Sistema de Virtualización con Recursos Distribuidos

Pablo Pessolani
Departamento de Ingeniería en Sistemas de Información
Facultad Regional Santa Fe
Universidad Tecnológica Nacional
Santa Fe, Argentina
ppessolani@frsf.utn.edu.ar

Toni Cortes*
Barcelona Supercomputing Center y
Departamento de Arquitectura de Computadores
Universitat Politécnica de Catalunya
Barcelona - España
toni.cortes@bsc.es

Silvio Gonnet
CIDISI- Instituto de Desarrollo y Diseño (INGAR), CONICET
Universidad Tecnológica Nacional
Santa Fe - Argentina
sgonnet@santafe-conicet.gov.ar

Fernando G. Tinetti
III-LIDI
Facultad de Informática
Universidad Nacional de La Plata
La Plata, Argentina
Comisión de Inv. Científicas, Prov. Bs. As.
fernando@info.unlp.edu.ar

RESUMEN

Si bien los Sistemas Operativos disponen de características de seguridad, protección, gestión de recursos, etc. éstas parecen ser insuficientes para satisfacer los requerimientos de los sistemas informáticos actuales. Las tecnologías de virtualización existentes han sido y continúan siendo masivamente adoptadas para cubrir esas necesidades de los sistemas y aplicaciones por sus características de particionado de recursos, aislamiento, capacidad de consolidación, seguridad, soporte de aplicaciones heredadas, facilidades de administración, etc. Una limitante de las tecnologías de virtualización tradicionales es que el poder de cómputo de una Máquina Virtual está limitado al poder de cómputo de la máquina física que la contiene. Por otro lado, los Sistemas Operativos Distribuidos de Imagen Unica ofrecen un poder de cómputo que integra en forma transparente los recursos computacionales de los nodos que constituyen un cluster. Sus características de transparencia evita la reprogramación de las aplicaciones heredadas que fueron desarrolladas para sistemas centralizados. La línea de I/D aquí presentada se orienta hacia un Sistema de Virtualización con Recursos Distribuidos que permita construir Máquina Virtuales con mayor poder de cómputo que una máquina física combinando e integrando las tecnologías de virtualización y Sistemas Operativos Distribuidos de Imagen Unica.

Palabras claves: Virtualización, Máquinas Virtuales, Sistemas Operativos Distribuidos.

* Este trabajo está siendo subvencionado parcialmente por parte del Ministerio de Ciencia y Tecnología de España por la ayuda TIN2007-60625 y del Gobierno Catalán por la ayuda 2009-SGR-980

CONTEXTO

Este proyecto de I/D involucra a investigadores de varios laboratorios y centros de investigación (ver afiliaciones de los autores), en un área que por su amplitud requiere de múltiples enfoques. Más específicamente, se deriva de las tareas de investigación en el contexto de los programas de postgrado de la Facultad de Informática de la UNLP.

INTRODUCCION

La tecnología de virtualización desarrollada a fines de la década del ‘60 [1] ha resurgido a fines de los años ´90. Esto se debe a que han cambiado los modos de procesar la información, a los avances en el hardware y a las atractivas características que ésta tecnología ofrece tales como la capacidad para consolidar múltiples servidores virtuales en una misma computadora física, la implementación de *contenedores* aislados para la ejecución de aplicaciones con mayor seguridad y protección.

En los diferentes sistemas de virtualización la máquina virtual (abreviado VM en inglés) estará contenida en un computador, por lo que su poder de cómputo estará limitado al poder de cómputo del Host que la contiene. Recientemente han surgido tecnologías de virtualización que permiten integrar recursos de múltiples nodos para brindar la imagen de un sistema de Multi-Procesamiento Simétrico virtual o vSMP [2,3]. Esta tecnología equivale a conformar un Monitor Distribuido de Máquinas Virtuales que le virtualiza a los Sistemas Operativos (OSs) los recursos de hardware de todos los nodos del cluster en una imagen única de un computador SMP.

