
module.os.arch-x.y.z.package
```

where:

- | | |
|----------------|--|
| module | is the software name e.g. ScaMPI, ScaSCI, ScaPkg. |
| x.y.z | is the release number, e.g. 1.0.2, |
| os | is the operating system, e.g. SunOS5 or Linux2 |
| arch | is the architecture, e.g. sparc-u, i86pc or alpha |
| package | is e.g. pkg, rpm or exe depending on the operating system. |

A-3 Scali software packages overview

The `/pkg` directory on the SSP distribution contains all necessary software packages for the entire Scali product range: Universe, Universe XE and ClusterEdge. Hence, depending on your license and product, some of the packages in the table below may not have been installed on your system.

Package (module) name	Description
ScaComd	Communication daemon, provides a communication backbone for clusters
ScaConfC	Interconnect configuration clients: scaconftool
ScaConfNd	Interconnect configuration node daemon
ScaConfSd	Interconnect configuration server daemon
ScaCons	Console server package
ScaDesk	Scali Universe graphical interface
ScaDiag	Diagnostic tools
ScaDiagSC	Diagnostic tools for SCI systems
ScaEnv	Environment settings
ScaExecd	Remote execution daemon
ScaFmpich	Free MPICH binary packaging
ScaFPDisp	Front panel display driver (LEDs)
ScaFPMB2	Free PMB2 binary packaging
ScaIP	Internet Protocol driver for SCI
ScaIPadap	ScaIP adaption kit
ScaLM	License manager
ScaMAC	Media driver for SCI
ScaMACadap	MAC driver adaption kit

Table 10-5: Overview of Scali software packages distributed with the SSP.

Appendix A: Scali software details

Package (module) name	Description
ScaMACddk	MAC driver development kit
ScaMond	Monitoring server daemon
ScaMPE	MPE packaging
ScaMPI	High performance MPI implementation
ScaMPICHT	MPICH test packages
ScaMPItst	ScaMPI test packages
ScaOPBS	Open PBS binary packaging
ScaPkg	Cluster software package installation tool
ScaPowd	Remote power switch control daemon
ScaSCI	SCI driver
ScaSCIadap	ScaSCI adaption kit
ScaSCIddk	ScaSCI driver development kit
ScaSensor	Sensor configuration package
ScaSH	Parallel shell tools
ScaSHMeme	Compatibility library for Cray/SGI ShMem
ScaSISCI	SISCI interface to SCI driver
ScaSNMPd	SNMP daemon
ScaSNMPt	SNMP text tools
ScaSSP	the Scali Software Platform main package, contains install programs and documentation.

Table 10-5: Overview of Scali software packages distributed with the SSP.

A-4 Scali software daemon overview

After a successful installation of the SSP there will be quite a few new daemons running on your system. Common to all is that they can be started/stopped/restared using the standardised scripts located in `/opt/scali/init.d/` using the following command:

```
# /opt/scali/init.d/<daemon> [start|stop|restart|info]
```

Note that if one of your nodes doubles as front-end it will run all daemons.

daemon	location	description
mpid	node	mpi application launcher, please refer “ScaMsPI Users Guide”
scacomd	node	Scali communication daemon, provides communication services for other Scali software
scaconfsd	front-end	Scali configuration server daemon, handles interconnect configuration, and remote power switching
scaconfnd	node	Scali configuration node daemon, this is the interface between the configuration system and the node interconnect hardware
scaexecd	front-end	Scali remote execution daemon, provides secure remote execution
scamond	front-end	Scali monitoring server daemon, provides compression and filtering of monitoring data from the SNMP daemons to its clients
scasmpd	node	Scali SNMP daemon, this daemon provides the monitoring system with information.

Table 10-6: Scali daemon overview

A-5 Scali configuration file overview

Configuration files for Scali tools and programs will always be stored in

`/opt/scali/etc`

Files with the extension `.example` are example configuration files installed by some packages. The table below gives a summary of configuration files.

file	description
<code>scaConf.conf</code>	Configuration file for ScaConf the cluster configuration system.
<code>scaConf.nodeidmap</code>	If present this file ensures node names are mapped to specific node Id's
<code>scaDesk.conf</code>	Default configuration file for Scali Desktop, will be copied as basis for the personalised user profile in <code>\$HOME/.scali</code>
<code>scaMond.conf</code>	Scali monitoring daemon configuration file
<code>scaPkg.conf</code>	Scali software installation system configuration file. Lists packages, node categories and the relation between the two.
<code>scaPkg.adminfile</code>	Solairs specific configuration file for pkgadd
<code>scaSH.conf</code>	ScaSH parallel shell tools configuration file.
<code>scaSH.scahosts</code>	Host list for ScaSH parallel shell tools
<code>conserver.conf</code>	ScaCons console server configuration file
<code>conserver.passwd</code>	Password file for the ScaCons console server

Table 10-7: The configuration files in `/opt/scali/etc`

Appendix B Scali Software Licensing

B-1 Introduction

All necessary components of the Scali software licensing system will be installed and configured during SSP installation by the **install** program. If you for some reason should run into problems, or need to alter licenses manually, this appendix explains the Scali licensing system in more detail.

B-2 Requesting licenses

Your first experience with the Scali licensing system will be when you need the license to enable the initial SSP installation. Normally this is handled using the procedure described in "B-2.1 Automated demo licenses generation from customer ID tag". Please note that different E-mail addresses are used for the different types of license requests.

B-2.1 Automated demo licenses generation from customer ID tag

To enable license generation outside office hours, and to speed up license generation in general, Scali has set up an automated demo license generation service. This service provides our customers with demo licenses of 10 days validity, 24 hours a day, 7 days a week. Using this service requires your private customer identification tag which can be found on your package list or invoice. The format of the tag is like this:

`SDxxxxxxx`

where xxxxxxx may be any letter or number combination. To request a demo license using your customer ID tag, just create an E-mail where the subject field is set to the tag number (only the 'SDxxxxxxx' part), and send it to **demolicense@scali.com**. Within a few minutes you should receive a new demo license. Note that you must send this E-mail from a working E-mail account.

B-2.2 Demo licenses - general requests

Scali will also provide demo licenses for evaluation purposes. Since demo licenses are valid for all hosts until the expiration date, no system information is required. You may send a simple E-mail request or use the "License Request" form in the support section of the Scali web-site. General demo license requests should always be directed to **sales@scali.com**. If your request is granted you will most probably receive a customer ID tag which can be used to get demo licenses following the procedure in B-2.1

Appendix B: Scali Software Licensing

B-2.3 Permanent licenses

Permanent licenses for Scali software are node-locked, therefore, in order to receive a permanent license, you will need to supply Scali with the hostid for all nodes in the system, including the hostname and hostid of the front-end and some other technical details. You must also provide:

1. Resellers company name (if applicable)
2. License owner's company name.
3. E-mail address to recipient of license keys.
4. Your order reference number

Requests for permanent licenses should always be sent to: **license@scali.com**.

B-2.3.1 Permanent license request format example

Here is an example of a correctly formatted license request for a 4 node system:

