Project no. IST-033576

XtreemOS

Integrated Project
BUILDING AND PROMOTING A LINUX-BASED OPERATING SYSTEM TO SUPPORT VIRTUAL ORGANIZATIONS FOR NEXT GENERATION GRIDS

Design and implementation of scalable mechanisms in LinuxSSI

D2.2.2

Due date of deliverable: November 30^{th}, 2007
Actual submission date: November 30^{th}, 2007

Start date of project: June 1^{st} 2006

Type: Deliverable
WP number: WP2.2
Task number: T2.2.2

Responsible institution: NEC
Editor & and editor's address: Erich Focht
NEC High Performance Computing Europe
Hessbruehlstr. 21b
70565 Stuttgart
Germany

Version 1.0 / Last edited by Erich Focht / September 20^{th}, 2007

Project co-funded by the European Commission within the Sixth Framework Programme		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision history:

Version	Date	Authors	Institution	Section affected, comments
0.1	4.10.07	Erich Focht	NEC	initial version of the document
0.2	24.10.07	Erich Focht	NEC	most of the text added
0.3	29.10.07	Erich Focht	NEC	almost finished full draft
0.4	30.10.07	Erich Focht	NEC	added hardcoded params part
0.5	26.11.07	Erich Focht	NEC	switched to new coverpage style
0.6	26.11.07	Erich Focht	NEC	modifications according to reviewers comments
1.0	30.11.07	Erich Focht	NEC	updated version number for delivery

Reviewers:
Michael Schoettner (UDUS), Samuel Kortas (EDF).

Tasks related to this deliverable:

Task No.	Task description	Partners involved°
T2.2.2	Building scalable SSI mechanisms	NEC*
T2.2.4	Design and implementation of advanced reconfiguration mechanisms	INRIA*
T2.2.10	Support high performance network devices	NEC*

°This task list may not be equivalent to the list of partners contributing as authors to the deliverable
*Task leader

Abstract

LinuxSSI-XOS/Kerrighed aims at simplifying cluster administration and programming as well as increasing parallel program performance on clusters by the use of single system image techniques. With the raise of multi-core processors and the general increase of cluster sizes scalability problems are expected to become a limiting factor in the usability of LinuxSSI-XOS. Task 2.2.2 from workpackage WP2.2 was dedicated to identify and work towards overcoming scalability problems.

The work of task T2.2.2 was attributed a deliverable (the present document) which is due at month 18 of the XtreemOS project. But scalability improvements are a continuous effort which needs to be sustained throughout the entire project duration, therefore this deliverable can only report on the current state of the work in progress and document some of the results. The report describes backgrounds of developments and sketches their progress without going into deep technical details, listing code, repository commit diffs or log messages.

Chapter 1 motivates the work on scalability and lists the areas on which the improvement work is focussed.

Chapter 2 summarizes the results of an early assessment of Kerrighed from the point of view of scalability.

The following chapter (3) describes the improvements done during the first 18 months of the project: rewrite of the low level network layers, 64 bit porting progress, port to SMP and algorithmic improvements.

The document is completed by a short concluding chapter. It shows that huge progress was made on most of the identified scalability issues, for example allowing to run clusters with more than 256 CPUs from SMP nodes in single system image mode.

Contents

Chapter 1

Introduction

The task of improving LinuxSSI-XOS / Kerrighed scalability was initially not linked to a deliverable. The work mainly consists of identifying scalability problems, convincing the developer community of their importance, developing or helping to develop solutions for the scalability problems and, if possible or necessary, implementing or helping to implement some of the solutions. This document describes the progress made during the first 18 months of the project.

The targets set by LinuxSSI-XOS scalability requirements are correlated to general HPC cluster technology development. During the project's lifetime top performance clusters will reach petaflops range, therefore LinuxSSI-XOS needs to aim at supporting clusters with hundreds or even thousands of nodes. The component nodes should be able to use several multi-core 64 bit processors. The nodes will be interconnected by specialized high speed interconnects. With this large number of components the mean time between failures will increase dramatically and LinuxSSI-XOS should be able to handle such failures gracefully by dynamically reconfiguring the SSI cluster in case of failures or intentional node addition or eviction.

