SSI-OSCAR: a Cluster Distribution for High Performance Computing Using a Single System Image

Geoffroy Vallée, Stephen Scott, Christine Morin, Jean-Yves Berthou, Hugues Prisker

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

SSI-OSCAR: a Cluster Distribution for High Performance Computing Using a Single System Image

Geoffroy Vallée [1 2] , Stephen L. Scott [3] , Christine Morin [4] , Jean-Yves Berthou [5] ,

Hugues Prisker [5]

N°5538

Mars 2005

Systèmes communicants

ISSN 0249-6399 ISRN INRIA/RR--5538--FR+ENG

SSI-OSCAR: a Cluster Distribution for High Performance Computing Using a Single System Image

Geoffroy Vallée [1 2] , Stephen L. Scott [3] , Christine Morin [4] , Jean-Yves Berthou [5] , Hugues Prisker [5]

Systèmes communicants
Projet PARIS

Rapport de recherche n˚5538 — Mars 2005 — 13 pages

Abstract: The ease use and management of clusters needs tools to install, update and transparently manage distributed clusters resources. The management of clusters (node installations and updates) is a well-known problem and some high performance computing specific distributions are available today. These distributions, like OSCAR, allow users to install and manage clusters without specialized knowledge, allowing quick cluster deployment. The ease use of cluster is possible globally and transparently managing cluster resources. Single System Image (SSI) has this approach and some projects provide a SSI for Linux clusters.

But today, there is no solution combining these two advantages: integration of a SSI and a distribution for high performance computing on Linux clusters. This paper presents SSI-OSCAR which aims at providing such a solution, combining OSCAR and the Kerrighed SSI.

Key-words: cluster management, distributed system, operating system, single system image, OSCAR, Kerrighed

(Résumé : tsvp)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Process fantôme: mécanisme de base pour la mise en œuvre de mécanismes de duplication, migration et de création/reprise de points de reprise de processus

Résumé : Pour administrer et utiliser simplement les grappes de calculateurs, des outils d'installation, d'administration et de gestion transparentes des ressources distribuées sont nécessaires. L'administration des grappes de calculateurs (installation et d'administration de nœuds de la grappe) est un problème connu et des distributions pour le calcul haute performance sont aujourd'hui disponibles. Ces distributions, comme OSCAR, permettent aux utilisateurs d'installer et d'administrer les grappes de calculateurs sans compétences particulières, permettant ainsi un déploiement rapide de grappes de calculateurs. Cette simplicité d'utilisation est possible est gérant les ressources des grappes de calculateurs de manière globale et transparente. Les systèmes à image unique (Single System Image - SSI) ont cette approche et plusieurs projets fournissent actuellement des SSI pour grappes de calculateurs utilisant Linux.

Mais aujourd'hui, il n'existe pas de solution regroupant à la fois ces deux avantages : l'intégration d'un SSI et d'une distribution pour le calcul haute performance pour grappe de calculateurs utilisant Linux. Ce document présente SSI-OSCAR qui vise à fournir une telle solution, associant OSCAR et le système à image unique Kerrighed

Mots-clé : administration de grappes, systèmes distribués, système d'exploitation, système à image unique, OSCAR, Kerrighed

1 Introduction

The distributed architecture of clusters implies two major issues: clusters are difficult to manage (cluster installation and update) and to use (application programmers and users have to manage cluster resources themselves). Moreover, a large part of clusters run Linux which was become the operating system of choice for high performance computing on clusters.

Today, some software suites for clusters allow to easily manage clusters, provide tools to install and update Linux clusters, *e.g.*, OSCAR [10] and Rocks [12]. These tools provide all well-known software packages needed to use clusters for high performance computing (*e.g.* programing framework like MPI [7, 2], PVM [14], or batch systems like OpenPBS [4]) through a unified interface. Finally, clusters managers can easily and quickly install a new cluster with software that users need. But these current tools do not provide completely the illusion of a unique single machine, *i.e.*, do not give the illusion of a single SMP machine. For that, all resources have to be globally managed; this is the approach of Single System Image (SSI) for clusters.