A diferencia de un vSMP que virtualiza hardware, un Sistemas Operativos Distribuidos de Imagen Unica (abreviado SSI-DOS en inglés) ejecuta procesos en forma distribuida virtualizando abstracciones de software. Estas abstracciones son idénticas o similares a las que brinda un

OS tradicional (centralizado) tales como archivos, tuberías (pipes), sockets, colas de mensajes, memoria compartida, semáforos, mutexes. Los usuarios y aplicaciones tienen la visión de un único computador virtual conformado por los recursos computacionales, procesos, abstracciones y servicios que se encuentran distribuidos en los distintos nodos componentes del cluster.

Podría interpretarse que un SSI-DOS también representa una forma de virtualización (*virtualización reversa*). A diferencia de la virtualización tradicional que permite crear múltiples computadores virtuales en un *único* computador físico, los SSI-DOS integran en un OS los recursos de múltiples computadores físicos en un *único* computador virtual.

Las tecnologías de Virtualización y SSI-DOS, aunque aparentan ir en direcciones opuestas, podrían integrarse en un Sistema de Virtualización con Recursos Distribuidos (DRVS) que permita disponer de VMs conformadas por recursos computacionales de diferentes nodos del cluster. La integración de éstas tecnologías en un DRVS representa un desafío y es el objeto del presente proyecto de I/D.

Los requisitos formales que debe reunir la tecnología de virtualización fueron enunciados por Popek y Goldberg [4] en 1974. Con el paso del tiempo, a medida que la tecnología se fue desarrollando, los términos *Virtualización* y *Máquina Virtual* pasaron a tener diferentes significados dependiendo del entorno en que se utilizan y de la forma en que se implementan. En la siguiente sección se presenta una taxonomía (parcial, por limitaciones de espacio) de tecnologías de virtualización existentes con una breve descripción de sus características (Figura 1). Esta taxonomía permitirá identificar las tecnología de virtualización en la que se basará el modelo de DRVS de este proyecto de I/D.

Taxonomía de Tecnologías de Virtualización

Virtualización de Hardware/Sistema

Las CPUs modernas disponen de un set de instrucciones (ISA: Instruction Set Architecture) en donde un subconjunto de ellas son instrucciones privilegiadas. Las VMs encapsulan OSs, denominados Guests (invitados), creándoles la ilusión de estar ejecutándose sobre un computador real. Los kernels de los OS-Guests creen disponer de todos los privilegios para operar sobre el hardware (modo kernel) aunque realmente se ejecutan con privilegios de modo usuario.

Si el OS-Guest ejecuta en modo usuario instrucciones privilegiadas tales como aquellas que se utilizan para controlar puertos de Entrada/Salida o inhabilitar interrupciones, al disponer de menos privilegios de los necesarios producirán fallos (Faults) a la CPU. Estos fallos serán tratados por el Hypervisor o Monitor de la VM (VMM) tomando el control de la ejecución para emular al hardware o para realizar esa operación en representación del OS-Guest.

El VMM puede ejecutar directamente sobre el hardware (Tipo-1) o integrado a un OS (Tipo-2). Esta tecnología es utilizada en productos tales como VMware, VirtualBox, QEMU, Parallels y Microsoft Virtual Server.

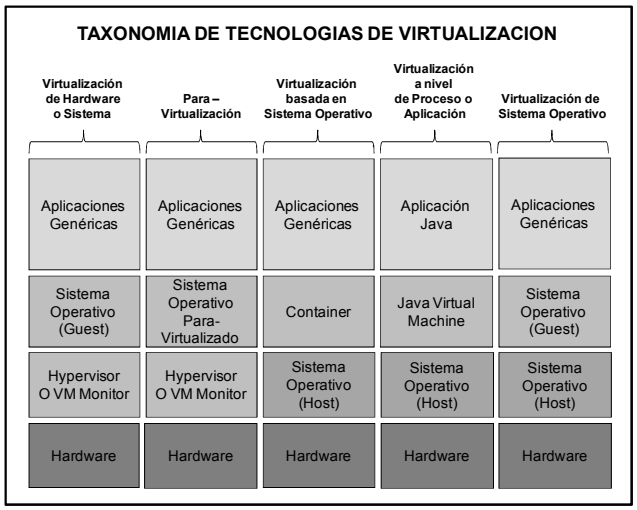

Figura 1. Taxonomía de Tecnologías de Virtualización.