Scalability impacts on several aspects of Kerrighed: SMP platforms, 64 bit CPUs, algorithm choice and implementation, dynamic reconfigurability and high speed interconnects. In the specification document for LinuxSSI-XOS [2] an assessment of the scalability status was done. The results are described in more detail in chapter 2. For the first 18 months of the project the targets in decreasing order of priority were:

- Dynamic reconfigurability: survive single node failure, dynamic node addition and eviction. Allow clean shutdown of the SSI cluster. Very high priority.

- SMP nodes support: Add support for shared memory parallel nodes and multicore CPUs. Very high priority.

- 64 bit processor support: add support for x86_64 CPUs. High priority.

- Removing hardwired parameters: Get rid of limitation to 128 nodes and increase number of containers in the system. High priority.
- Functional scalability: prepare benchmarking methodology for detecting issues in this area. Medium priority.
- Support for high speed interconnects: add support for the OpenIB infiniband stack. Medium priority.

The task of improving the scalability of LinuxSSI-XOS was to work towards eliminating the limitations listed above by convincing the Kerlabs members and raising awarenes among the XtreemOS members, as well as by changing the code where possible.

The expected changes for scalability improvements were huge, involving major rewrites. Some of the work has been finished successfully and most of the objectives were reached, but the scalability improvemnt task is continuous and cannot be considered to be finished at any time. This document describes the current status, the achieved progress, and also work in progress.

Chapter 2

Early Assessment

2.1 Methodology

As part of the planning for the LinuxSSI subproject and as part of the deliverable D2.2.1 an audit of Kerrighed has been done during August - September 2006. The SVN revision of the kerrighed code was $\leq r600$. The audit contained an assessment of scalability problems.

At the time of the assessment the stability of the Kerrighed code was very poor and practical measurements on bigger clusters were impossible. Therefore the investigation was mainly carried out by examining the source code and trying to anticipate problems which would appear on real hardware. This approach lead to a better understanding of the code base but possibly didn't uncover all scalability issues.

2.2 Results

The following potential scalability issues have been identified:

- SMP: No support for shared memory parallel nodes.
- 64 bit support: No support for 64bit CPUs.
- Hardcoded parameters: Certain parameters and dimensions are hardcoded and will lead to problems when increasing the number of nodes.
- Dynamic reconfigurability: Dynamic node addition and eviction not supported.
- Functional scalability: Potential problems due to non-scalable code and algorithms.
- High speed interconnects: Only TCP/IP stack is currently supported. Native support for more advanced high speed interconnects is required.

2.2.1 SMP support

The motivation for SMP support is twofold: (1) the CPU development is currently aiming at multi-core processors, and (2) most server platforms have at least two CPU sockets. Therefore the single CPU machines will pretty soon become very rare.

Kerrighed was not coded with SMP support in mind. Currently the kernel and the kerrighed.ko module do not build when CONFIG_SMP is enabled. The reason for this is missing implementations of certain functions in the SMP path.

Because Kerrighed was coded with a uni-processor (UP) platform in mind, the protection and access to all shared data-structures as well as the logic of RPC calls must be re-evaluated and checked for SMP design problems like missing locks and potential race conditions.

2.2.2 64 bit architectures support

Problem and program sizes are growing and request more and more memory. Kerrighed was initially developed on 32 bit IA32 processors which are inherently limited to a 4GB virtual address space. Support for 64 bit processors is therefore a memory space scalability question.

With 64 bit processors like AMD Opteron and Intel Nocona and Intel Woodcrest (Xeon 5100) becoming the main deployed server processors, the support for the x86_64 CPU architecture is mandatory. This is correlated with the growing demand for bigger memory spaces seen in today's applications, which can only be covered by the 64 bit virtual address space of 64 bit processors.