Combining characteristics of tools like OSCAR and characteristics of SSI, a complete solution can be created to easily manage and use Linux clusters. The SSI-OSCAR project aims at providing such a solution, integrating the Kerrighed SSI into OSCAR.

2 OSCAR Overview

OSCAR is a snapshot of the best known methods for building, programming, and using Beowulf clusters [13]. It consists of a fully integrated and easy to install software bundle designed for high performance cluster computing. Everything needed to install, build, maintain, and use a modest sized Linux cluster is included in the suite. Using such a bundle, it is unnecessary to download or even install any individual software packages on your cluster.

2.1 OSCAR Architecture

An OSCAR cluster is composed of a head node which is the file server (using NFS), machine for user connection to the cluster and may be a Domain Name Server (see Figure 1). Other nodes are compute nodes.

For that, OSCAR merges different Linux packages in one *packages set*. To be able to manage these packages at the cluster scale, a new level of package has been created: *OSCAR packages*. OSCAR packages and tools allow for management of software components at the cluster level. Typically, a package's state on nodes (*e.g.* installation state, versioning) is managed through the OSCAR databas.

[1]ORNL/INRIA/EDF - Computer Science and Mathematics Division - Oak Ridge National Laboratory - Oak Ridge, TN 37831, USA
[2]INRIA Postdoc co-funded by EDF R&D and ORNL
[3]ORNL - Computer Science and Mathematics Division - Oak Ridge National Laboratory - Oak Ridge, TN 37831, USA
[4]IRISA/INRIA, PARIS project-team - Campus Universitaire de Beaulieu, 35042 Rennes, Cedex, France
[5]EDF R&D - 1 avenue de Général de Gaulle, BP408, 92141 Clamart, France

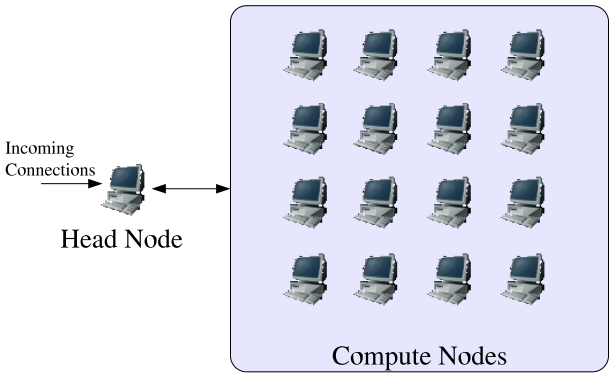

Figure 1: OSCAR Cluster Architecture

An OSCAR package is composed of an XML file, some configuration scripts and binary packages (see Figure 2). The XML file specifies information about the package version and dependences

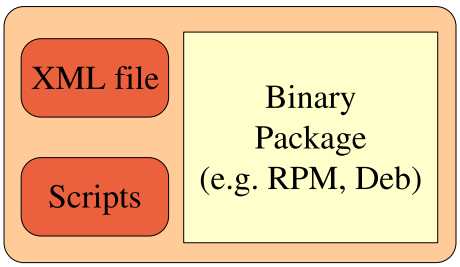

Figure 2: Architecture of OSCAR Packages

with other OSCAR packages. Scripts are used to install, update and remove OSCAR packages at the cluster scale (*versus* scripts included in binary files which install, update and remove packages at the local scale). When an OSCAR packages is installed, updated or removed on a cluster, the OSCAR database is used to store and manage this package configuration information. This database records the current state of installed packages, including dependencies state between OSCAR packages. Finally, OSCAR manages packages at the cluster scale, extending tools used by Linux distribution for local package management. This approach also allows one to create different *packages sets* and to be able to install different cluster configuration using a single interface.

