Un Sistema de Virtualización Distribuida

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
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

Este trabajo refiere a los avances, logros alcanzados y planes de investigación futuros sobre el proyecto del Sistema de Virtualización de Recursos Distribuidos presentado en WICC 2012 [1].

En las tecnologías de virtualización actuales, la potencia de cómputo y el uso de recursos de las máquinas virtuales se limitan a la máquina física donde se ejecutan. Para alcanzar otros niveles de rendimiento y escalabilidad, las aplicaciones Cloud suelen estar particionadas en varias VMs o Contenedores ubicados en diferentes nodos de un cluster de virtualización. Los desarrolladores a menudo usan ese modelo de procesamiento porque sus aplicaciones no pueden hacer uso de los servicios de la *misma instancia* del Sistema Operativo (OS) en todos los nodos donde se ejecutan suscomponentes.

El sistema propuesto combina e integra tecnologías de Virtualización y Sistemas Distribuidos que puede proporcionar la *misma instancia* de un Sistema Operativo Virtual (VOS) en cada nodo del clúster. El resultado es un Sistema de Virtualización Distribuida (DVS) con los beneficios de ambos mundos, adecuado para ofrecer servicios Cloud de alto rendimiento y con posibilidades de ofrecer otras características que son requeridas por los proveedores de IaaS. Un DVS es capaz de ejecutar concurrentemente múltiples instancias de diferentes VOS, asignando un subconjunto de nodos para cada instancia, y a su vez compartiendo nodos entre ellas.

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 TIN2015-65316-P y del Gobierno Catalán por la ayuda 2014-SGR-1051.

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 área de virtualización realizadas en el contexto de PIDs de la UTN en la Facultad Regional Santa Fe y de los programas de postgrado de la Facultad de Informática de la UNLP.

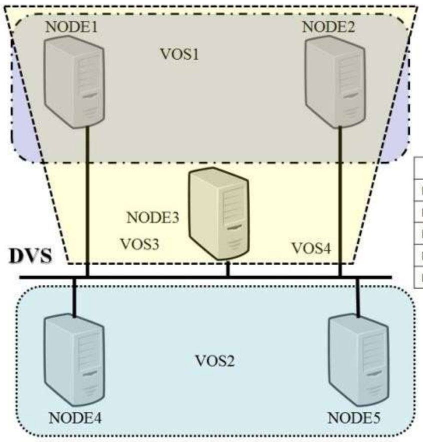
INTRODUCCION

La tecnología de virtualización desarrollada a fines de la década del '60 [2] 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 esta tecnología ofrece tales como: la capacidad para consolidar múltiples servidores virtuales en una misma computadora física y la implementación de entornos aislados para la ejecución de aplicaciones con mayor seguridad y protección.

En los diferentes sistemas de virtualización actuales una Máquina Virtual (abreviado VM en inglés), o en su caso un Contenedor están contenidos en un computador, por lo que su poder de cómputo se encuentra limitado por éste.

Por otro lado, los Sistemas Operativos Distribuidos de Imagen Única (abreviado SSI-DOS en inglés) ejecutan procesos en forma distribuida virtualizando recursos abstractos de software. Estos recursos son idénticos o de similares características a los que brinda un OS tradicional (centralizado) tales como archivos, tuberías (pipes), sockets, colas de mensajes, memoria compartida, semáforos o mutexes. Los usuarios y aplicaciones tienen la visión de un único computador virtual conformado por los recursos computacionales, procesos, recursos abstractos y servicios que se encuentran distribuidos en los distintos nodos que componen el cluster. Podría interpretarse que un SSI-DOS representa una forma de 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 único OS los recursos de múltiples computadores físicos.

El Sistema de Virtualización Distribuida propuesto combina e integra tecnologías de Virtualización y de SSI-DOS con los beneficios de ambos mundos. Por sus características se presenta como adecuado para brindar servicios Cloud de alto rendimiento. Eneste sentido, dispone de la posibilidad de ofrecer otras características atractivas para proveedores de servicios IaaS tales como alta disponibilidad, replicación, elasticidad, balanceo de carga y migración de procesos. Un DVS es capaz de ejecutar concurrentemente múltiples instancias de diferentes VOS, asignando un subconjunto de nodos para cada instancia, y a su vez compartiendo nodos entre ellas (Fig. 1).