```
-----  
SSP_3_0_1 (version: 1.94) license request  
  
Date : Mon Feb 18 14:42:27 CET 2002  
Resellers company : N/A  
Lic. owner company : Scali Scalable Linux Systems AS  
Recipient (email) : user@scali.com  
Order reference no : SC1234  
  
Frontend machine : scali1-11  
Domain name : scali.com  
  
Current license : Feature Version Expire Type  
                  clusterededge 3.000 1-mar-2002 Demo  
-----  
Host Hostid #Cpu Kernel Distro  
scali1-11 0080c9876b1a 2 2.2.19-7.0.8smp Red_Hat_7.0_Guinness  
  
scali1-11 0080c9876b1a 2 2.2.19-7.0.8smp Red_Hat_7.0_Guinness  
scali1-12 00a0c9976d57 2 2.2.19-7.0.8smp Red_Hat_7.0_Guinness  
scali1-21 00a0c9876bc1 2 2.2.19-7.0.8smp Red_Hat_7.0_Guinness  
scali1-22 00a0c9876acb 2 2.2.19-7.0.8smp Red_Hat_7.0_Guinness  
-----
```

B-2.3.2 Generating the request during installation

Fortunately, the SSP **install** program will generate and format a license request as part of the installation. You will be asked to provide the customer details while the necessary technical information is extracted automatically from the system. Finally you will be given an option to send the license request right away. The file containing the license request will be placed in:

```
/tmp/SSP_3_0_1.licrequest
```

This is for SSP 3.0.1, the SSP version number will of course change to equal the current distribution.

B-2.3.3 Generating a new license request

If you wish to generate a new license request at a later time, simply run the now installed **SSPinstall** program with the '-l' option and follow the instructions.

```
# /opt/scali/sbin/SSPinstall -l
```

B-2.3.4 Last resort

If you fail to run the SSPinstall program, you can collect the necessary host IDs for the license request manually by running a small ScaLM utility program called **lmhostid** on every node. The **lmhostid** is part of the ScaLM package and can be used in combination with **scash** to collect the hostid's for the entire system like this:

```
# /opt/scali/sbin/scash -pbw /opt/scali/bin/lmhostid
scali1-11      : The host ID of this machine is "0080c9876b1a"
scali1-12      : The host ID of this machine is "00a0c9876d57"
scali1-21      : The host ID of this machine is "00a0c9876bc1"
scali1-22      : The host ID of this machine is "00a0c9876acb"
```

Note that if you have a separate frontend you must remember to run **lmhostid** on this separately, because it will not normally be included in the **scash** host list.

B-3 The license file

B-3.1 Default location

The default license file for Scali software is:

```
/opt/scali/etc/license.dat
```

This file holds the licenses for all Scali software and a copy must be distributed to the same location on *all* nodes in the system.

Appendix B: Scali Software Licensing

B-3.2 SCALM_LICENSE_FILE environment variable

The SCALM_LICENSE_FILE environment variable may be used to alter the name and location of the license file. Simply set it to the full path of the license file. If SCALM_LICENSE_FILE contains a valid file name, ScaLM will look for licenses in this file before the file in the default location.

B-3.3 License file example

After generation by the SSP **install** program, or after manual editing, the license file could look something like this for a 4 node system with ClusterEdge product, additional support for large SMPs with ScaMPI and support:

```
#
# Scali AS - license file
#

# ClusterEdge license
FEATURE clusteredge scald 3.000 permanent 4 \
12ED7FE94B27402DFBAB30A645F794075F5BA2C580514DDAE421B000C8 \
HOSTID="0034567966b 00345677f4c 003456764c 0034457eac0"

# Large SMP support for ScaMPI
FEATURE mpismp scald 1.910 permanent 4 \
2123EF271B000FF12587EE12345DFBAB30A645F794543F5BA2C5805143 \
HOSTID="0034567966b 00345677f4c 003456764c 0034457eac0"

# Support
FEATURE support scald 1.000 1-apr-2003 1 \
97152A67768551B00052F7230301B1DBCFE8B58D31C988AB51E3C6F3DB \
HOSTID="0034567966b"
```

Lines starting with # are comments. You may have several FEATURE lines, one for each licensed software product or add-on.

B-3.4 Adding or updating features (licenses)

When you receive a new software license from Scali, be it a demo or a permanent license, this will be in the form of FEATURE lines. The FEATURE lines should be inserted in the license file: `license.dat`, replacing older ones for the same feature. When receiving a multi-line FEATURE it is important to keep the format exactly as received from Scali, including whitespaces and "\", as it otherwise might not work.

B-4 License installation

When a license file is in place it must be installed (or distributed) to all nodes in the system. During new SSP installations or SSP upgrades the install program will query you for FEATURE lines or a license file, and automatically generate and distribute the license file to the right locations. The problem is that you seldom have your permanent license at this point in time so you will probably have to install a new license for your existing installation later. This can be done in one of two ways:

B-4.1 License update/installation with SSPinstall

Starting with SSP 3.0.0 the **SSPinstall** program has a `-u` option for upgrading a license file. Using SSPinstall handles all aspects of license installation and is now the preferred way to install new licenses. Just do a

```
# /opt/scali/sbin/SSPinstall -u
```

and follow the instructions. This will run the entire license entry part of the SSP installation all over, so you can again chose to enter the license as a file or as separate FEATURE lines. Please refer to "4.3.2.2 Installation Configuration" for an example of license entry.

B-4.2 Manual license update/installation

If you fail to use the SSPinstall you may still install the license manually using the **scarcp** utility from ScaSH. Assuming you are on the front-end with an updated license file in the default location, you may copy the license file to the same location on all nodes with the command:

```
# /opt/scali/bin/scarcp /opt/scali/etc/license.dat
```

After the new license file has been installed you must also restart the daemons that depends on licensing. Assuming that you're still on the front-end you can restart the ScaMon daemon with the command:

```
# /opt/scali/init.d/scamond restart
```

If you have SCI installed you must also restart the ScaConf server daemon with the command:

```
# /opt/scali/init.d/scaconfsd restart
```

B-5 Meta-licensing

Starting with SSP 3.0.0 Scali introduced the concept of meta-licensing to simplify license handling. Using meta-licensing each Scali core product (Universe, Universe XE, ClusterEdge) will need only one FEATURE line in the license file. This single FEATURE line will unlock all individually licensed sub-components of the product. This differs from SSP 2.X releases where you would need a FEATURE line for every component.

The file `/opt/scali/etc/metalicence.dat` controls the different meta-licenses. Please note that there is an unchangeable one-to-one relationship between SSP releases and the meta-licensing file. End-users should never attempt to do *anything* with this file. Changing the `metalicence.dat` file will make your system useless.

B-6 Troubleshooting

If your program fails to start due to a license problem this will be clearly indicated by an error message logged to the system logfile. The log files are: `/var/log/messages` for Linux and `/var/adm/messages` for Solaris.

B-6.1 Error: Invalid FEATURE line

This means that the license file found by the application does not contain any valid FEATURE lines entries, or that the encryption key of the FEATURE is invalid.

B-6.2 Error: Invalid version x.x > y.y

This means that the version of the FEATURE you're trying to check out is newer than the one you have in your license file. The most common reason for this is that you are trying to use an outdated copy of the license file. This may happen if you upgraded your software and forgot to update the license file.

B-6.3 Error: FEATURE has expired

This error occurs only with time limited licenses and tells you that the license for this FEATURE has expired. The only time limited license used by Scali software are DEMO licenses

B-6.4 Error: Host ID is not found!

This error occurs with host bound licenses only and means that the requested FEATURE is not valid for this host. Most Scali software licenses are host-bound.

B-6.5 Missing license file

Naturally a missing license file will create problems. To ensure you that the license file has been properly distributed to all nodes you can use **scash** like this:

```
# /opt/scali/bin/scash -pwb ls -l /opt/scali/etc/license.dat
```

B-6.6 Missing license software - ScaLM package

You may also experience problems if the Scali licensing software is not properly installed. The Scali licensing software is distributed in the package ScaLM which must be installed on *every* node in the system including the front-end. To check that this is really the case use scash from the front-end and do a:

```
# /opt/scali/bin/scash -pwb rpm -q ScaLM
```

If you find ScaLM to be missing on some nodes the most efficient way to fix this is to use the **scapkg** software package installation utility. The command

```
# /opt/scali/sbin/scapkg -p ScaLM
```

will make a system wide installation of ScaLM. Please refer to "Appendix E ScaPkg - Scali Software installation program" for more details about **scapkg**.

B-7 Older versions with FLEXlm

Note: Prior to release 2.1 of the SSP, Scali were using the FLEXlm license manager system from Globetrotter Software Inc. If you are experiencing problems regarding FLEXlm please refer to this section of a version 2.0 or older version of the "Scali System Guide". Older versions of the System Guide are available from the download section on Scali's web-site. The FLEXlm End User's Manual is also available online from Globetrotter Software Inc. at:

```
http://www.globetrotter.com/TOC.htm
```

Appendix B: Scali Software Licensing

C-1 SCI hardware status programs

This chapter will describe some SCI utility programs. These programs are located in the `/bin` and `/sbin` directories of the Scali installation. They are useful for testing and monitoring the SCI hardware and of limited interest to the ordinary user, *and should not be used under normal circumstances*.

C-1.1 sciping

The **sciping** program checks the reachability of SCI nodes through an adapter. Default adapter number is 0. The most important options are listed below. The sciping command will send an SCI packet via the SCI interconnect from the node it is run on to the nodes with the nodeid specified on the command line. This utility is useful for testing connectivity between nodes.

Usage:

```
/opt/scali/bin/sciping [-L|[-x|-t|-s]<tag>] <nodeid0> <nodeid1> ..
```

Options:

-a <adapter>	Use adapter number 'adapter' instead of adapter 0
-m <count>	Send count requests for each node on command line
-c <sec>	Continuous mode. Repeat requested pings every sec seconds
-p	Print performance statistics (t/ping)
-u	Allow loop back pings (potentially unsafe)
-h	Print this text
-v	Only print nodes that fails to respond/links that are down
-L	Write a log message to the system log at both nodes (requires all involved nodes to run a similar sciping)

Appendix C: SCI Utility Programs

C-1.2 scimonitor

The **scimonitor** program watches link status changes for an SCI adapter. The default is adapter number 0.

Usage:

```
/opt/scali/bin/scimonitor [-a<adapter>]
```

[Options]:

-a <adapter> Adapter number to monitor.

C-1.3 sciemsg

The **sciemsg** script maps a return value from a call to the SCI API or a symbolic error type to the corresponding error message. See Table Table 10-3 on page 142 for some examples of symbolic SCI error types.

Usage:

```
/opt/scali/bin/sciemsg <SCI status error message number>
```

C-1.4 scidbx

The **scidbx** program is a text based command line tool for inspection and manipulation of the SCI hardware through the SCI driver. When started, scidbx will prompt you with **#scidbx>** ready to accept commands. Type **help** for further information about available commands in **scidbx**. The **scidbx** program is only available to root.

Usage:

```
/opt/scali/sbin/scidbx [-a<adapter>][-f]
```

[Options]:

-c or **-e** <string> Execute <string> as scidbx command in batch mode.
-f Force start even when adapter sanity checks fails.
-h print this text.

C-1.5 scideb

The **scideb** program will set the debug level of the SCI driver. This setting can change the amount and character of SCI driver information printed in the system log files.

Usage:

```
/opt/scali/sbin/scideb [-V] <new debug level>
```

[Options]:

-V Enable verbose mode

C-1 SCI hardware status programs

Possible levels (in any combination):

SSCI_INFO	0x1	/* Various infrequent info */
SSCI_ICM	0x2	/* Use of KalLog in OS independent code */
SSCI_CFG	0x4	/* Configuration info */
SSCI_UE	0x8	/* Details about user errors */
SSCI_CR	0x20	/* Connect and release details */
SSCI_ENT	0x40	/* Entry and exit of entry points */
SSCI_CB	0x80	/* Entry and exit of callbacks */
SSCI_FABRIC	0x100	/* Interconnect fabric related conditions */
SSCI_PKT	0x200	/* SCI software packet interface */
SSCI_AEH	0x400	/* Asynchronous error handling */
SSCI_MSEG	0x800	/* Memory segment details (not too verbose) */
SSCI_MMAP	0x1000	/* Entry in mmap */
SSCI_CH	0x2000	/* Channel interface */
SSCI_KAL	0x4000	/* Enter and exit of Kal funs except mutexes */
SSCI_SAL	0x8000	/* Suballocator */
SSCI_ICME	0x10000	/* Use of KdPrint in OS independent code */
SSCI_INTR	0x20000	/* User interrupt processing */
SSCI_SOFT	0x40000	/* DDI_soft */
SSCI_MEM	0x80000	/* Memory allocation and export process */
SSCI_MCTX	0x100000	/* Mmap context adm */
SSCI_PG	0x200000	/* Page pool adm.info */
SSCI_CH_V	0x800000	/* Verbose printout for channel interface */
SSCI_RCHUNK	0x2000000	/* Remote connect/disconnect details */
SSCI_LCHUNK	0x4000000	/* Local chunk maintenance details */

C-1.6 scinode

The **scinode** program prints/sets nodeid of the specified adapter. Default is to print the nodeid of adapter number 0 on standard output. Only root may set nodeid. The **scinode** options are listed below.

Usage:

```
/opt/scali/sbin/scinode [-h][-v][-s <new nodeid>][-a <adapter no>]
```

Options:

- h** Print this help string
- v** Print version information
- a<n>** Use adapter number <n>
- s<id>** Sets new nodeid for the adapter

Appendix C: SCI Utility Programs

C-1.7 scinfo

The **scinfo** utility gives global information and statistics for all SCI instances, and are particularly useful as an aid in error diagnostics.

Usage:

```
/opt/scali/sbin/scinfo [-v][-a <adapter number>]
```

[Options]:

- m** show memory and att usage statistics
- n** show processes with active SCI connections
- l** show SCI link statistics
- p** show pkt statistics (driver-to-driver comm.)
- c** show driver configuration and version
- i** show general interrupt statistics
- v** all of the above (equal to -mlpci)
- h** print help text

C-1.8 scireconf

The **scireconf** program may be used to reconfigure the Scali SCI driver. This utility is only to be used upon instructions from Scali.

Usage:

```
/opt/scali/sbin/scireconf [-d] parameter [value]
```

[Options]:

- d parameter** Delete (comment out) the parameter setting
- parameter** Display current setting if any
- parameter value** Change (and enable) parameter with this value

C-1.9 scireload

The **scireload** command will reload the Scali SCI driver on the machine the command is run on. This command is only available to root, and it should not be necessary to use this command under normal circumstances.

Usage:

```
/opt/scali/sbin/scireload
```


C-1 SCI hardware status programs

C-1.10 **scidle**

The **scidle** utility returns status code 0 if SCI links are ok and idle.

Usage:

```
/opt/scali/sbin/scidle [-a <adapter no>]
```

[Options]:

-a n Use adapter number *n* (default 0)

C-1.11 **scicards**

The **scicards** script prints the number of SCI adapter cards installed and recognised by a node.

Usage:

```
/opt/scali/sbin/scicards
```

Appendix C: SCI Utility Programs

D-1 Installation and package dependencies

Installation of the Scali interconnect configuration system is normally handled by the SSP install program, but the table below gives you a list of what to look for in case of problems. Note that the ScaSH package must be installed on the frontend. .

Name	Description
ScaComd	Communication daemon (node)
ScaConfNd	Configuration daemon (node)
ScaConfSd	Configuration server (frontend)
ScaConfC	Scaconftool, ASCII based configuration tool (frontend)
ScaSH	Parallel remote shell interface to nodes (frontend)

Table 10-8: ScaConf packages

D-2 scaconftool - command reference

This section describes all available **scaconftool** commands and their syntax. The `<list>` parameter is essential. `<list>` is to be replaced with a set of nodes. For commands where a list of nodes are required, it is at least five different ways to state these nodes. For further information about node selection, read section 7.3.2.3. `<list>` is to be replaced by:

- keyword **all**, meaning all nodes.
- node state, meaning all nodes in this state (e.g OK, NO_DAEMON etc.)
- list of node names separated by blanks, meaning all nodes explicitly listed.
- left blank, meaning all nodes in internal list.
- routing partition number, e.g. #1, meaning all nodes with this routing partition number.

Appendix D: ScaConf Reference

D-2.1 ConfTool> console

The *console* command is used to connect to the console on a specified node. This command only works if a console server is available and the ScaCons package is installed correctly. The `<console options>` are passed as is to the program `/opt/scali/bin/console`. See 'man console' for an explanation of the console program.

Usage:

```
console <console options>
```

Example:

```
console node1
scaconftool will try to connect to console on node1.
```

D-2.2 ConfTool> daemon

The *daemon* command is used to start or stop daemons on nodes.

Usage:

```
daemon <daemon name> <action> [<list>]
```

Example:

```
daemon scacomd restart UNREACHABLE
Restart the communication daemon on all nodes in state UNREACHABLE
```

Example:

```
daemon scaconfnd restart NO_DAEMON
Restart the configuration daemon on all nodes with state NO_DAEMON
```

D-2.3 ConfTool> fix

The *fix* command attempts to bring the nodes in the list into state OK. This is achieved by restarting daemons that does not run, reloading essential drivers and reconnecting the nodes to configuration server.

Usage:

```
fix [<list>]
```

Example:

```
fix UNREACHABLE
scaconftool will attempt fix all nodes with status UNREACHABLE.
```

D-2.4 ConfTool> getopt

The *getopt* command lists server options and their values. If the list of options is empty, all available options will be listed. Some of the options are defined as ACTION. These options does not have a value, but represents actions that the server will perform when activated with the *setopt* command.

Usage:

```
getopt [<option> .. <option>]
```

<option>:
Can be replaced by any legal options.

Example:

```
getopt
getopt AUTO_REPAIR
```

D-2.5 ConfTool> setopt

The *setopt* command sets the value of the specified server options. Some options are defined as ACTION. Applying setopt on this type of option, will cause the configuration server to perform the action. Saving options to file and restoring factory defaults are available actions.

Usage:

```
setopt <option=value> [<option=value>... ]
```

<option=value>:
Can be replaced by any legal options. Use getopt with no parameters to get a list of all legal options. Options are described in Table 10-9:

Example:

```
setopt AUTO_REPAIR=OFF AUTO_REROUTE=OFF
```

OPTION	VALUE	DESCRIPTION
AUTO_REPAIR	ON*/OFF	Turn auto repair on or off. When on, server will automatically poll erroneous nodes to check if they are ok, and reconnect them if they are.
REPAIR_INTERVAL	[3,300]	Server will wait for this many seconds before it will try to reconnect nodes that is reported as <i>UNREACHABLE</i> . Recommended value: 10*
AUTO_REROUTE	ON*/OFF	Server will automatically reroute cluster if change of state in one or more nodes is detected.
AUTO_RECONNECT	ON*/OFF	Nodes will be reconnected to the cluster automatically when they come up if this option is ON.
AUTO_NODE_ID	ON*/OFF	Automatic checking and setting of SCI node id on or off.
ROUTING_TYPE	SCA_ROUTE_DEFAULT*	Default routing type for the topology. Use 'info routalg' to get a list of all routing type available for your topology.

Table 10-9: Available server options. Default values are marked with *

Appendix D: ScaConf Reference

OPTION	VALUE	DESCRIPTION
REROUTE_DELAY	[0,..,3*,.,10]	Seconds to wait after a reroute is issued before new status is sampled.
SAVE	<none>	Save the options to file. This will cause server to start with the saved settings if restarted.
RESTORE	<none>	Restore factory defaults for all options

Table 10-9: Available server options. Default values are marked with *

D-2.6 ConfTool> info

The *info* command prints information on specified topics.

Usage:

```
info <topic> ... <topic>
<topic>: topology      - prints the topology of the machine.
        routalg       - prints the available routing algorithms.
```

Example:

```
info topology routalg
```

D-2.7 ConfTool> select

The *select* command adds nodes to the list of active nodes.

Usage:

```
select <list>
```

Example:

```
select scali2-11 scali2-12
select all
select NO_DAEMON
```

D-2.8 ConfTool> unselect

The *unselect* command removes nodes from the list of active nodes.

Usage:

```
unselect <list>
```

Example:

```
unselect scali2-13 scali2-12
unselect all
unselect UNREACHABLE
```

D-2.9 ConfTool> list

The *list* command displays various information about nodes.

Usage:

```
list [<format>] [<list>]
or
list configuration [<stacked link>]
<format>::
[ [status] [nodeid] [L0] [L1] [L2] [links][partition] [position] ] | configuration
status:      Status for the node.
nodeid:      Node identification for SCI devices.
L0:          Status for link 0 .
L1:          Status for link 1 (if available).
L2:          Status for link 2 (if available).
partition:   Routing partition number.
position:    X/Y/Z coordinates.
Node name is always printed. Default is that only nodeid and node name is
printed. The format is stored, and will be used if the next list-command does
not specify any format.
```

configuration:

List the configuration of the cluster in a matrix representation reflecting the topology of the cluster. On 3D systems, to display the system in a flat view, one link is “stacked”, and what you see is “layers” where the links connecting these layers are not shown. The <stacked link> parameter decide which link that is not shown. Default is link 2.

Display formats:

```
Name:      Node name
Nodeid:    Hex number
Status:    OK indicates that everything is ok.
           UNREACHABLE indicates the node is down or unreachable from
           ScaConf's point of view. The reason might be no power or that
           scaconfd does not run on this node.
           NO_DAEMON indicates the ScaConf configuration daemon is
           not running on the node.
           NO_SCI_DEVICE indicates the SCI driver is not loaded.
           UNKNOWN is an initial state, indicating that no information is
           received from the node yet.
Link0:     EN indicates that SCI link controller is enabled.
           DIS indicates that SCI link controller is disabled.
           UP indicates that the SCI link is up.
           DOWN indicates that the SCI link is down.
Link1:     (same as Link0)
Link2:     (same as Link0)
Pos:       Tuple/triple coordinates for 0,1 and 2 link to indicate where in
```

Appendix D: ScaConf Reference

the topology the node is located.

Part: Integer, routing partition number.

Example output:

```
ConfTool> list nodeid status L0 L1 all
Name      NodeId Status  Link0 Link1
scali2-11 0x1100 OK     EN/UP  EN/UP
scali2-12 0x1200 OK     EN/UP  EN/UP
scali2-13 0x1300 OK     EN/UP  EN/UP
scali2-14 0x1400 OK     EN/UP  EN/UP
```

Example:

```
list
```

Lists information on nodes in built-in list. Use select to add nodes to the internal list. The format from previous list command will be used:

Example:

```
list nodeid status L0 L1 scali2-11 scali2-13
```

Lists nodeid, status and links states for scali2-11 and scali2-13.

Example:

```
list status all
```

List status for all nodes.

D-2.10 ConfTool> fail

The *fail* command turns off nodes in the sense that status become UNREACHABLE.

Usage:

```
fail [<list>]
```

Example:

```
fail
```

Fail nodes in the active list.

Example:

```
fail all
```

Fail all nodes.

D-2.11 ConfTool> reconnect

The *reconnect* command does a reconnect to specified nodes.

Usage:

```
reconnect [<list>]
```

Example:

```
select UNREACHABLE
```

```
reconnect
```

Attempts to reconnect all nodes with state UNREACHABLE.

D-2.12 ConfTool> log

The *log* command is used to enable or disable server log messages in the tool.

Usage:

```
log [enable/disable]
```

Example:

```
log disable
Disable log messages from the server.
```

Example:

```
log enable
Enable log messages from the server.
```

D-2.13 ConfTool> nodeid

The *nodeid* command is used to set SCI nodeIDs.

Usage:

```
nodeid [<node name> <id>]
<id>: The id can be either decimal, octal (leading 0) or hex (leading 0x).
Without arguments nodeid will be set to default for all nodes.
```

Example:

```
nodeid
Set nodeid to default for all nodes.
```

Example:

```
nodeid scali2-24 0x2400
Set nodeid for node scali2-24 to 0x2400.
```

D-2.14 ConfTool> reload

The *reload* command does a reload of the SCI driver on the nodes.

Usage:

```
reload [<list>]
```

Example:

```
reload scali2-22
Reload SCI-driver on node scali2-22. Please note that a reload of the SCI
driver also implies that several other daemons are restarted (e.g. scid, mpid,
scaconfnd, scasnmpd).
```

D-2.15 ConfTool> reroute

The *reroute* command does reroute the cluster with specified algorithm. Use '*info routalg*' to get a list of available routing algorithms for your topology.

Usage:

```
reroute [routing algorithms]
```

Example:

Appendix D: ScaConf Reference

`reroute maxcy`

Use the maxcy routing algorithm to route the 2d torus.

D-2.16 ConfTool> status

The `scaconftool` command `status` prints information about nodes or links according to the cluster configuration. The output looks like the `list configuration` command, but instead of node name, a three letter code is indicating the status of the node when `nodes` is the argument. Enabled/disabled or up/down information is displayed if `controllers` or `links` is the argument. In 3D torus topology, the torus is 'stacked' into layers. The `<stacked link>` parameter specify which link that is not shown in the output. Default is link 2.

Usage:

```
status [nodes|controllers|links] <stacked link>
```

Example:

```
status nodes
```

Print status information for each node.

Example:

```
status controllers
```

Print information about enabled/disabled links.

Example output for `node` argument which display three letter status codes:

```
  1      2      3      4
UNK ---000 ---000 ---000
|      |      |      |
... ---000 --- 000 ---0..
000      : The node is OK.
...      : The node is UNREACHABLE.
0..      : The node has no node daemon (NO_DAEMON).
00.      : The SCI driver is not loaded (NO_SCI_DEV).
INV      : This is an invalid state, and should never occur.
UNK      : The node is in state UNKNOWN.
```

Example: Controller/link output with display codes.