The 64 bit port is currently in work by Arkadiusz Danilecki from Poznan University. During compilation the code shows many warnings of integers treated like pointers, but the kernel has booted and very basic functionality like global process view was working.

2.2.3 Limits through hardcoded Parameters

Kerrighed has several hardcoded parameters which will lead to scalability limitations for bigger clusters. The currently identified ones are:

- Number of Kerrighed nodes: this number is limited to seven bits, which leads to an upper limit of 128 nodes in a Kerrighed cluster. The limitation is visible for example when dealing with global process IDs.
- Node addressing map: the node addressing is currently done with a static MAC address map.
- Hash-tables: The hash tables are used for accessing several variables like container pointers, RPC callback function pointers, etc. The size of a hash-table is fixed at compile time. With all Kernel Distributed Data Management (KDDM) sets being addressed through one hash-table we foresee a limitation

to the address space when using one set per file and per process. The access speed for finding KDDM set metadata slows down as the number of KDDM sets increases.

2.2.4 Dynamic Reconfigurability

At the time of the audit Kerrighed could only run with a number of nodes which was specified at boot time. Each booting node received the number of nodes in the cluster and its own ID on the kernel boot command line. The failure of one node lead to a lockup in the entire cluster. A clean shutdown of the cluster was not possible, this leads to filesystem corruption.

Meanwhile the reconfigurability is included into Kerrighed through the Hot-Plug component. The system can tolerate the failure of one node at a time. A clean shutdown is not yet possible (svn r800).

2.2.5 Algorithmic and functional limits

Until now Kerrighed experiments have been carried out on relatively small clusters (less than 32 nodes). Scalability issues due to non-scalable algorithms were not significant. A quick look into the code showed that such problems are around and might become significant. Example: the broadcast of a message to all cluster nodes is coded serially, the handling of the */proc* global process view is ineffective and might lead to slowing down significantly a big cluster.

It will be useful to prepare a benchmarking methodology in order to localize and quantify such performance and scalability problems.

2.2.6 High speed interconnects support

Currently Kerrighed uses exclusively the **netdevice** interface for communication. This is working well and reliably with Ethernet devices was also tested on Myrinet a while ago. No native support for Infiniband, Myrinet or Quadrics is available. For big HPC clusters of hundreds or thousands of nodes the support for some high speed interconnect is desirable.

Chapter 3

Improvements and Status

The development of Kerrighed / LinuxSSI was significantly speeded up in the past year. Several developers from the XtreemOS consortium have joined forces with the Kerlabs developers, who are the originators and maintainers of the Kerrighed core code. The Kerrighed community has grown and the code was exercised by more testing. This resulted in probably four releases of Kerrighed in year 2007:

- Version 2.0.0: March 2007. Final release of port to kernel version 2.6.11.
- Version 2.1.0: June 2007. Port to kernel version 2.6.20.
- Version 2.1.1: August 2007. Incorporated fixes to previous version.
- Version 2.2.0: November 2007 (?). SMP support.

The increased activity around Kerrighed resulted in several scalability improvements. The effort of preparing Kerrighed for big systems is continuous and work in progress.

3.1 Network layer

3.1.1 Problems

The initial communication stack of Kerrighed was built in three layers:

- legolas: the RPC layer, was renamed to “rpc”.
- gimli: the communication protocols layer, was renamed to “comm”.
- gloin: the low level network layer, was renamed to “kernetdev”.

The lowest layer was the network communication layer, implemented on top of the kernel network device knetdev. It is a low overhead, simple, non-routed and unsecure network communication layer and Kerrighed has added to it a new protocol for its messages. Some of the scalability limitations originate from the need

to re-invent and implement cluster communications primitives on top of this layer. For example: the Kerrighed network stack had 256 slots for cluster nodes. The Kerrighed code was using the fixed number given by the slots in many places, for example as node vectors for filtering access to online nodes where hardcoded to allow a maximum of 256 nodes.