	VOS1	VOS2	VOS3	VOS4
NODE1	X			X
NODE2	X		X	X
NODE3			X	X
NODE4		X		
NODE5		X		

Figura 1. Sistema de Virtualización Distribuido (DVS)

Bases Tecnológicas del Proyecto

El proyecto se nutre de las tecnologías de sistemas operativos tales como: 1) Multiservidor, 2) SSI-DOS y 3) Unikernel. En tanto que las siguientes tecnologías de virtualización aportan conceptos y estrategias al proyecto: 1) Paravirtualización, 2) Virtualización *basada en* Sistema Operativo, 3) Virtualización *de* Sistema Operativo. En particular, desarrollaremos un resumen sólo de esta última por cuestiones de espacio y porque la misma no suele ser considerada en las taxonomías de virtualización.

Virtualización de Sistema Operativo

Un OS es la capa de software que se encuentra entre las aplicaciones y el hardware. Gestiona los recursos de hardware y proporciona servicios (llamadas al sistema) para los programas y aplicaciones.

En la tecnología de virtualización de Sistema Operativo, el OS-guest no opera directamente sobre el hardware, sino que utiliza los servicios ofrecidos por el OS-host. Como en la tecnología de paravirtualización, el OS-guest debe ser modificado, no para utilizar servicios de un hipervisor sino los de un OS-host. User Mode Linux (UML) [3] y Minix Over Linux (MoL) [4] se basan en esta estrategia. UML permite que múltiples Linux-guest puedan ejecutarse como una aplicación en un sistema Linux-host. Como cada OS- guest es un proceso que se ejecuta en espacio de usuario, UML proporciona a los usuarios una forma de ejecutar múltiples instancias de Linux en un único computador sin requerir de privilegios de ***root***.

De manera similar, MoL es un sistema de Virtualización de OS desarrollado en UTN-FRSF que permite ejecutar múltiples instancias de Minix [5] sobre un Linux-host en forma aislada y segura. En MoL se emulan las APIs, System Calls y servicios de Minix. Manteniendo la arquitectura de Minix, MoL se compone de un conjunto de servidores, de tareas y de un micro-kernel (virtual) que se ejecutan enteramente en modo usuario dentro de un Contenedor. Sus procesos no tienen acceso a ningún recurso del Linux-hostni de otros procesos MoL, a excepción de aquellos que le fueron específicamenteasignados.

LÍNEAS DE INVESTIGACIÓN Y DESARROLLO

La línea de I/D de este proyecto refiere al desarrollo de un modelo de Sistema de Virtualización Distribuida como tecnología para brindar servicios del tipo IaaS. Para ello se requiere el conocimiento y dominio de tecnologías de Virtualización, de Sistemas Operativos Distribuidos de Imagen Única y de Sistemas Operativos Multiservidor.

Durante el tiempo transcurrido desde el inicio del proyecto (año 2012) se implementó un prototipo que se ejecuta en un cluster de computadores x86 y Linux como OS-host. Se desarrollaron dos VOS sencillos para ser ejecutados como *guests* sobre el prototipo como prueba de concepto. Uno de ellos es un VOS multiservidor y el otro es un unikernel, ambos en condiciones de brindar servicios de Internet (servidor web).

Sistema de Virtualización Distribuida

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 sí y dentro del kernel del sistema. En el modelo de DVS 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 Fig. 2):

- *Guest Virtual Operating System*: Es un VOS multiservidor constituido por un conjunto de procesos que pueden estar distribuidos en los distintos nodos de un cluster físico. Un VOS es una emulación de un OS adaptado para trabajar utilizando los servicios e interfaces que le ofrece el DVS [6]. Un VOS presenta a las aplicaciones las APIs y llamadas al sistema de un OS original.