Two kind of packages compose OSCAR: (i) *OSCAR application packages* and (ii) *OSCAR tools packages*. *Application packages* provides well-known applications, libraries to program and use the clusters, *e.g.*, MPI[15], PVM[14], and Ganglia. *OSCAR tools packages* provide some internal facilities which are used to configure the cluster. For example, the package *kernel_picker* allows a user to select the kernel installed on the compute nodes.

2.2 OSCAR Cluster Installation

An OSCAR cluster is installed in seven steps (see Figure 3): (i) package selection, (ii) packages configuration, (iii) head node installation (iv) creation of the image for compute node, (v) configuration of the node nodes, (vi) network configuration, (vii) cluster configuration. This section does not present details of each step which is out of scope of this paper but gives an overview of the cluster installation.

First of all, the head node has to be installed: the Linux distribution and the OSCAR software have to be installed. To be able to install head nodes and compute nodes, OSCAR uses a local repository of all packages included in the Linux distribution used, using a local */tftpboot/rpm* folder.

When the head node is completely setup, OSCAR can be launched to install computes nodes. During the OSCAR initialization, some scripts install and setup basic packages. These packages are composed of OSCAR tools to manage a complete cluster, *e.g.*, the OSCAR database, Packman/Depman.

After the initialization, to install and setup a cluster, OSCAR first setup the head node for OSCAR needs (packages installation and setup), and then an image is created for compute nodes. With this image, the cluster administrator can define the set of compute nodes which have to be based on this image. Then, a network boot of the compute nodes install automatically and transparently installs and configures all compute nodes.

Finally, the cluster is setup executing post-install scripts of already selected packages.

3 Single System Image Overview

A Single System Image (SSI) globally and transparently manages cluster resources. With such a system, users and programmers have a single view of the distributed resources just as if the system was a SMP machine. SSI aims at providing a support for both parallel and sequential programming paradigms, *e.g.* MPI, OpenMP). For that, SSI systems provide basic mechanisms like process migration, process checkpoint/restart, software distributed shared memory, and global data streams management (mechanisms for migration and checkpoint/restart of sockets, pipes and signals). Finally, users can execute applications, taking advantage of distributed resources, without application modification. SSI also aims at managing node failures with high availability mechanisms. The goal is to be able to tolerate node failures, guaranteeing application and system execution.

Today, several project aims at providing a SSI for Linux clusters. The three major projects for Linux SSI are: openMosix citetortone-openmosix, OpenSSI [11] and Kerrighed [9, 8, 16]. All of these systems are composed of kernel patches, some kernel modules and some user space tools and libraries. Each of these projects take a different approach for creating a SSI. Because of these different approaches, performance for each system differs depending on the application [6].

OpenSSI is the integration of existing projects to provide a complete solution for global resource management. OpenSSI allows a user to globally manage processes, disks and data streams. However the global memory management is not supported.

OpenMosix is a SSI focused on high performance through a global scheduling mechanism to guarantee efficiency for application execution. OpenMosix does not provide a global memory man-