```
  ?      ?      1      1
? ? --? ? --0 0 --0 0
?      ?      1      1
|      |      |      |
0      0      1      1
1 1 -- 1 1 --1 1 -- 1 1
0      0      1      1

0      : disabled/down
1      : enabled/up
```

? : unknown/unknown

D-2.17 ConfTool> link

5. The *link* command disables or enables links.

Usage:

```
link <[disable | enable]> <controller number> [<list>]
```

Example:

```
link disable 0 all
Disable controller 0 on all nodes.
```

D-2.18 ConfTool> update

The *update* command ask the configuration server for updated system information.

Usage:

```
update [ <list> | system]
```

Example:

```
update scali2-13 scali2-23
Ask for updated information on nodes scali2-13 and scali2-23.
```

Example:

```
update system
Ask for new information on whole system. This command is more or less
obsolete, as updated system information is received each time the server is
contacted, that is for nearly every command performed from scaconftool.
```

D-2.19 ConfTool> restart

The *restart* command does an on-line restart of the configuration server.

Usage:

```
restart
```

D-2.20 ConfTool> sciping

The *sciping* command pings nodes over SCI interconnect. It uses scash.

Usage:

```
sciping [<list>]
```

Example:

```
sciping node1 node2 node3
send sciping from the listed nodes to the listed nodes.
```

D-2.21 ConfTool> help

The *help* and *?* command print a help message.

Usage:

```
help/? [<command>]
```

Example:

```
help/? restart
```

Appendix D: ScaConf Reference

Print information about the *restart* command.

Example:

```
help/?
```

Print a list of all available commands.

D-2.22 ConfTool> quit

The *quit* command exits the command tool.

Usage:

```
quit
```

D-2.23 ConfTool> version

The *version* command prints the version of scaconftool.

Usage:

```
version
```


Appendix E: ScaPkg - Scali Software installation program

- s Enable "sync" mode. scapkg will then try to install and remove packages in order to reflect the setup in the config file (`/opt/scali/etc/ScaPkg.conf`). The need to remove packages can occur if one has manually installed packages.
- f Enable "force" mode. This will skip the version test and install it anyway.
- e Enable "erase" mode. This will remove packages. This switch may not be used in combination with "sync" mode.
- D Enable dryrun mode. No action taken, but everything is checked.
- d Enable debugging mode. This will generate logfiles under /tmp.
- S Enable sequential mode. The default is to run as much as possible in parallel.
- F Specify fanout factor other than default for `scash` and `scarcp`.
- v Enable verbose mode.
- V Show version.
- h/? Usage information.

It is possible to use **scapkg** for handling general packages (i.e not Scali specific packages). A general package may not have the string "Sca" as prefix in its filename. Another restriction for general packages is that the first part of the filename should be identical with the name of the package

E-2 Configuration

The ScaPkg configuration file is located in:

```
/opt/scali/etc/ScaPkg.conf
```

The configuration file will be created as part of the SSP installation process and contains information for **scapkg** which packages belongs to which categories. Each node will be assigned packages according to the category to node mappings specified in the configuration file. Changes to the configuration file can be made either manually with a text editor or as a result of running the SSP installation program. The ScaPkg configuration file is divided into five logical sections which are described in sections E-2.1 through E-2.5.

E-2.1 ScaPkg.conf - path to package files (repository)

The path-to-repository section defines a path to the directory holding the software packages (repository). Note, an architecture identification string will be appended at the end of the path corresponding to the output from the **systype -c** script from the ScaEnv package when searching for packages made by Scali. For example, if `/usr/local/src/pkg` is the repository, a Linux i86pc system would expect to find package files in `/usr/local/src/pkg/Linux2.i86pc` and an UltraSPARC system

would expect to find package files in `/usr/local/src/pkg/SunOS5.sparc-u`. When searching for general packages, (not Scali packages), the search order would be `/usr/local/src/pkg/Linux2.i86pc` and then `/usr/local/src/pkg` if the package was not found at the first search location. It is possible to specify more than one `path-to-repository` locations. The search order for each package will be as listed in the `path-to-repository` section of the configuration file. The `path-to-repository` specified in the configuration file may be overridden with the `-r` option.

E-2.2 ScaPkg.conf - package list

The package list section defines which packages are associated with which categories. Each line in the package list is on the format:

```
package <name> <categories>
```

where `<name>` is the package module part, and `<categories>` is one or more categories separated by space. The categories are freely defined and are referred to in the Super-categories section or directly in the Node list section of the configuration file. Each package can belong to any number of categories. Packages with no category are not to be installed (only kept in repository). Packages will only be installed on nodes with corresponding categories or supercategories. No user changes should normally be necessary.

Here is an example package list:

```
# Format: package <name> <categories>
package ScaBoot      base_n
package ScaConfC     comm_sci_f
package ScaEnv       base_f base_n devel user
```

E-2.3 ScaPkg.conf - dependency list

The dependency list section defines dependencies between packages. Each package may have several dependencies. A line in the dependency list is on format:

```
dependency <pkgname> <deppkg>
```

where `<pkgname>` is the package module and `<deppkg>` is a list of all packages that this package depend upon. Note that no user changes should be necessary in this section. Here is an example package dependency list entry, defining dependencies for the package `ScaConfC`:

```
# Format: dependency <pkgname> <deppkg>
dependency ScaConfC      ScaConfSd ScaConfNd ScaSH
```

Appendix E: ScaPkg - Scali Software installation program

E-2.4 ScaPkg.conf - Super-categories

The supercategories section contains mappings where several categories are mapped into single supercategories. This extra level of abstraction has been introduced to make administration of the node configuration easier, both for direct editing and for modifying the configuration file during installation of the SSP.

The format for a supercategory entry is:

```
supercategory <supercategory> <categories> ...
```

where <supercategory> is the name of supercategory, and <categories> is a list of all categories mapping to this supercategory.

Here is an example of super-category mappings:

```
supercategory workstation user mon_n
supercategory sci_node base_n mon_n comm_sci_n queue_n
supercategory smp_node base_n mon_n comm_smp_n
supercategory eth_node base_n mon_n comm_eth_n queue_n
supercategory sci_frontend base_f mon_f comm_sci_f devel_sci
supercategory smp_frontend base_f mon_f comm_smp_f queue_f
supercategory eth_frontend base_f mon_f comm_eth_f queue_f
```

Here the supercategory `workstation` will contain the categories `user` and `mon_n`. When an entry in the `nodelist` section refers to the supercategory `workstation`, it will include packages belonging to the categories `user` and `mon_n`.

E-2.5 ScaPkg.conf - Node list

The `nodelist` section defines which nodes to install packages on and which categories or supercategories they belong to. Each host may belong to any number of categories.

The format for a node list entry is

```
node <name> <categories>
```

where <name> is the name of the node and <categories> is a list of all categories or supercategories this node is part of.

Here is an example `nodelist` defining four nodes `node-1`, `node-2`, `node-3`, `node-4` and a frontend machine, `boss`:

```
# Format: node <name> <categories> ...
node node-1 sci_node eth_node
node node-2 sci_node eth_node
node node-3 sci_node eth_node
node node-4 sci_node eth_node
```