3.1.2 TIPC

Many of the Kerrighed functionalities of the old communication layers were already present in different forms in the Linux kernel, for example in pieces of distributed filesystem codes like GFS or OCFS2 and in the distributed lock manager. The acceptance of the Kernel community for the functionality re-invented and re-implemented by Kerrighed was rather improbable, particularly with the hardcoded limitations. Therefore we started searching for existing components which deliver similar functionality as required by Kerrighed: define some notion of cluster and cluster membership, offer some way to implement easilly multiplexed RPCs and are flexible and scalable with respect to the number of cluster member nodes. The most appropriate candidate seemed to be TIPC [3].

TIPC is an acronym for **Transparent Inter-Process Communication** and is a communication protocol specially designed for intra-cluster communication. It was developed at Ericsson and was used by the company for carrier-grade cluster applications before it was released into the open source. TIPC has some interesting features:

- location transparency of services in a cluster,
- cluster node auto-discovery and connection monitoring mechanisms,
- reliable transport, reliable multicast,
- allows subscription to network events,
- includes a standard socket interface as well as a lower level API,
- supports connectionless, connection-oriented and multicast messaging.

TIPC nodes periodically broadcast messages on their network media to detect neighboring cluster nodes. They establish logical point-to-point connections to live nodes which belong to the same cluster and permanently monitor the state of the links.

The addressing scheme used by TIPC is cluster centric and is mapped to its logical network topology: nodes with point-to-point links to each other build clusters, clusters able to communicate to each other are grouped in zones. Addresses can be written in the form [Zone, Cluster, Node] and are represented by 32 bit integers. The most significant 8 bits build the zone number, the next 12 bits are the cluster number and the least significant 12 bits represent the node number.

TIPC is included into the Linux mainline since kernel version 2.6.16.

3.1.3 LinuxSSI Integration with TIPC

The switch of the Kerrighed / LinuxSSI network layer to TIPC was discussed at several occasions between NEC and Kerlabs and finally done during March - April 2007 mainly by Pascal Gallard (Kerlabs). It has solved a multitude of scalability problems:

- The design of the communication layer was simplified by reusing features already available in TIPC.
- TIPC has lifted the limitation on the number of nodes in a cluster. The limit was raised theoretically from 256 to 4096 nodes. The practical maximum depends of the kernel configuration parameters for TIPC, which are by default setting the maximum node number to 256. Therefore the limit is not hardcoded any more, but has to be set at kernel compile time.
- Multiplexed RPCs are now set up by using TIPC low level API functionality.
- During the process of adapting the Kerrighed network layer to TIPC the network communication mid-layer was cleaned and all data exchanges were adapted to use the pack/unpack protocol. This helped to solve problems with locking under SMP.
- The use of TIPC membership and network events subscription functionality improved the design of the hotplug layer and its capabilities. Node hot-add and autoconfiguration with TIPC mechanisms was added.

An overall positive effect was the removal of the kernetdev layer and simplification of code, thus helping to lower the future maintenance effort of Kerrighed.

3.2 64 bit architectures support

The motivation for providing support for 64 bit CPUs is that *all* server processors produced today are 64 bit processors. The big majority of them is compatible to the x86_64 architecture, the 64 bit extended variant of the IA32 CPUs. Other 64 bit CPU architectures commonly used in high performance computing are: ia64, ppc64, power, sparc64.

Kerrighed was developed on IA32 architectures and aims at providing support for x86_64. Porting to a 64 bit architecture requires the adaptation of the architecture and platform specific functions and include file to Kerrighed, changes in the build system and – most important and difficult – detect and adapt variables which are used and appropriate on 32 bits but lead to failures on 64 bit systems. These can originate from the use of 32 bit datatypes, but also from algorithms which were designed with 32 bit addressing in mind.