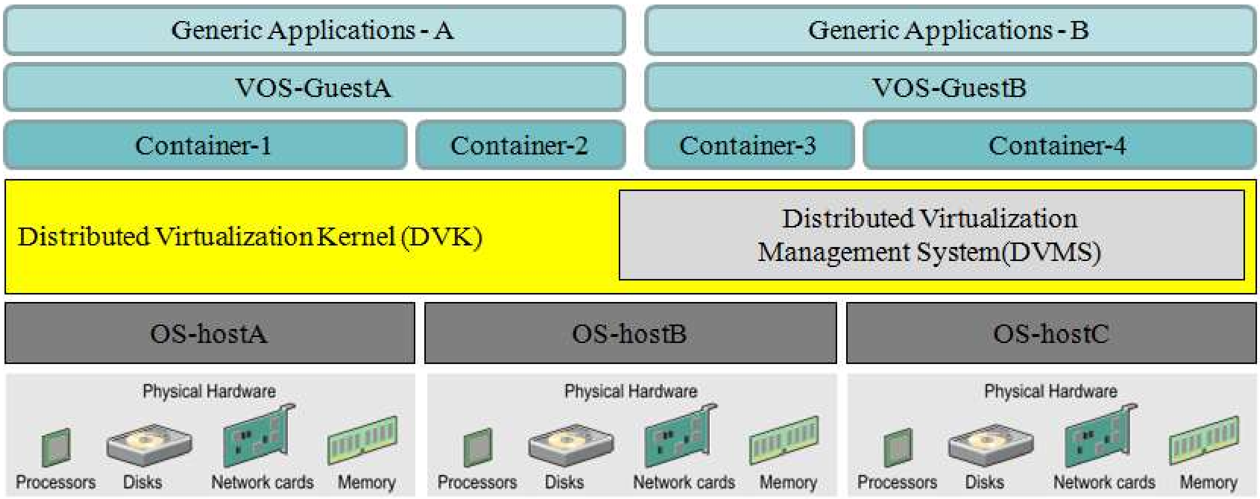


Figura 2. Arquitectura del DVS

- *Containers*: Es el entorno o subsistema en donde se ejecutan componentes de un VOS sobre un OS-hosts. Existe un Container por cada VOS en cada nodo del cluster.
- *Distributed Virtualization Kernel (DVK):* Es el software que integra todos los recursos del cluster, gestiona y limita los recursos asignados a cada VOS. Es responsable de brindar a los VOS las interfaces para los protocolos de bajo nivel tales como IPC, comunicaciones de grupo, sincronización, locking, elección de líder, detección de fallos, consenso, exclusión mutua, censado de parámetros de rendimiento, mecanismo de migración de procesos, servicios de clave-valor y virtualización de dispositivos.
- *Distributed Virtualization Management System (DVMS)*: Es el software que le permite al supra-administrador (admitiendo que hay un administrador por cada VOS) gestionar los recursos del cluster, asignar recursos a los VOS, realizar diversos tipos de monitoreo del sistema y tomar acciones sobre el DVS.

La unidad de asignación de recursos a un VOS no es el nodo sino los recursos virtuales que brinda el OS-host en cada nodo. Esta granularidad en la asignación de recursos permite una mejor explotación de los mismos brindando mayor elasticidad y facilitando 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 entre diferentes VOS y la *agregación* de conjunto de sus recursos distribuidos en múltiples nodos en un único VOS.

Comunicación entre Procesos (IPC)

Los OS multiservidor representan una base conceptual muy importante en el proyecto y en ellos son fundamentales los mecanismos de comunicación entre procesos (IPC). Por esta razón, se dedicó una línea de investigación y desarrollo al tratamiento de este crítico componente. En principio, se desarrolló un mecanismo de IPC embebido en el kernel de Linux que emulaba las primitivas de IPC de un OS-Minix al que se denominó M3-IPC. En su primera fase las comunicaciones se limitaban a un único host, aunque ya soportaba el aislamiento de comunicaciones entre procesos de diferentes VOS [7]. Esto significa que el mismo kernel de IPC permitía comunicar procesos de un mismo VOS, pero adicionalmente permitía ejecutar múltiples VOS en forma concurrente. Esta característica preserva el aislamiento requerido a cualquier sistema de virtualización.