Para-virtualización

En esta tecnología se requiere modificar el código del OS-Guest para que ejecute en un entorno virtualizado, que coopera con el VMM. De esta forma se evita el uso de instrucciones privilegiadas que provocan fallos y deben emularse. El OS-Guest (Sistema Operativo Para-Virtualizado en Figura 1) solicita servicios al VMM a través de las que se denominan *Hypervisor Calls*. Esta tecnología es utilizada por productos tales como Xen, Denali y Hyper-V.

Virtualización a nivel de Sistema Operativo

Los OSs ofrecen una Interfaz de Programación de Aplicaciones (en inglés APIs) y Llamadas al Sistema (System Calls) utilizadas por las aplicaciones para solicitar servicios al OS. La virtualización a nivel de OS consiste en interceptar las *System Calls* y alterar su comportamiento para simularle a la aplicación un entorno específico.

El sistema de virtualización construye un Contenedor (Figura 1) o Prisión (Jail) que aísla a cada una de las aplicaciones en su propio entorno de ejecución. Aparece como más adecuado referirse a *Entorno Virtual* y no *Máquina Virtual* dado que no se virtualiza Hardware. Con este mecanismo de virtualización se pueden ejecutar múltiples instancias del OS-Guest en forma aislada y segura pero éstas deben ser iguales al OS-Host. Esta tecnología es utilizada por OpenVZ, Virtuozzo, Linux-VServer, Solaris Zones y FreeBSD Jails.

Virtualización de Lenguaje/Proceso/Aplicación

La VM se ejecuta como un proceso de usuario dentro de un OS-Host y soporta una única aplicación. Se crea cuando el proceso arranca y se termina cuando éste finaliza. Su objetivo es brindar un entorno de programación y ejecución independiente de la plataforma, de los detalles del hardware y del OS subyacente. El ejemplo más conocido es la Máquina Virtual Java (JVM).

Virtualización de Sistema Operativo

Típicamente se denomina *Sistema Operativo* a la capa de software que se encuentra entre las aplicaciones y el hardware. En esta tecnología de virtualización el OS-Guest no se apoya directamente sobre el hardware sino sobre los servicios que le brinda el OS-Host. Parece adecuado referirse al OS-Guest como un *Sistema Operativo Virtual* (VOS). Tiene aspectos comunes con la tecnología de Para-virtualización, entre ellos la necesidad de modificar el OS-Guest, pero se diferencia de ésta en que un OS-Host lleva a cabo las funciones de un Hypervisor. User Mode Linux, NomadLinux y Minix Over Linux utilizan esta tecnología.

Trabajos Relacionados

El núcleo del proyecto de investigación requiere de la integración de al menos tres tecnologías existentes:

- Virtualización de OS y a nivel de OS.
- Sistemas Operativos Distribuidos de Imagen Unica.
- Sistemas Operativos Multi-servidor.

A continuación se resumen los trabajos relacionados acerca de éstas tecnologías que formarán parte de la base de conocimiento para esta línea de I/D del DRVS.

Virtualización de OS y a nivel de OS

Durante el trabajo de I/D se considerarán aspectos y aportes de las tecnologías de virtualización de hardware o sistema, tales como VMware, Virtual Box, etc. y de para-virtualización como la utilizada por Xen, Denali y Hyper-V. Sin embargo, el DRVS objeto del proyecto tomará en dirección de la virtualización de OS y a nivel de OS.

Un ejemplo de Virtualización de OS es UML [5]. UML permite que múltiples Linux-Guest puedan ejecutarse como una aplicación en un sistema Linux-Host. Como cada Guest es un proceso que se ejecuta en espacio de usuario, UML le proporciona a los usuarios una forma de ejecutar múltiples VMs Linux en un único computador sin requerir de privilegios de ***root***.

Linux-Vserver [6] se basa en los *Contextos de Seguridad* que construyen una abstracción denominada cárcel o prisión (Jail). Se pueden particionar recursos de cómputo de tal forma que la ejecución de un proceso dentro de una prisión no pueda afectar la ejecución en otra prisión.