The work for x86_64 support was started by Arkadiusz Danilecki from the University of Poznan, who was collaborating with the Kerrighed team. Basic support

was added for Kerrighed versions based on 2.4 kernels and partially forward ported to kernel 2.6.11. During the port to kernel version 2.6.20 the 64 bit support had low priority. After the 2.6.20 port Erich Focht from NEC has worked on adding x86_64 support. The rest of this section outlines the steps done and the current status of the work.

The kernel build system had to be adapted to include the Kerrighed build system under the architecture x86_64.

The kernel debugger kdb, which is used for i386, had to be integrated with its 64 bit parts for x86_64 support.

The first build attempts then revealed the easy-to-fix problems in the source code, like:

- missing include files due to different logic under 64 bits,
- missing or not properly adapted system calls,
- missing exported symbols under 64 bits,
- static functions which need to be global under Kerrighed.

The attempts to boot and run the ported 64 bit Kerrighed code revealed a set of problems originating in changed variable values and ranges under 64 bits. In one case a variable type has implicitely changed from 32 bits long to 64 bits long. The 32 bit code was working, but turned out to be wrong for 64 bits, where values like `0x00000000ffffffff` are not interpreted as negative numbers.

The most involved change was the debugging and re-write of the algorithm that sorts and locates kddm-set indexes. It was organised as a two or four level tree, two level tree when indexes could be represented as a 20 bit number, four level tree for 32 bit indices. The hardcoded search methods were rewritten to be able to deal with arbitrary index widths, offering predefined structures for 20, 32 and 64 bit indices. The problem was hard to detect, because kddm_set indices stayed often below the 32 bit limit, so everything worked fine. Mostly.

The support for x86_64 was tested with the set of LTP (Linux Test Project) [4] tests, which are used for regular Kerrighed testing. The last tests were done in June 2007, when all tests suited for Kerrighed were passing on x86_64. At the end of the SMP porting work the 64 bits port will need to be verified again.

The Kerrighed development is expected to switch to 64 bits at the beginning of the year 2008. This will ensure that x86_64 support will be at the same level of correctness as the 32 bit version.

3.3 SMP support

Kerrighed is a distributed operating system and as such a highly complex distributed program where variable values and system states are spread over multiple nodes and instances of the Linux operating system.

When built on top of a UP (uni-processor) kernel the Linux operating system instances on the compute nodes are effectively serializing the operations done on each node. This allows to simplify the programming and often ignore locking mechanisms for protecting variables against concurrent access on the same node. Kerrighed was developed for single processor machines running a UP Linux kernel.

Nowadays multiprocessor machines are common, multi-core CPUs are built even into Laptops. The demand for supporting SMP (symmetric multi-processor) Linux kernels is huge, therefore development for SMP Kerrighed was started right after the 2.6.20 port was finished. It has been mainly done by the developers from Kerlabs.

At the beginning of the SMP port the KDDM layer was not SMP aware, i.e. it had almost no intra-node locking mechanisms implemented. Adding locks wasn't simple, either, because the way how communication protocols were mainly used in the KDDM layer was not designed to be done in parallel. The code was improved and KDDM has switched to new RPCs and communication protocols, such that it is now SMP aware. This work was done by Renaud Lottiaux, Pascal Gallard and Louis Rilling from Kerlabs.

The old communication layer was written with SMP in mind, but the switch to TIPC has introduced new problems. Pascal Gallard from Kerlabs has discovered SMP bugs in TIPC and fixed them.

The PROC layer dealing with distributed process management, signal forwarding, process checkpointing, migration, etc. was partially supporting SMP but needed to be reviewed and checked for SMP compliant logic. This work was done by Louis Rilling from Kerlabs.

The SMP port is still work in progress. Kerrighed passes many LTP tests but there are several SMP related bugs open in the bug tracker [1].