El paso siguiente en el diseño y desarrollo fue llevar las comunicaciones a procesos en diferentes nodos en forma transparente para la aplicación. Para ello, se decidió realizar las comunicaciones inter-nodos mediante procesos *proxies* ejecutando en modo usuario. Si bien esto representa una reducción del rendimiento por el agregado de más cambios de contextos necesarios para las comunicaciones, tiene la ventaja de otorgar flexibilidad al momento de seleccionar el protocolo de comunicaciones entre nodos. Aún utilizando esta estrategia, el rendimiento de M3-IPC supera a RPC tanto en la transferencias de mensajes (11805 msg/s vs. 9900 msg/s) como en la de bloques de datos (58 Mbytes/s vs. 49 Mbytes/s) entre nodos conectados a un switch dedicado con interfaces de 1Gbps.

Durante el transcurso del proyecto se desarrollaron proxies utilizando protocolos TCP, UDP, TIPC, UDT y Raw Sockets demostrando la versatilidad que otorga el diseño. Por otro lado, éste también permitió incorporar de manera sencilla facilidades tales como la compresión de datos (LZ4), cifrado de datos (OpenSSL), o priorización de mensajes de acuerdo a criterios tales como el tipo de mensaje a transferir, el VOS origen/ destino o cualquier otro criterio, sin necesidad de modificar el kernel del sistema. La arquitectura permite utilizar proxies con protocolos diferentes para comunicar diferentes pares de nodos. Si bien son atractivas las ventajas que presentan los proxies en modo usuario, se dispone de proxies como módulos de kernel Linux utilizando protocolos TCP y TIPC (en desarrollo) para evaluar comparativamente el rendimiento.

Otra característica a destacar de M3-IPC es la transparencia en la localización, en la migración de procesos y en la replicación. Un proceso de un VOS no necesita conocer en que nodo se ejecuta el destinatario de su mensaje, solo necesita conocer su *endpoint*. De igual forma, si el supra-administrador o el DVMS deciden migrar un proceso de un nodo a otro, los otros procesos del VOS no necesitan realizar ninguna operación para adecuarse a ese cambio.

Para aplicaciones que requieren tolerancia a fallos se suelen utilizar esquemas de redundancia basados en *State Machine* [8] o en *Primary-Backup* [9]. M3-IPC soporta este último modelo asignándose a múltiples procesos de un mismo VOS localizados en diferentes nodos el mismo *endpoint*, aunque solo uno de ellos será el *endpoint* Primario o Activo. En caso de fallos, la aplicación podrá reasignar la función de Primario a uno de los procesos Backup. Todos estos cambios resultan transparentes a los otros procesos del VOS, particularmente a aquellos que utilizaban los servicios del servidor fallido.

Virtual Operating System

Con el desarrollo previo de MoL que utilizaba el IPC provisto por Linux, el paso siguiente fue migrar estos mecanismos a M3-IPC para conformar un VOS multiservidor. Se presentan a continuación algunos de los proyectos que resultaron de este trabajo.

Replicated Disk Server (RDISK)

Uno de los principales modelos de servicio de la computación en la nube es el de Infraestructura como Servicio (IaaS). En ella se ofrecen recursos computacionales y de almacenamiento a través de tecnologías de virtualización (ejemplo: SAN- Storage Area Network).

Si bien existen diversas tecnologías de virtualización de almacenamiento, el DVS ofrece transparencia en la ubicación, replicación y migración de procesos para mejorar la disponibilidad del servicio. Para ello se desarrolló RDISK [10], que es un servidor de almacenamiento que soporta dispositivos físicos, imágenes de discos en archivos o imágenes de discos en memoria o cualquier otro tipo de archivo que el OS-host pueda brindar, tal como un archivo remoto a través de NFS.

Para comunicarse con RDISK se requiere de un protocolo que debe utilizar M3-IPC para la transferencia de datos y mensajes. De todos modos, está planteado un proyecto para desarrollar un *wrapper* entre el protocolo NBD y M3-IPC para que su utilización pueda ser aprovechada por Linux sin necesidad de soportar M3-IPC.

RDISK puede utilizarse con o sin replicación. En éste último caso, para sacar provecho de las características de M3-IPC se debe utilizar el esquema *Primary-Backup*. Para la comunicación entre estos procesos replicados en diferentes nodos (los cuales tienen asignado el mismo *endpoint*) se utiliza un sistema de comunicaciones grupales denominado Spread Toolkit [11]. El proceso RDISK Primario realiza un multicast atómico de las actualizaciones que se realizan sobre el dispositivo a todas las réplicas de RDISK. La replicación de datos entre el Primario y las réplicas soporta la compresión de datos (utilizando LZ4) y se planea incorporar el soporte de encripción utilizando OpenSSL.