OpenVZ [7] es una tecnología de virtualización a nivel de OS que utiliza Linux como Host. OpenVZ permite que se ejecuten múltiples instancias aisladas de un OS conocidas como Contenedores (Containers), VPSs, o Virtual Environments (VEs).

El paquete LXC [8] combina mecanismos en el kernel de Linux con Contenedores a modo de sistemas virtuales "livianos" en espacio de usuario con completo aislamiento y control de recursos de las aplicaciones. Fue construido a partir de ***chroot*** incorporando mecanismos de virtualización, de gestión y aislamiento en la infraestructura existente de gestión de procesos del kernel de Linux.

FreeBSD Jail [9] es un mecanismo que implementa Virtualización a nivel de OS permitiéndole a los administradores particionar un computador con FreeBSD en varias prisiones.

Minix over Linux (MoL) [10] es un sistema de Virtualización de OS desarrollado en UTN-FRSF con características comunes a UML. Es un VOS que emula las APIs, System Calls y servicios de un Minix y está compuesto de servidores, tareas de gestión de dispositivos, un kernel virtual y dispositivos virtualizados.

MoL se ejecuta enteramente en modo usuario sobre un sistema Linux (Host). Los procesos de usuario se ejecutan dentro de un Contenedor y no tienen acceso a ninguno de los recursos del Linux subyacente ni de otros procesos MoL a excepción de aquellos que le fueron asignados.

Sistemas Operativos Distribuidos de Imagen Unica

La Imagen Unica de Sistema (SSI) es la propiedad que oculta la naturaleza heterogénea y distribuida de los recursos disponibles y los presenta a usuarios y aplicaciones como un recurso computacional simple y unificado [11].

El V-System [12] es un SSI-DOS basado en microkernel. Uno de sus aportes más destacables es el mecanismo de transferencia sincrónica de mensajes entre procesos y el protocolo denominado Versatile Message Transaction Protocol (VMTP).

Amoeba [13] es un SSI-DOS de código abierto basado en microkernel desarrollado por Andrew S. Tanenbaum y otros. Habiendo sido desarrollado desde cero, pronto quedó sin mantenimiento ni actualizaciones provocando su desaparición temporal hasta que fue reflotado por el Dr. Stefan Bosse[14]. Bosse desarrolló un entorno denominado AMUNIX que virtualiza AMOEBA como Guest sobre un sistema tipo Unix estándar como Host.

MOSIX [15] es un SSI-DOS para clusters que se implementa sobre Linux como una capa de virtualización de OS (sandbox). En sus últimas versiones [16] MOSIX permite la conformación de clusters, multi-clusters y clouds. Entre sus innovaciones se ofrece la posibilidad de ejecutar MOSIX sobre VMWare integrando la migración de procesos y la migración de VMs [17].

OpenSSI [18] es un sistema implementado como una extensión de Linux que hace que un cluster de computadores se presente a las aplicaciones como un gran computador. Los procesos que se ejecutan en un nodo tienen pleno acceso a los recursos de todos los nodos y pueden migrar de un nodo a otro de forma automática.

Kerrighed [19] es un SSI-DOS que se basa en la tecnología de Memoria Compartida Distribuida (DSM) con un modelo de consistencia secuencial que le permite migrar hilos (threads). Los procesos son encapsulados en *Contenedores* donde la planificación local la realiza Linux, mientras que Kerrighed realiza su gestión a nivel de cluster. Kerrighed está implementado como una extensión de Linux que incluye módulos de kernel.

FOS [20] es un SSI-DOS cuyo principal objetivo está centrado en la escalabilidad y en el futuro (muy cercano) de los sistemas multi-core masivos y cloud. Está basado en microkernel y es multiservidor, pero a diferencia de otros OS con estas características, los servicios pueden ser brindados por un conjunto de servidores coordinados en paralelo denominados ***fleets***.

Sistemas Operativos Multi-servidor

En un OS-Multiservidor se utiliza el mecanismo de transferencia de mensajes para realizar las *System Calls*. Esto permite remotizar sus servicios a través de una red. Por otro lado, existe un menor acoplamiento entre el kernel y los procesos, facilitando los mecanismos de migración. Estas características los hace aptos para ser considerados en el desarrollo de sistemas distribuidos.