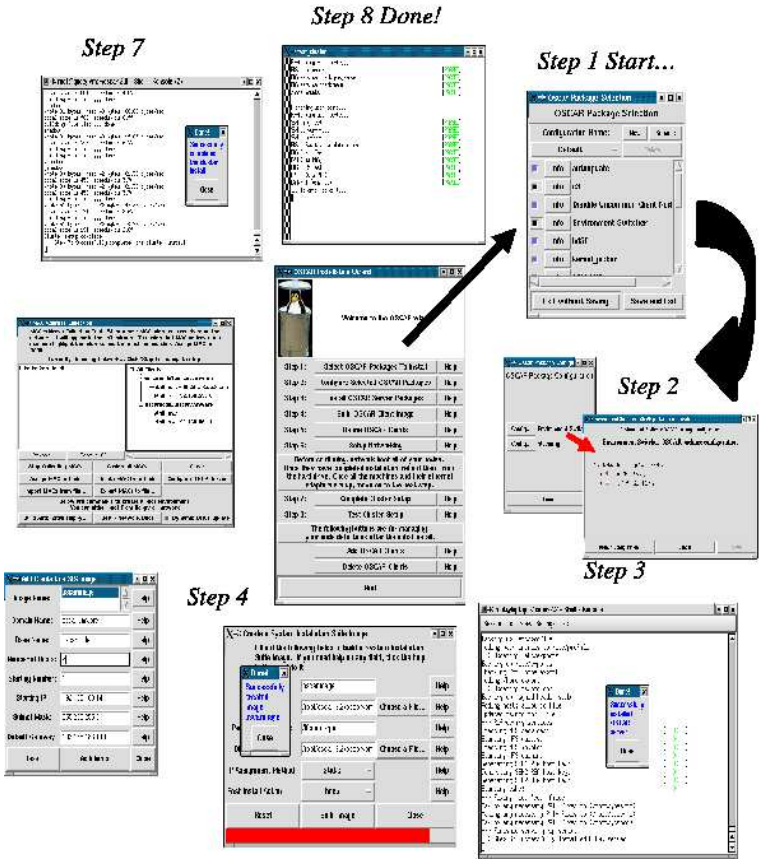

Figure 3: Cluster Installation Using OSCAR

agement, nor an efficient migration of data streams (like sockets), because of a dependency with the original node when a migration occurs. High availability issues are out the scope of the openMosix project.

Kerrighed is a SSI, developed from scratch that extends the Linux kernel, which globally manage all resources: disks, memories, data streams, and processors. An advantage of Kerrighed is to be able to setup the system according to an application's needs.

4 SSI-OSCAR Architecture

OSCAR allows a user to install a Beowulf cluster, which has a head node and compute nodes. The head node provides centralized services which can be used by compute nodes. In a SSI, all services are distributed to give the illusion of a single machine, no head node distinction is necessary. But the current OSCAR version is not modular and not configurable enough to activate only needed services. Nevertheless, OSCAR's cluster installation method is based on the traditional way of building a cluster, first build a head node, then an image for compute nodes and finally install compute nodes. To avoid significant OSCAR modifications, a simple solution is to still use a Beowulf architecture where the head node is used to manage (launch and stop the SSI) and monitor the SSI. With this approach, the method to build a SSI cluster is exactly the same as that used for a Beowulf cluster; SSI-OSCAR installation and OSCAR installation are identical. Furthermore, to avoid major modifications to OSCAR, distributed services like distributed file system (NFS) are configured by OSCAR and are still used with the SSI cluster, even if the SSI provide a similar service.

Being based on OSCAR, SSI-OSCAR developments have to be directly integrated into OSCAR. For that, kernels have to be managed by the OSCAR's kernel tool (*kernel_picker*) and OSCAR modifications have to be limited. Therefore, packaged software must use the tools and libraries provided by OSCAR.

The packaging is composed of two steps: first the creation of packages for the supported Linux distribution (*e.g.* RPM) and then the creation of OSCAR packages. The creation of packages for the Linux distribution can be difficult if the SSI is not initially developed on this distribution. The port on a new distribution can be complex because each distribution provides a specific version of tools needed to create a new kernel and associated modules. Depending of the compiler version, of the kernel version supported by the distribution, some critical issues can appear, if the kernel used by the SSI is not close enough to the distribution kernel. In this case, the SSI kernel has to be ported to the new Linux distribution, which can be complex and time consuming. After that, tools and libraries also have to be ported which is similar in time complexity as creating a package.

When Linux packages have been created, they can be rolled into OSCAR packages. The creation of OSCAR packages for SSI is no different than the creation of standard OSCAR packages. The current version of *kernel_picker* does not enable the automatic inclusion of new kernels. The OSCAR package *kernel_picker* must be modified in order to include the SSI kernel. So for each new kernel, the *kernel_picker* package has to be modified.