En caso de fallo del Primario, Spread notifica a los servidores Backup. Estos realizan una elección del nuevo Primario en base a la información entregada por Spread y asignan el *endpoint* Primario al servidor electo. Spread le permite a RDISK detectar fallos de procesos y de nodos, particiones/reconstitución de red y rearranque de nodos.

MoL Filesystem Server(MoL-FS)

Otros de los componentes del VOS multiservidor es el servidor de sistema de archivos MoL-FS [12] que, en principio es la migración del FS server de Minix. A este servidor que se tomó como base para el desarrollo se le incorporaron otras facilidades tales como la posibilidad de utilizar diferentes tipos de dispositivos donde gestionar el sistema de archivos. Los tipos de dispositivos previstos actualmente son: imagen de dispositivo en memoria RAM, imagen de dispositivo en archivo Linux (cualquier tipo de archivo, incluso accesible vía NFS), dispositivos RDISK, y está previsto desarrollar el soporte de dispositivos con protocolo NBD.

MoL-FS, utiliza M3-IPC para recibir peticiones por parte de las aplicaciones y para acceder a los servicios RDISK. A los fines de extender su uso a aplicaciones Linux estándares, se creó un módulo FUSE (Filesystem in USEerspace).

VOS Multiserver

Como se mencionó anteriormente, se tomó a MoL como base para el desarrollo de un VOS multiservidor. La arquitectura de este VOS está compuesta de los siguientes procesos que tienen su equivalencia con Minix 3:

- *Tarea del Sistema (SYSTASK):* Es responsable de realizar operaciones de bajo nivel tales como la creación/terminación de procesos, realizar la copia de datos entre los espacios de direcciones de diferentes procesos, gestionar privilegios, temporizadores y alarmas. Dispone de funcionalidades relacionadas a la ejecución en múltiples servidores tales como la migración de procesos, y la replicación.
- *Process Manager Server (PM):* Es responsable de ofrecer a las aplicaciones de usuario las APIs y System Calls POSIX. Gestiona además la jerarquía de procesos y sus PIDs.
- *Reincarnation Server (RS)*: Es responsable de someter a la ejecución procesos de tipo Tareas o Servidores. Los mismos pueden ponerse a ejecución en cualquier nodo del cluster donde esté previsto que el VOS ejecute, no solo en el nodo donde ejecuta el RS.
- *File System Server (FS):* Se utiliza MoL-FS.
- *Disk Task:* Se utiliza RDISK.
- *TTY Task:* Es responsable de gestionar la interacción entre el usuario y el VOS. Para ello utiliza dispositivos TTY virtuales (PTYs) que brinda el host Linux.
- *Ethernet Task (ETH):* Es responsable de gestionar las interfaces virtuales de red del VOS en uno de los nodos. Para ello utiliza dispositivos virtuales (TAP) que brinda el host Linux.

- *Internet Server (INET):* Es el servidor donde se implementan los protocolos de red TCP,UDP,IP, etc.
- *Information Server (IS):* Es el servidor que permite obtener información de los componentes del sistema.

Actualmente restan por terminar los trabajos de desarrollo y pruebas de TTY, ETH e INET.

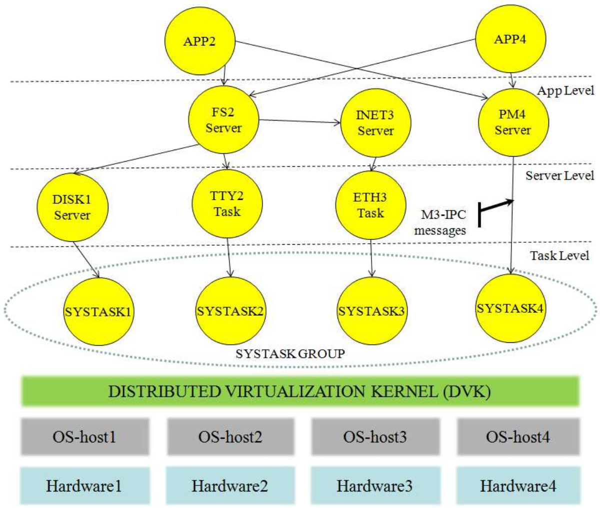


Figura 3. Arquitectura del VOS multiservidor ejecutando en DVS.