Minix [21] es un OS desarrollado de cero por el profesor Andrew S. Tanenbaum. Entre sus características principales se pueden mencionar que se basa en una arquitectura de microkernel, multi-capas, multi-servidor, y con mínimo código ejecutable en modo privilegiado lo que mejora su confiabilidad.

Existe versiones experimentales de Minix, que al igual que FOS [20] pueden trabajar en un Single Chip Cloud Computer [22], o como sistema distribuido [23] lo que demuestra sus versatilidad y capacidad de adaptación para enfrentar los nuevos desafíos que presenta el hardware a los diseñadores de OSs.

L4 [24] es un microkernel diseñado e implementado por Jochen Liedtke. El propósito de L4 era probar que los microkernels podían ser tan eficientes como los sistemas monolíticos. Un Linux modificado denominado L4Linux se ejecuta para-virtualizado sobre un hypervisor sin requerir privilegios ejecutándose en modo usuario.

LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

La línea de I/D de este proyecto refiere a desarrollar un modelo de DRVS. Estos trabajos requieren del conocimiento y dominio de tecnologías de Virtualización, de Sistemas Operativos Distribuidos de Imagen Unica y de Sistemas Operativos Multi-servidor.

Como parte del proyecto se desarrollará un prototipo de DRVS con un OS multi-servidor (Minix) como base para el desarrollo del OS-Guest (como DVOS) y modificaciones del kernel del OS-Host (Linux).

Sistema de Virtualización con Recursos Distribuidos

En un SSI-DOS se implementan dentro del propio sistema las políticas y mecanismos de balanceo de carga, migración de procesos, sincronización, locking, elección de líder, consenso, exclusión mutua, censado de parámetros de rendimiento, replicación, monitoreo y administración de recursos. El resultado es que los módulos de software que lo componen se encuentran fuertemente acoplados entre si.

En el modelo de DRVS propuesto se relaja el acoplamiento entre los diferentes módulos de software separando los servicios y funcionalidades en diferentes componentes de la arquitectura. Estos componentes son los siguientes (ver Figura 2):

- *Distributed Virtual Operating System (DVOS)*: Un DVOS está constituido por un conjunto de procesos que pueden estar distribuidos en los distintos nodos de un cluster físico. Un DVOS es una emulación de un OS multi-servidor adaptado para trabajar utilizando los servicios e interfaces que le ofrecen las DRVMs. Presentan las interfaces a las aplicaciones que emulan las APIs y System Calls de un OS original.

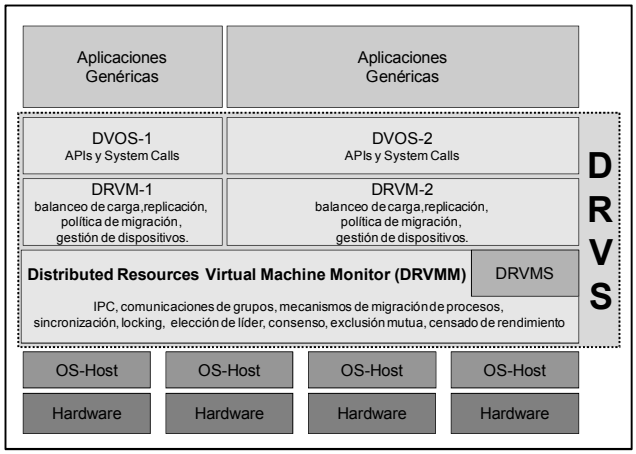

Figura 2. Sistema de Virtualización con Recursos Distribuidos

- *Distributed Resources Virtual Machine (DRVM)*: Es el entorno o subsistema en donde se ejecuta un DVOS. En la DRVM se gestionan los recursos (físicos o virtuales) que están distribuidos en los distintos nodos que le fueron asignados. La DRVM realiza el balanceo de carga y establece la política de migración de procesos para brindar mayor rendimiento y admite la replicación de procesos para mejorar la disponibilidad.
- *Distributed Resources Virtual Machine Monitor (DRVMM):* Es el software que integra todos los recursos del cluster, gestiona y limita los recursos asignados a las DRVMs. Es responsable de brindar a las DRVMs las interfaces para los protocolos de bajo nivel tales como IPC, comunicaciones de grupo, sincronización, locking, elección de líder, consenso, exclusión mutua, censado de parámetros de rendimiento, mecanismo de migración de procesos y virtualización de dispositivos.
- *Distributed Resources Virtualization Management System (DRVMS)*: Es el software que le permite al supra-administrador gestionar los recursos del cluster, asignar recursos a las DRVMs, y realizar diversos tipos de monitoreos y acciones sobre el DRVS.