5 Implementation

SSI-OSCAR 1.0 is based on OSCAR-3.0 for RedHat 9 and includes Kerrighed-1.0rc8. The steps for the creation of SSI-OSCAR include: (i) the creation of Linux packages for the supported Linux distribution (*i.e.* RedHat 9), (ii) the creation of OSCAR packages that involve these Linux packages and (iii) the integration of the new OSCAR packages into OSCAR.

5.1 Creation of Linux Packages

The code of Kerrighed 1.0rc8 is organized in two parts: kernel code and two user level components providing tools and libraries (see Figure 4). The kernel code comprise of modules code and the kernel patch to create a Kerrighed kernel. Libraries are composed of the POSIX thread [5] interface and some interfaces for Kerrighed tools. Kerrighed tools allow users to test and monitor the system. Initially, Kerrighed was not developed to be included directly into a Linux distribution. All tools

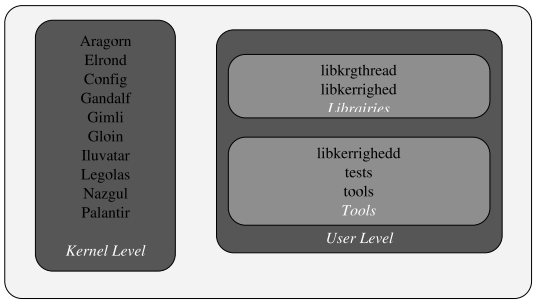

Figure 4: Kerrighed's Code Organization

and scripts are dependent on the Kerrighed's sources location. Therefore, to integrate Kerrighed in OSCAR, we had to create packages providing all binaries and scripts needed to execute the system. Therefore binaries have to be directly installed into the Linux distribution and scripts have to be modified for that. Currently, five Kerrighed packages have been created: *module*, *lib*, *include*, *tools* and *linux-krg*. Moreover, two additional packages have been created to provide all sources needed to compile a complete Kerrighed from scratch: *kerrighed-source* and *linux-krg-source* (kernel sources with Kerrighed's patch applied).

5.2 Integration in OSCAR

When Linux packages are ready, they can be integrated into OSCAR. This integration is composed of two steps: (i) the creation of OSCAR packages and (ii) the integration into OSCAR which may need modifications of other OSCAR packages.

5.2.1 Creation of OSCAR packages

The Kerrighed integration into OSCAR just needs to create one OSCAR package which merges all the Linux packages. For that, OSCAR rules on package creation are followed: an XML file to describe the OSCAR package and scripts to configure the OSCAR package.

For Kerrighed, scripts are pretty simple: the installation is managed by the *kernel_picker* tools, therefore only configuration, update and removal scripts have to be created. However the Kerrighed version packaged in SSI-OSCAR does not support update or removal. Therefore only the configuration script is needed. This script generates all the configuration files needed by Kerrighed. Kerrighed 1.0rc8 needs two configuration files: (i) */etc/kerrighed/kerrighed_nodes* (to define nodes included in the Kerrighed cluster, and the network interface on each node to use for Kerrighed communications - the current version is limited to use of the first ethernet interface on each node) and (ii) */etc/kerrighed/kerrighed_session* (to specify the session identifier for the Kerrighed cluster). This information enables the creation of cluster partitions. These two files are created using OSCAR information about the cluster configuration.

5.2.2 OSCAR Modifications

The Kerrighed integration into OSCAR is simple because the Kerrighed system does not modify kernel interfaces. Therefore it is primarily a replacement of the stock RedHat 9 kernel with a Kerrighed kernel package.

So, Kerrighed packages just have to be included in the OSCAR distribution and no heavy OSCAR modifications are needed. To add a new OSCAR package, a developer just needs to create a directory tree according to the OSCAR packaging rules [1]. When the directory tree is created, scripts we added to initialize, update and remove the OSCAR package, and binary packages are integrated.