VOS Unikernel

Paralelamente al VOS multiservidor se desarrollaron tres variantes de VOS Unikernel [13] utilizando como base para el desarrollo a lwIP-tap [14]. En la Tabla I, se encuentran subrayados los componentes externos al Unikernel los que son accedidos utilizando M3-IPC.

TABLA I. VARIANTES DE VOS UNIKERNEL

Modelo	Web Server	TCP IP	Network Interface	File system	Storage
1	nweb	lwip	TAP	MoL-FS	Linux image file
2	nweb	lwip	TAP	FAT library	RDISK
3	nweb	lwip	ETH	FAT library	Linux image file

Queda demostrada la versatilidad que ofrece el modelo de arquitectura de virtualización propuesto para el desarrollo de diferentes VOS.

RESULTADOS Y OBJETIVOS

El objetivo primario del proyecto aquí presentado es desarrollar un modelo de DVS y acompañado de un prototipo básico como prueba de concepto. Se espera que con esta tecnología los VOS dispongan en forma transparente (para las aplicaciones y usuarios) del poder de cómputo y demás recursos de los nodos que los constituyen.

La tecnología DVS 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 individuales y de VOSs.

El DVS permite el particionado de los recursos de un cluster para conformar múltiples VOS-guests que podrán ser utilizados y administrados por diferentes comunidades de usuarios. Los nodos del cluster físico pueden ser asignados para uso exclusivo de un VOS en particular o compartir sus recursos entre múltiples VOS.

FORMACIÓN DE RECURSOS HUMANOS

El núcleo de la línea de I/D presentada se inició como una tesis de doctorado (aún no finalizada), pero dada la amplitud de la temática se desarrollaron varios trabajos en el ámbito de PIDs y de la cátedra de Virtualización y Sistemas Operativos Avanzados de la Facultad Regional Santa Fe de UTN. Se realizaron trabajos de fin de cátedra de varias promociones, tesinas de grado y proyectos relacionados a cargo de otros docentes-investigadores.

Se prevén para el futuro un número importantes de proyectos, algunos ya mencionados en otras secciones, a ser desarrollados por integrantes del PID o de la cátedra de acuerdo a la complejidad que los mismos representen.

REFERENCIAS

[1] Pablo Pessolani, Toni Cortes, Silvio Gonnet, Fernando G. Tinetti; "*Sistema de Virtualización con Recursos Distribuidos*". WICC 2012. Pág. 59-43. Argentina, 2012.

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

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

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

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

[6] D. Hall, D. Scherrer, J. Sventek, "*A Virtual Operating System*", Journal Communication of the ACM, 1980.

[7] Pablo Pessolani, Toni Cortes, Silvio Gonnet, Fernando G. Tinetti. "*Un mecanismo de IPC de microkernel embebido en el kernel de Linux*". WICC 2013. Argentina, 2013.

[8] Fred B. Schneider, "*Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial*", ACM Computing Surveys, 1990.

[9] Navin Budhiraja, Keith Marzullo, Fred B. Schneider, and Sam Toueg. "*The primary-backup approach*". In Distributed systems (2nd Ed.), Sape Mullender (Ed.). ACM Press/Addison-Wesley Publishing Co., NY, 199-216, 1993.

[10] Mariela Alemandi, Oscar Jara, "*Un driver de disco virtual tolerante a fallos*", Venado Tuerto, JIT 2015.

[11] The Spread Toolkit. http://www.spread.org.

[12] Diego Padula; Mariela Alemandi; Pablo Pessolani; Silvio Gonnet; Toni Cortes; Fernando Tinetti, "*A User-space Virtualization-aware Filesystem*" , Buenos Aires, CoNaIISI 2015.

[13] Anil Madhavapeddy and David J. Scott. "*Unikernels: Rise of the Virtual Library Operating System.*", Queue - Distributed Computing December 2013.

[14] https://github.com/takayuki/lwip-tap