3.4 Hardcoded parameters

3.4.1 Global process IDs

On one stand-alone Linux instance the numeric process IDs (PIDs) are limited to 32768 on 32 bit architectures, and to 4194304 on 64 bit machines. On a distributed operating system like LinuxSSI these limitations would be too strong for global process IDs. Kerrighed had hardcoded 7 bits for node identification in an SSI cluster for defining global proces IDs. This effectively limited the SSI cluster size to 128 nodes.

In the process of porting Kerrighed to the x86_64 architecture Erich Focht from NEC has cleaned up the way how cluster bits and local PID bits are defined. The numeric PID data type is *integer*, with two bits being reserved for futex use. The number of PID bits for node IDs is now configurable as
`NR_BITS_IN_MAX_NODE_ID` in the file *include/kerrighed/sys/types.h* , leaving $30 - 1 -$ `NR_BITS_IN_MAX_NODE_ID` bits for local PID values. This allows

lifting the limit for the number of cluster nodes, on the expense of the local number of PIDs.

3.4.2 KDDM-set namespaces

The addressing of KDDM sets was organized by storing the set IDs in a hash-table. As described in chapter 2, the hash-table represents a limitation to the number of KDDM sets and slows down access with increasing number of entries.

During the first quarter of 2007 Renaud Lottiaux from Kerlabs has added an additional addressing level to the KDDM sets. Instead of just using a set number (32 or 64 bit long, depending on architecture) now an additional integer called *name space* can be used. Name spaces can be created by kernel applications which use KDDM sets, for example by the checkpointer or the distributed filesystem. Each of the name spaces owns its own hash-table. This way the limitation in the address space has been removed and the addressing speed scalability was improved.

3.5 Algorithmic improvements

Some scalability limitations are given by the choice or implementation of distributed algorithms. Since we're still not testing on really big clusters the Kerrighed code might hide such problems. Some of them were uncovered and fixed during the past year. This section documents some of the progress made.

Some algorithmic changes which helped improving scalability were already mentioned in the previous sections:

- The improvement of the KDDM tree for searching indices of kddm sets.
- The improvement of KDDM communications. The change of communication protocol to new RPCs and pack/unpack has allowed adding correct locking for SMP support.

An example for a distributed algorithm that brought scalability improvements is a new protocol for removing KDDM objects which was implemented by Renaud Lottiaux in July 2007. He also added an in-kernel benchmarking module for measuring the performance of KDDM operations. This way performance regressions when changing KDDM algorithms can be avoided in future.

Chapter 4

Conclusion and Future Work

The first 18 months of the XtreemOS - LinuxSSI project were very successful in speeding up the Kerrighed development. One of the major results is that the core scalability limitations were identified and work has been done to remove them. The biggest achievements are:

1. The network stack was replaced with TIPC, a component from the mainline Linux kernel which reduces the size of Kerrighed, improves the hotplug design, makes autoconfiguration possible and lifts the limitation of the number of cluster nodes.

2. The port to the x86_64 64 bit architecture is basically done. This allows supporting modern processors and bigger memory spaces.

3. The SMP port is finished, support for multi-processor machines and multi-core CPUs will is currently possible, Kerrighed has been shown at Super-Computing 2007 running on a cluster with 64 nodes of 4 CPU cores, each, thus aggregating 256 CPUs in one single system image.

4. Some algorithmic changes helped improve performance and scalability.

Still, the scalability improvement task is work in progress and will probably never be finished. The next major step will be to add support for high speed interconnects like Infiniband. This work contributes to the LinuxSSI-XOS scalability but is attributed to a separate XtreemOS WP2.2 task and targetted to be finished in month 30 (M30) of the project.

Bibliography

[1] Kerrighed bug tracker. https://gforge.inria.fr/tracker/?group_id=69.

[2] C. Morin, D. Margery, E. Focht et al. Specification of federation resource management mechanisms in xtreemos - d2.2.1. 2007.

[3] TIPC project website. http://tipc.sourceforge.net/.

[4] Linux Test Project website. http://ltp.sourceforge.net.