The only real modification needed by the Kerrighed integration is to modify the *kernel_picker* tool to be able to manage the Kerrighed kernel. *kernel_picker* can manage two kinds of kernels: (i) kernels directly included in the supported Linux distribution which are directly managed by *kernel_picker* and (ii) kernels specified by users. To transparently install the Kerrighed kernel on compute nodes, without any configration from users, *kernel_picker* has to be modified. This modification is simply the addition of few lines of code in two *kernel_picker* scripts: (i) *post_clients* and (ii) *pre_configuration*.

5.3 SSI-OSCAR Installation

The SSI-OSCAR installation and typical OSCAR installation are quite similar: (i) OSCAR packages are selected and by default the OSCAR package for Kerrighed is selected, (ii) OSCAR packages are configured and by default *kernel_picker* is set to use the Kerrighed kernel (iii) the head node is setup (*e.g.* package installations), (iv) an image for compute nodes is created and the modifications to *kernel_picker* include the Kerrighed kernel by default, (v) compute nodes are defined, (vi) the network is configured, compute nodes connect to the head node for a network install, (vii) final cluster setups are done.

Ultimately, SSI-OSCAR installation and OSCAR installation differ in the kernel included into the compute nodes image and in the installation of the OSCAR packages for Kerrighed. These two points are transparent for users.

5.4 Conclusion

With the OSCAR package for Kerrighed, Kerrighed can be installed on all the compute nodes and the Kerrighed cluster is completely configured and ready to be launched. Then, the cluster manager simply has to log on the head node and launch the Kerrighed command *krgreboot*, which loads kernel modules on all compute nodes. After that, users can take benefit of Kerrighed services by just logging on to a compute node and launching their applications.

6 Experimentations

For SSI-OSCAR experiments, we used SSI-OSCAR 2.0 release candidate 1 which supports OSCAR 4.0, Kerrighed 1.0rc8 for the Linux RedHat 9.0 distribution. The only difference between SSI-OSCAR 1.0 and SSI-OSCAR 2.0 is that SSI-OSCAR 2.0 uses the more recent OSCAR 4.0 release.

The Kerrighed 1.0rc8 version is based on the Linux 2.4.24 kernel and provides mechanisms for global process management (basic mechanisms like process migration, checkpoint/restart of sequential applications, and load balancing), software distributed shared memory (shared memory applications can be executed on top of Kerrighed, deploying threads on different nodes) and global data stream management (processes communicating by sockets can be migrated).

6.1 Experimental Environment

We built a cluster comprised of Pentium IV 1.7GHz head node with 1GB RAM, 40GB HD space and two 100MB Ethernet network interface cards; and four dual Pentium III 450MHz processor compute nodes with 512 MB RAM, 20GB HD space and one 100MB Ethernet network interface card. Network interface cards do not allow any network boot (using PXE), therefore, a floppy disk is used in order to connect the server and then install the system through the network.

In these experiments, we first evaluate the size difference for compute nodes on one hand with OSCAR 4.0 and on another hand with SSI-OSCAR 2.0 release candidate 1. Then we evaluate the installation time both with OSCAR and SSI-OSCAR.

6.2 Image Size of Compute Nodes

An image is created using SSI-OSCAR to install compute nodes with a full SSI, *i.e.*, all libraries and tools of the traditional OSCAR distribution as well as the Kerrighed operation system. On another hand, we create a "standard" image for compute nodes to install an OSCAR cluster.

The size of the SSI image for compute nodes is 678MB; the size of the OSCAR image is 615MB. The size difference between the two images (63MB) is due to the additional packages needed spe-

cially for SSI-OSCAR: all Kerrighed packages (modules, libraries, tools, includes and sources) and supporting packages with a dependence used by Kerrighed.