La unidad de asignación de recursos a una DRVM no es el nodo por completo sino sus recursos físicos y virtuales en forma individual. Esta mayor granularidad en la asignación de recursos permitirá una mejor explotación de los mismos y facilitará su gestión.

La contribución esperada de esta línea de investigación es un nuevo modelo de sistema de virtualización que permita el particionado de los recursos de un cluster en VMs conformadas por recursos distribuidos en múltiples nodos.

A diferencia de un SSI-DOS, en un DRVS pueden ejecutarse diferentes DVOS. Esto se debe a que los servicios distribuidos y la transparencia (de recursos, de migración, de replicación, etc.) la brindan las DRVMs. Con este sistema se mantienen los beneficios tan apreciados de la virtualización

tales como aislamiento, consolidación y seguridad, y los beneficios de los SSI-DOS de transparencia, mayor rendimiento, disponibilidad y escalabilidad.

RESULTADOS Y OBJETIVOS

El objetivo primario del proyecto aquí presentado es desarrollar un modelo de DRVS. Se espera que con esta tecnología las DRVMs dispongan en forma transparente (para las aplicaciones y usuarios) del poder de cómputo y recursos computacionales de los nodos que las constituyen.

Como primer etapa del proyecto se ha desarrollado MoL [10] (en una versión experimental) con el objetivo de convertir a Minix [21] en un DVOS que ejecute como Guest de un Linux que actúa como Host.

La tecnología DRVS se presenta como apta para brindar servicios Cloud de tipo Infraestructura como Servicio (del inglés IaaS) porque admite balanceo de carga para mejorar el rendimiento, facilita el mantenimiento del hardware sin interrupción del servicio y permite reducir el consumo energético del cluster desconectando nodos ociosos utilizando la migración de procesos y DRVMs.

El DRVS permitirá el particionado de los recursos de un cluster para conformar múltiples DRVMs que podrán ser utilizadas y administradas por diferentes comunidades de usuarios. Los nodos del cluster físico podrán ser asignados para uso exclusivo de una DRVM en particular o compartir sus recursos entre múltiples DRVMs. Por ejemplo, en una Universidad con un único cluster, se podrán particionar los recursos de éste asignando diferentes DRVMs a los distintos departamentos, centros de investigación, facultades, etc.

Las DRVMs admiten reconfiguración (reasignación de recursos) en forma dinámica, esta característica le otorga una gran flexibilidad de escalamiento. Se podrán reasignar recursos de DRVMs ociosas a DRVMs con mayores necesidades de procesamiento logrando una explotación más eficiente de la infraestructura. De igual forma se permite remover nodos en forma dinámica para efectuar mantenimiento del equipamiento o para reducir el consumo energético, y luego incorporarlos nuevamente cuando la demanda aumente.

FORMACIÓN DE RECURSOS HUMANOS

El trabajo central de la línea de I/D presentada se plantea en principio como una tesis de doctorado, aunque por la amplitud del tema, seguramente dejará abiertas muchas cuestiones que podrán ser tratadas en tesinas de grado, tesis de magister/doctorado o en trabajos de otros investigadores. Se pueden mencionar algunos de los temas de interés que pueden derivarse del proyecto:

– *Seguridad*: El proyecto no incluye la investigación en cuestiones relacionadas con la seguridad del DRVS.

– *Ahorro Energético*: Utilizar las facilidades del DRVS para reducir el consumo energético.

– *Mejoras en Disponibilidad*: El DRVS admite la replicación de componentes para incrementar la disponibilidad.

