Kerrighed: A Single System Image Cluster Operating System for High Performance Computing

Christine Morin[1], Renaud Lottiaux[1], Geoffroy Vallée[2], Pascal Gallard[1], Gaël Utard[1], R. Badrinath[1,3], and Louis Rilling[4]

[1] IRISA/INRIA – PARIS project-team
[2] EDF
[3] IIT Kharagpur
[4] ENS-Cachan, antenne de Bretagne

Abstract. Kerrighed is a single system image operating system for clusters. Kerrighed aims at combining high performance, high availability and ease of use and programming. Kerrighed implements a set of global resource management services that aim at making resource distribution transparent to the applications, at managing resource sharing in and between applications and at taking benefit of the whole cluster resources for demanding applications. Kerrighed is implemented as a set of modules extending the Linux kernel. Legacy multi-threaded applications and message-passing based applications developed for an SMP PC running Linux can be executed without re-compilation on a Kerrighed cluster. The proposed demonstration presents a prototype of Kerrighed running on a cluster of four portable PCs. It shows the main features of Kerrighed in global memory, process and stream management by running multi-threaded and MPI applications on top of Kerrighed.

1 Topics

We propose to demonstrate a prototype of Kerrighed[1], a single system image operating system for clusters. The research work presented in this prototype has been carried out in the PARIS project-team at IRISA/INRIA and relates to several topics:

Topic 01: Support Tools and Environments. A single system image operating system like Kerrighed can be considered as a tool or an environment to conveniently and efficiently execute parallel applications on clusters.

Topic 03: Scheduling and Load Balancing. Kerrighed's prototype implements a configurable global scheduler to balance the load on cluster nodes. Kerrighed's global scheduler is based on novel efficient process management mechanisms.

[1] Kerrighed (previously named Gobelins) has been filed as a community trademark.

H. Kosch, L. Böszörményi, H. Hellwagner (Eds.): Euro-Par 2003, LNCS 2790, pp. 1291–1294, 2003.

Topic 09: Distributed Algorithms. The prototype of Kerrighed implements the container concept for efficient global memory management. Based on containers, Kerrighed provides shared virtual memory segments to threads executing on different cluster nodes, a cooperative file cache and remote memory paging.

Topic 14: Routing and Communication in Interconnection Networks. Kerrighed's prototype implements a portable high performance reliable communication system providing a kernel level interface for the implementation of Kerrighed's distributed system services. The standard communication interface used by communicating Linux processes (pipe, sockets, etc...) is also available in Kerrighed on top of this communication system. In Kerrighed, communicating processes can be transparently migrated and can still efficiently communicate with other processes after migration.

2 Originality of the Demonstrated Prototype

The main originality of the prototype we propose to present is that it is a single system image operating system for clusters. Kerrighed provides the same interface as the standard operating system running on each cluster node. The current prototype has been implemented as a set of Linux modules and a small patch to the kernel (less than 200 lines of code, mainly for exporting kernel functions). Hence, existing applications that have been developed for an SMP PC running Linux can be executed on a cluster without even being recompiled. Unix applications can be easily ported on Kerrighed by recompiling them. So, sequential processes requiring huge processing and/or memory resources, multi-threaded applications and parallel applications based on message passing can be easily and efficiently executed on a cluster running Kerrighed. Unlike other systems Kerrighed supports efficiently both the message-passing and the shared memory programming models on clusters.

There are very few other research projects working on the design and implementation of a single system image operating system for clusters. Mosix [3] and Genesis [1] are examples of operating systems targeting the single system image properties. Mosix offers processor load balancing on top of a cluster. However, it does not support memory sharing between threads or processes executing on different cluster nodes unlike Kerrighed. A process which has migrated in Mosix cannot communicate efficiently with other processes after migration as messages are forwarded to it by the node on which it was created. Moreover, processes in Mosix cannot take benefit of the local file cache of their current execution node after migration, leading to poor performance for file accesses. Kerrighed has none of these drawbacks as it preserves direct communications between processes even after migration and as it implements a cooperative file cache.

Genesis is a single system image operating system for clusters which, in contrast to Kerrighed, has been developed from scratch and is based on the microkernel technology. As Kerrighed it supports both the shared memory and the message-passing programming paradigms. However, it implements a distributed

shared memory system which does not provide a standard interface. Thus, legacy shared memory parallel applications (such as Posix multi-threaded applications) cannot be executed on Genesis without a substantial porting effort. To our knowledge, Genesis only supports PVM for message-passing parallel applications.