6.3 Installation Time Analysis

Figure 5 shows the installation times for 1 to 4 nodes between SSI-OSCAR and OSCAR. The installation time is the time between the beginning of the installation (the floppy boot to install a node) and the complete installation (the time just before the reboot of the machines upon completion of the installation). The Figure 5 shows that the overhead incurred by the Kerrighed integration is

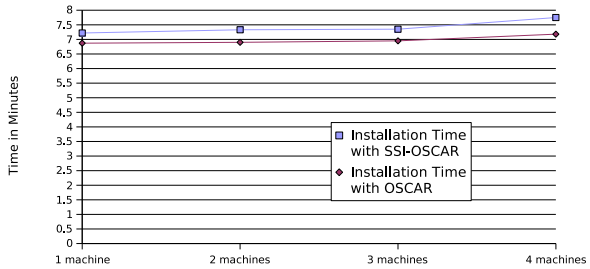

Figure 5: Cluster Installation Times for OSCAR and SSI-OSCAR

limited, less than 10%. This overhead can be decreased by avoiding the installation of redundant mechanisms'software packages like the batch system (PBS) or the distributed file system (NFS).

SSI-OSCAR has to be evaluated on a larger cluster. Nevertheless, OSCAR modifications for the SSI-OSCAR implementation are not significant therefore SSI-OSCAR should have performances quite similar of the OSCAR distribution. Compared to OSCAR, SSI-OSCAR only scales up to 32 nodes because of current technical limitations of the Kerrighed operating system (this scalability limitation will be removed in a next Kerrighed version which will scale up to a few hundred nodes).

7 Conclusion

Combining a SSI and OSCAR, SSI-OSCAR is the first solution for users to easily install, configure and use Linux clusters. The integration of a SSI allows OSCAR to provide another solution for clustering. This solution provides: (i) tools, to simply and quickly install and configure the cluster, (ii) a complete set of software for high performance computing (like MPI or PVM) and (iii) a SSI to guarantee a efficient and transparent use of resources.

The current stable version (SSI-OSCAR 1.0) is based on OSCAR 3.0 for RedHat 9 and Kerrighed 1.0rc8. The SSI-OSCAR 2.0 version is in development and is based on OSCAR 4.0 and Kerrighed 1.0 for both the RedHat 9 and Fedora Core 2 Linux distributions.

New work is currently made to support new processors (*e.g.* Opteron machines) and new Linux distribution, *e.g.*, porting OSCAR and Kerrighed to Debian in collaboration with Electricity of

France (EDF) Research and Development. A direct integration of the SSI enhancements into OSCAR is also being, but modifications are needed to the OSCAR architecture and SSI-OSCAR must be port to new Linux distributions and processors (*e.g.* RHEL 3 and IA64 machines).

Finally, some OSCAR "spin-offs" like HA-OSCAR [3] may benefit from a SSI. HA-OSCAR aims at providing the combined power of high availability and high performance computing for Linux clusters. For that, HA-OSCAR is based on checkpoint/restart mechanisms to tolerate failures. Kerrighed provides such mechanisms which can be used with other services of global resource management. Finally, the merge of SSI-OSCAR and HA-OSCAR may offers an interesting solution for high availability Linux clusters, taking benefit of policies developed in HA-OSCAR and efficient mechanisms of SSI-OSCAR.

The direct integration of such "spin-off" projects into OSCAR is also very interesting to provide a global solution for clustering. With such solution, just creating some *package sets* of OSCAR packages, a cluster administrator will be able to install and easily manage a cluster better adapted to the users needs. The current OSCAR implementation does not allow packages authors to specify dependencies and constaints between multiple packages, in order to create a coherent set of packages. For example, it is not possible to specify that the Kerrighed package depends to the *kernel_picker* package and to specify a configuration file for *kernel_picker* (the file sets the default kernel used in the image for the compute node). It is also impossible to specify that the Kerrighed does not need a batch system like OpenPBS or Torque. OSCAR provides such packages but SSI has mechanisms to manage jobs and perform global resource management. Therefore using both a batch system and a SSI, scheduling decisions may not work together.