– *Mejoras de Rendimiento*: Muchos componentes podrán ser optimizados en su rendimiento derivándose en nuevos algoritmos, protocolos, mecanismos, etc.

– *Nuevos Device Drivers Virtuales y nuevos OS-Guest*: El prototipo del proyecto solo incluirá un número limitado de dispositivos virtuales y un único tipo de OS-Guest.

REFERENCIAS

[1] Galley S., "*PDP-10 Virtual Machines*". In Proc. ACM SIGARCH-SIGOPS Workshop on Virtual Computer Systems, Cambridge, MA, 1969.

[2] ScaleSMP; "*The Versatile SMP™ (vSMP) Architecture and vSMP Foundation™ Aggregation Platform*"; whitepaper.

[3] Chapman M., Heiser G.; "*vNUMA: Virtual Shared Memory Multiprocessors*"; Proceedings of the 2009 conference on USENIX.

[4] Popek G.; Goldberg R., "*Formal Requirements for Virtualizable Third Generation Architectures*". Communications of the ACM 17 (7): 412 –421. (1974).

[5] Dike J., "*A user-mode port of the Linux kernel*", USENIX Association, Atlanta, Oct 10 –14, 2000.

[6] Linux Vserver, http://linux-vserver.org/Paper

[7] OpenVZ Wiki, http://wiki.openvz.org/Main_Page

[8] LXC Linux Containers, http://lxc.sourceforge.net/

[9] Henning Kamp P.; Watson R., "*Jails: Confining the omnipotent root*", Proc. 2nd Intl. SANE Conference. 2000.

[10] Pessolani P.; Jara O., "*Minix over Linux: A User-Space Multiserver Operating System*", Computing System Engineering (SBESC), Florianópolis-Brazil, November 2011.

[11] Buyya, R; Cortes, T; Jin, H., "*Single System Image*", International Journal of HPC Applications 15 (2): 124,2001.

[12] Cheriton D.; "*The V Distributed System*"; Communications of the ACM 31 (3): 314–333; March 1988.

[13] Mullender S.; van Rossum G.; Tanenbaum A.; van Renesse R; van Staveren H., "*Amoeba: A Distributed Operating System for the 1990s*",Vrije Universiteit, May 1990.

[14] Bosse S., "*The VAMNET Book*", BSS Lab, 2005.

[15] Barak A.; La'adan O; Shiloh A., "*Scalable Cluster Computing with MOSIX for Linux*" Proc. 5th Annual Linux Expo Raleigh, NC, pp. 95–100, 1999.

[16] Barak A.; Shiloh A., "*The MOSIX Cluster Operating System for High-Performance Computing on Linux Clusters, Multi-Clusters and Clouds*", 2012.

[17] Maoz T.; Barak A.; Amar L., "*Combining Virtual Machine Migration with Process Migration for HPC on Multi-Clusters and Grids*", Proc. IEEE Cluster 2008, Tsukuba, 2008.

[18] Walker B., "*Open Single System Image (openSSI) Linux Cluster Project*", Hewlett-Packard.

[19] Morin, C.; Lottiaux, R.; Vallee, G.; Gallard, P.; Margery, D.; Berthou, J. Y.; Scherson, I.D., "*Kerrighed and data parallelism: cluster computing on single system image operating systems*", IEEE Computer Society, 2004.

[20] Wentzlaff D., Gruenwald III C., Beckmann N., Modzelewski K., Belay A., Youseff L., Miller J., Agarwal A.; "*An operating system for multicore and clouds: mechanisms and implementation*"; Proceedings of the 1st ACM Symposium on Cloud Computing 2010.

[21] Tanenbaum A., Woodhull A., "*Operating Systems Design and Implementation, Third Edition*", Prentice-Hall, 2006.

[22] Linnenbank N.; "*Implementing MINIX on the Single Chip Cloud Computer*"; Master Thesis, Vrije Universiteit, 2011.

[23] Pasin, M.; "*Trix, a transputer based multiprocessor operating system, with distributed process management*"; Master Thesis, Universidade Federal do Rio Grande do Sul, 1994.

[24] Liedtke J.; "*On µ-kernel construction*"; Symposium on Operating System Principles ACM, 1995.