```
node boss    sci_frontend eth_frontend smp_node
```

E-2.6 Package response files

If a file with the name <package>.response is found in the package repository (without architecture extension), this file will be read when installing <package>. Example is `scasCI.response`. No user changes should be necessary for these files.

Appendix E: ScaPkg - Scali Software installation program

Appendix F Related Documentation

F-1 References

- [1] **“MPI: A Message-Passing Interface Standard”**, The Message-Passing Interface Forum, Version 1.1, June 12, 1995, <http://www.mpi-forum.org>.
- [2] **“MPI: The complete Reference: Volume 1, The MPI Core”**, Marc Snir, Steve W. Otto, Steven Huss-Lederman, David W. Walker, Jack Dongarra. 2e, 1998. The MIT Press.
- [3] **“MPI: The complete Reference: Volume 2, The MPI Extension”**, William Grop, Steven Huss-Lederman, Ewing Lusk, Bill Nitzberg, William Saphir, Marc Snir. 1998. The MIT Press.
- [4] **“Scali Library Guide”**, Scali AS, <http://www.scali.com>
- [5] **“ScaMPI release notes”**. Scali AS, <http://www.scali.com>.
- [6] **“The Scali parallel tools environment”**, Draft 1999, Scali AS, <http://www.scali.com>.
- [7] **“CCS: Computing Center Software resource management for networked high-performance computers”**. Paderborn Center of Parallel Computing, <http://www.uni-paderborn.de/pc2>
- [8] **“VAMPIRtrace for Solarisx86/ScaMPI Installation and User’s Guide”**, Pallas GmbH, Release 1.0 for VAMPIRtrace version 1.5, 1998, <http://www.pallas.de>.
- [9] **“Review of Performance Analysis Tools for MPI Parallel Programs”**, <http://www.cs.utk.edu/~browne/perftools-review/>.
- [10] High Performance Debugger Forum, <http://www.ptools.org/hpdf/>.
- [11] NHSE, National HPCC Software Exchange - Parallel Tools Library, <http://www.nhse.org/ptlib>.
- [12] TFCC, IEEE CS Task Force on Cluster Computing, <http://www.dgs.monash.edu.au/~rajkumar/tfcc>.
- [13] The Extreme Linux Organisation, <http://www.extremelinux.org>.

Appendix F: Related Documentation

- [14] **“IEEE Standard for Scalable Coherent Interface(SCI)”**, IEEE Std 1596-1992. Additional information on SCI is available from: <http://www.scizzl.com/>
- [15] **“UCD-SNMP Home page”**, <http://ucd-snmp.ucdavis.edu/>

List of tables

1-1	Acronyms and abbreviations.....	12
1-2	Basic terms	13
1-3	Typographic conventions.....	13
3-1	Topology vs. SCI hardware compatibility	26
3-2	Connection list example for cabling an 8 node SCI ring	27
3-3	Example cabling of one vertical ring in a 4x4 system (x=[1-4])	28
3-4	Example cabling of one horizontal ring in a 4x4 system (y=[1-4]).....	29
3-5	Example cabling of L0 on a 3x3x2 torus (z=[1,2], x=[1,2,3])	30
3-6	Example cabling of L1 on a 3x3x2 torus (z=[1,2], y=[1,2,3])	31
3-7	Example cabling of L2 on a 3x3x2 torus (x=[1,2,3], y=[1,2,3])	31
5-1	System MainWindow node symbols	68
5-2	System MainWindow menu overview	68
5-3	Main window: Run menu items explained.....	70
5-4	Management menu options.....	75
5-5	Software menu options.....	75
6-1	The Status Menu	79
6-2	Monitoring view window options menu.....	80
6-3	Alarm Main window control summary.....	85
6-4	Alarm Editor control overview.....	86
6-5	Default "Performance" monitoring variables	89
6-6	Default "System" monitoring variables.....	90
7-1	The Interconnect menu items	97
7-2	SCI link status window pop-up menu options	99
7-3	ScaConf node states.....	103
9-1	Options in the file ScaSH.conf	137
10-1	Software installation troubleshooting.....	140
10-2	LED behaviour on D33x cards.....	141
10-3	Some SCI error types	142
10-4	ScaSCI troubleshooting.....	143
10-5	Overview of Scali software packages distributed with the SSP.	153
10-6	Scali daemon overview	155
10-7	The configuration files in /opt/scali/etc	156
10-8	ScaConf packages	171
10-9	Available server options. Default values are marked with *	173

List of figures

2-1	System architecture of a 4x4 Scali system.....	16
2-2	Scali system cabinet examples: Standard half-size Terarack (left) and a ruggedized version for portable supercomputing (right).....	17
2-3	OEM partner packaging: Fujitsu- Siemens hpcLine (left) and Dell (right) ...	17
2-4	A 4x8 Scali system boxed in Scali cabinets.....	18
2-5	Open front and rear side of a Scali system in a Scali cabinet.....	18
3-1	A 4x4 cluster with XY coordinates: physical view.....	19
3-2	Using private network for the Scali System.....	21
3-3	Scali system with channel bonding.....	22
3-4	SCI hardware from Dolphin: PCI adapter, daughter card, standard cable (left) and flexi cable (right).....	23
3-5	Identifying L0 and L1 connector pairs on SCI cards (D335).....	24
3-6	D31x SCI adapter card jumpers.....	24
3-7	Node with SCI adapter card and daughter card inserted.....	25
3-8	Avoid long cables by interleaving interconnect in each ring.....	27
3-9	Example of 1x8 single ring cabling in a 1x8 cabinet.....	28
3-10	Example of 2x4 two-dimensional system cabling in a 2x4 cabinet.....	29
3-11	Example of 2x4 two-dimensional system cabling in a 1x8 cabinet.....	29
3-12	A 3x3x2 3D SCI torus.....	30
3-13	Example cabling of a 3x3x2 3D torus, please note that SCI links have been drawn mirrored for clarity.	31
5-1	Scali Universe overview.....	61
5-2	Initial unconfigured Scali System Management window.....	63
5-3	An example Scali Universe session with multiple monitoring views.....	64
5-4	Configuration window showing system defaults.	65
5-5	Example System configuring.....	65
5-6	Scali System Management window with five Scali Systems.....	66
5-7	Login dialogue box.....	66
5-8	Scali Universe system main window with four selected nodes.....	67
5-9	Main Window views: "Detail View"(left) and "Small Icons"(right).....	69
5-10	Scali MPI monitor window.....	71
5-11	Scali MPI monitor window with output from an MPI program.....	72
5-12	Example of a Parallel shell command.....	73
5-13	The Master window for terminal broadcasting.....	74
5-14	The Management menu.....	74
5-15	Software installation window.....	76
6-1	ScaMon components in context.....	77
6-2	The Status Menu.....	78

6-3	3D view camera controls.....	80
6-4	Compact view examples with 4and 132 nodes.	81
6-5	2D Bar small-icons view of system load	81
6-6	3D bar view on a large system (132 nodes)	82
6-7	2D history view showing two of four nodes	82
6-8	3D History view	83
6-9	3D System view on a 2D SCI system.....	83
6-10	The Alarm window.....	84
6-11	The alarm editor window.....	85
6-12	The alarm log viewer	87
6-13	Adding a new alarm.....	87
7-1	ScaConf components in context.....	95
7-2	Interconnect menu added to the Main Window	96
7-3	High speed SCI network link status window.	97
7-4	Link status graphics explained.....	98
8-1	Example of submitting a ScaMPI job from Scali Universe to 4 nodes with 2 processes at each node.	124
8-2	Submit window with Reserve nodes only selected.....	125
8-3	Example of submitting a MPICH job from Scali Universe to 4 nodes, one process pr. node.....	125
8-4	Options, Advanced... window for Job Submission.....	126
8-5	Example Queue Status display in Scali Universe.....	127
8-6	Example Queue detail window in Scali Universe	128
10-1	Vertical, horisontal and rectangle groupes of nodes running the same test application job in a 4x4 2D torus. The wraparound connections of the tours are not shown.	147

Index

Symbols

2D bar view	81
2D history view	82
3D bar view	82
3D History View.....	83
3D System View	83

A

Alarm Editor Window.....	85
Alarm log viewer.....	87
Alarm Window	84
Alarms	84
activating.....	88
defining.....	87

B

bidirect.....	148
---------------	-----

C

camera controls	80
CCS resource management software.....	187
channel bonding.....	22
Compact View	81
compound view.....	83
configuration files	
overview	156
ScaConf.conf.....	107, 110
ScaDesk.conf	63
console switching	
cabling	32
configuration.....	49
OS preparation.....	37
customer ID tag.....	157

D

D312.....	146
D335.....	24

D336	26
D337	24, 26
daemons	
overview	155
scaconfnd	111
scaconfsd	110
scamond	88
scasnmpd	90
E	
environment variable	
SCALM_LICENSE_FILE	160
Ethernet setup	21
F	
Frontend Terminal	74
H	
Hardware installation problems	139
I	
Interconnect Menu	97
Interrupt conflict	143
L	
LED	141
License	140
license	
demo	157
file	159
permanent	158
requesting	159
link errors	144
M	
mailing lists	150
Management Menu	74
Management Window	63
Master Terminal Window	74
monitoring	
overview	77
variables (default)	89

MPI	141
mpimon	141
N	
NIS	38
node selection	67, 101
Node Terminal	73
O	
OpenGL	39
P	
Package response files	185
Pallas benchmark	148
Parallel Shell	73
power switching	
configuration	49
private network	22
R	
References	187
reroute	144
Reset Counters	98
Ring topology	107
rlogin	36
routing	107
C3	109
dimensional	108
maxcy	108
partitions	109
rsh	36
Run Menu	70
run_permutated_bidirect	148
runsets	70
S	
scaconfnd	111
scaconfsd	110, 140, 146
setting server options	106
scaconftool	140, 144
ScaConfTool commands	171
scacp	133

scahosts.....	132
scakill.....	135
Scali software platform.....	131
Scali system	
guide.....	11
overview.....	15
Scali Universe	
getting started.....	62
overview.....	61
running.....	63
using.....	67
scamond.....	88
ScaMPI.....	141
ScaMPI FAQ.....	141
scapkg.....	140
scaps.....	134
scarep.....	133
scarup.....	135
ScaSCI error messages.....	141
ScaSCI trouble shooting.....	143
ScaSCIadap.....	140
ScaSH.....	131
scash.....	131
scasnmpd.....	90
scasub.....	122
SCI	
flexi cable.....	23
hardware overview.....	23
reloading driver.....	105
setting nodeID.....	104
standard cable.....	23
SCI cabling.....	26
2D torus.....	28
3D torus.....	30
ring.....	27
sci_hwdiag.....	145, 147, 148
sci_hwdiag.....	144
sci_hwid.....	144
sci_hwtop.....	144, 145
scicards.....	169
scidbx.....	166
scidle.....	169

sciemsg	166
scimonitor	166
scinode	167
sciping.....	144, 165
scireconf.....	146
Software installation problems	139
Software Menu	75
SSP.....	12
CD-ROM.....	151
directory structure.....	152
installation program.....	35
uninstall program	59
SSPinstall.....	140
Status Menu	78
support.....	149
System Configuration Window.....	65
T	
Task Force on Cluster Computing	187
terminal broadcasting.....	74
topology.....	145
torus	145
Torus Topology	107
troubleshooting	139
U	
user profiles.....	63
V	
VAMPIRtrace Installation and User's Guide.....	187
View Menu.....	69
W	
WulfkitTM.....	150