References

[1] Core OSCAR Team. HOWTO: Create an OSCAR package, January 2004. http://oscar.openclustergroup.org/tiki-index.php.

[2] Al Geist, William Gropp, Steve Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, William Saphir, Tony Skjellum, and Marc Snir. MPI-2: Extending the Message-Passing Interface. In Luc Bouge, Pierre Fraigniaud, Anne Mignotte, and Yves Robert, editors, *Euro-Par '96 Parallel Processing*, number 1123 in Lecture Notes in Computer Science, pages 128–135. Springer Verlag, 1996.

[3] Ibrahim Haddad, Chokchai Leangsuksun, and Stephen L. Scott. Ha-oscar: the birth of highly available oscar. *Linux J.*, 2003(115):1, 2003.

[4] Henderson and L. Robert. Job scheduling under the portable batch system. In Dror G. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, pages 279–294. Springer-Verlag, 1995. Lecture Notes in Computer Science vol. 949.

[5] IEEE. *IEEE P1003.4a/D4 draft standard, Threads Extension for Portable Operating Systems*, AUG 1990.

[6] Renaud Lottiaux, Benoit Boissinot, Pascal Gallard, Geoffroy Vallée, and Christine Morin. Openmosix, openssi and kerrighed: A comparative study. Technical Report RR-5399, Institut National de Recherche en Informatique et en Automatisme (INRIA), NOV 2004. ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-5399.pdf.

[7] Message Passing Interface Forum. MPI: A Message Passing Interface. In *Proc. of Supercomputing '93*, pages 878–883. IEEE Computer Society Press, November 1993.

[8] Christine Morin, Pascal Gallard, Renaud Lottiaux, and Geoffroy Vallée. Towards an efficient Single System Image cluster operating system. *Future Generation Computer Systems*, 20(2), January 2004.

[9] Christine Morin, Renaud Lottiaux, Geoffroy Vallée, Pascal Gallard, Gaël Utard, Ramamurthy Badrinath, and Louis Rilling. Kerrighed: a single system image cluster operating system for high performance computing. In *Proc. of Europar 2003: Parallel Processing*, volume 2790 of *Lect. Notes in Comp. Science*, pages 1291–1294. Springer Verlag, August 2003.

[10] John Mugler, Thomas Naughton, Stephen L. Scott, Brian Barret, Andrew Lumsdaine, Jeffrey M. Squyres, Benoît des Ligneris, Francis Giraldeau, and Chokchai Leangsuksun. Oscar clusters. In *Linux Symposium*, Ottawa, Ontario, Canada, July 2003.

[11] openSSI.org. *Introduction to the SSI Cluster*. http://www.openSSI.org/.

[12] Philip M. Papadopoulos, Mason J. Katz, and Greg Bruno. Npaci rocks: tools and techniques for easily deploying manageable linux clusters. *Concurrency and Computation: Practice and Experience*, 15(7-8):707–725, 2003.

[13] T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, and C. V. Packer. BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 24th International Conference on Parallel Processing*, pages I:11–14, Oconomowoc, WI, 1995.

[14] V. S. Sunderam. PVM: A framework for parallel distributed computing concurrency. In *Practice and Experience*, volume 2, pages 315–339, December 1990.

[15] T. Takahash, F. O'Carroll, H. Tezuka, A. Hori, S. Sumimoto, H. Harada, Y. Ishikawa, and P.H. Beckman. Implementation and evaluation of MPI on an SMP cluster. In *Parallel and Distributed Processing. IPPS/SPDP'99 Workshops*, volume 1586 of *Lecture Notes in Computer Science. Springer-Verlag*, April 1999.

[16] Geoffroy Vallée, Renaud Lottiaux, Louis Rilling, Jean-Yves Berthou, Ivan Dutka-Malhen, and Christine Morin. A case for single system image cluster operating systems: Kerrighed approach. *Parallel Processing Letters*, 13(2), June 2003.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifi que,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399