3 Mechanisms for Demonstrating the Prototype

We will show a cluster of five portable PCs interconnected by a Fast Ethernet network. Four nodes of the cluster will run Linux and the Kerrighed modules. On this cluster, we will show the execution of instances of two parallel applications, one being a multi-threaded application, Volrend, the other one being an MPI application. Volrend is a multi-threaded application which implements ray-tracing to display 3D images. The fifth node executes a standard Linux system and is used to display the applications' results and to show some performance values (processor load, memory occupation and network throughtput for instance) as well as the location of all applications' threads and processes on the cluster's nodes (graphical interface with different colors for the different applications).

Kerrighed's features that are demonstrated are the container concept for memory sharing between threads (Volrend), the process migration mechanism which allows to migrate any process or thread even if it communicates with other threads or processes by shared memory (Volrend) or by message-passing (Mandelbrot fractal), the configurable global scheduler in which the scheduling policy can be changed without stopping Kerrighed and the applications currently running on top of it.

Note that a demonstration of an initial prototype of Kerrighed (not including process management and global scheduling features) has been successfully demonstrated at Supercomputing in Baltimore (USA) in November 2002 and at Linux Expo in Paris in February 2003 on the same cluster of portable PCs. The demonstration is stand-alone and does not require an Internet access.

4 Scientific Content

Our goal in designing Kerrighed is to combine high performance, high availability and ease of use [5]. Kerrighed performs global resource management. With a set of global resource management services, Kerrighed aims to make resource distribution transparent to the applications, to manage resource sharing in and between applications and to take benefit of the whole cluster resources for demanding applications. These services are distributed and each of them is in charge of the global management of a particular resource (memory, processor, disk).

Global memory management in Kerrighed is based on the container concept [4]. In modern operating systems, block devices and memory are managed on page granularity. A container is a software object allowing to store and share pages cluster wide. Containers are managed in a software layer in between high level services of the node's standard operating system (virtual memory, virtual

file system) and its device managers. Thus, the standard user interface is kept while Kerrighed offers virtual shared memory segments, a cooperative file cache and remote paging, all based on containers.

Kerrighed's global process management service consists in a global scheduler based on efficient process state management mechanisms. As characteristics of the workloads executed on clusters may differ from one environment to another, Kerrighed's global scheduler has been designed to allow the specialization of the scheduling policy [6]. Changing the global scheduling policy can be done dynamically without rebooting the cluster nodes. The description of the scheduler to be used in a particular environment need only be described using XML configuration files. Kerrighed implements a basic primitive to extract the state of a process from the kernel. This primitive is exploited to perform process duplication, migration and check-pointing [7,2]. Processes exchanging messages can also be efficiently migrated as Kerrighed implements migrable sockets that ensure a direct communication between two communicating processes even after migration of one of them. These sockets are implemented on top of a dynamic stream service that allows processes to attach or detach from a stream.

Kerrighed is an open source software distributed under the GNU GPL license. Scientific publications and a first (demonstration) version can be downloaded at http://www.kerrighed.org.

References

1. M.J. Hobbs A.M. Goscinski and J. Silock. Genesis : The operating system managing parallelism and providing single system image on cluster. Technical Report TR C00/03, School of Computing and Mathematics, Deakin University, February 2000.
2. Ramamurthy Badrinath and Christine Morin. Common mechanisms for supporting fault tolerance in DSM and message passing systems. Rapport de recherche 4613, INRIA, November 2002.
3. Amnon Barak, Shai Guday, and Richard G. Wheeler. *The MOSIX Distributed Operating System*, volume 672 of *Lecture Notes in Computer Science*. Springer, 1993.
4. Renaud Lottiaux and Christine Morin. Containers: A sound basis for a true single system image. In *Proceeding of IEEE International Symposium on Cluster Computing and the Grid (CCGrid '01)*, pages 66–73, Brisbane, Australia, May 2001.
5. Christine Morin, Pascal Gallard, Renaud Lottiaux, and Geoffroy Vallée. Towards an efficient single single system image cluster operating system. In *ICA3PP*, 2002.
6. Geoffroy Vallée, Christine Morin, Jean-Yves Berthou, and Louis Rilling. A new approach to configurable dynamic scheduling in clusters based on single system image technologies. In *International Parallel and Distributed Processing Symposium*, April 2003.
7. Geoffroy Vallée, Christine Morin, Jean-Yves Berthou, Ivan Dutka Malen, and Renaud Lottiaux. Process migration based on Gobelins distributed shared memory. In *Proc. of the workshop on Distributed Shared Memory (DSM'02) in CCGRID 2002*, pages 325–330, Berlin, Allemagne, May 2002. IEEE Computer Society.