Universidad Michoacana de San Nicolás de Hidalgo
División de Estudios de Posgrado
Facultad de Ingeniería Eléctrica

ALTA DISPONIBILIDAD EN MÁQUINAS VIRTUALES CON UN ESQUEMA DE REPLICACIÓN DE SERVICIOS TCP/IP

TESIS

Que para obtener el grado de
DOCTORA EN CIENCIAS EN INGENIERÍA ELÉCTRICA
OPCIÓN EN SISTEMAS COMPUTACIONALES

Presenta
VIOLETA MEDINA RIOS

Dr. Juan Manuel García García
Director de Tesis

Dr. J. Aurelio Medina Ríos
Co-Director de Tesis

Febrero 2013

Resumen

En varias áreas de la computación tales como en la virtualización, los sistemas de bases de datos o en la prestación de servicios de red, la replicación es considerada como un mecanismo para proporcionar alta disponibilidad a los sistemas. Estos sistemas deben permanecer activos la mayoría del tiempo, mostrando un tiempo de inactividad casi cero y una intervención humana mínima en caso de que se requiera la ejecución de un proceso de recuperación. La replicación permite que los datos consistentes se almacenen en más de un sitio de manera simultánea, proporcionando las condiciones para tener un sistema de alta disponibilidad. En esta tesis, se propone un esquema de replicación de servicios TCP/IP que pueda ser ejecutado tanto en redes de área local como en redes de área amplia. Este protocolo se implementa en máquinas virtuales manejada por Xen. Cada paquete que una máquina virtual llamada *original* recibe is replicado hacia una máquina virtual llamada *réplica*. El próposito principal de éste protocolo es preservar el mismo estado de la máquina original en la máquina réplica para proporcional alta disponibilidad. La máquina original puede ser sustituida por una réplica, adaptándose dinámicamente a las fallas. El protocolo selecciona una máquina de una lista de réplicas disponibles y la intercambia por la máquina original para proveer características de calidad en el servicio. No requiere modificaciones al protocolo TCP/IP o a los sistemas operativos anfitrión o huésped. Se analizan los resultados cuantitativos del comportamiento del protocolo en un escenario real con máquinas geográficamente distribuidas con especial atención en consistencia e integridad de datos y en rendimiento de la red. Los resultados obtenidos en replicación en ambiente WAN reportaron que aún existiendo dos réplicas activas, en comparación con una conversación sin réplicas se sufre una degradación de RTT del 16 %, lo cual no es perceptible para los usuarios en los sitios de trabajo. Se realizaron pruebas de replicación en un ambiente LAN, donde se configuró un esquema de tolerancia a fallas. Dicho esquema considera que puede suceder una falla tanto en la máquina original como en la réplica. En esta configuración, la replicación desde el punto de vista del cliente puede sufrir una degradación promedio de hasta el 87.92 % cuando hay tres réplicas activas, lo cual representa 0.050710 ms., lo cual no es un tiempo que llegue a interrumpir las actividades del usuario. Se reportaron tiempos de hasta 5.64 ms. para que una máquina réplica tome el lugar de una máquina original, lo cual tiene un impacto despreciable en las actividades del usuario.

Abstract

In many computation areas such as virtualization, database systems or network services, replication has been considered as a mechanism to provide high availability. Such systems should be active most of the time, with almost zero downtime, and minimal human intervention, if a recovery process is demanded. Replication allows consistent data stored in more than one site simultaneously providing conditions for a system high availability. In this thesis, a TCP/IP services replication scheme to provide a deployable mechanism in local and wide area networks is proposed. This protocol is implemented on virtual machines managed by Xen. Each package received by a virtual machine called *original* is replicated to a virtual machine called *replica*. The main purpose of this protocol is to preserve the same state of the original virtual machine in the replicated virtual machine for high availability. The original machine is substituted by a replica dynamically adapting to failures. Protocol selects a machine from a list of available replicas and switches it, providing the desired QoS features. It does not require modifications of TCP/IP protocol, host or guest operating systems. Quantitative results of protocol behavior in real scenarios with geographically distributed machines are analyzed with special attention in consistency, data integrity, and network performance. The results in a WAN environment reported that the average RTT with two active replicas suffers a 16 % degradation in comparison with the average RTT without replicas, with is imperceptible to users in the working sites. Replication tests were performed in a LAN environment, and a fault tolerant scheme was set up. Such scheme considers a fault in the original machine and in the replica machine. In this configuration, the replication from the point of view of the client suffers an average degradation up to 87.92 % when there are three active replicas, it represents 0.050710 ms., which is an imperceptible time for the user activities. Times up to 5.64 ms. were reported when a replica machine assumes the role of original machine. These times have a negligible impact in the user activities.

DEDICATORIA

A mi madre, por su apoyo infinito en todas las etapas de mi vida. A mi padre, que siempre esta conmigo. A mis hermanas Hortensia y Silvia por permanecer a mi lado en todas las circunstancias de la vida. A mis hermano Yoyo por su guía y fé aún en los momentos más díciles. A mis soles en la vida Mitzi y Fanny. A mis hermanos Toño y Raúl que están en mi corazón. Con mi amor eterno para Beto y Pepe.

Gracias al Dr. J. Aurelio Medina Ríos por el apoyo brindando a este proyecto durante toda la realización del mismo.

Gracias al Dr. Antonio Ramos Paz por sus consejos acertados, por su ayuda incondicional durante todo este trayecto. Gracias por brindarme su mano sincera en todo momento.

Gracias al Dr. Juan Manuel García García por darle dirección a este proyecto.

Gracias al Dr. Andrei Chernykh por el nuevo impulso dado a este proyecto. Mi admiración y respeto para usted, como profesionista y gran ser humano.

Al Dr. J. Antonio Camarena Ibarrola por sus consejos, observaciones y sugerencias aportadas, ha sido una experiencia gratificante trabajar con usted.

Agradezco al Consejo Nacional de la Ciencia y Tecnología el apoyo económico que como becaria recibí y que me permitió llegar a la culminación de este proyecto.

Una etapa de la vida se cierra y se abre otra, dando paso al nacimiento de una Luz Nueva.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

VM	Virtual Machine, Máquina Virtual
VMM	Virtual Machine Monitor, Monitor de Máquina Virtual
RM	Real Machine, Máquina Real
OVM	Original Virtual Machine, Máquina Virtual Original
RVM	Replica Virtual Machine, Máquina Virtual Réplica
PSC	Principal Server Connection, Conexión del Servidor Principal
SAN	Storage Area Network, Red de Área de Almacenamiento
NAS	Network Attached Storage, Almacenamiento Anexo a una Red
$VNIC$	Virtual Ethernet Network Card, Tarjeta Virtual Ethernet de Red
MAC	Media Access Control, Control de Acceso al Medio
ARP	Address Resolution Protocol, Protocolo de Resolución de Direcciones
$iSCSI$	Internet Small Computer System Interface, Interface de Sistema para Pequeñas Computadoras por Internet
NFS	Network File System, Sistema de Archivos de Red
WWS	Writable Working Set, Conjunto Escribible de Trabajo
ESX	Elastic Sky X
PTE	Page-Table Entries, Entradas Página-Tabla
CPU	Central Process Unit, Unidad Central de Procesamiento
TLB	Translation Lookaside Buffer, Búfer de Traducción Anticipada de Instrucciones
ABI	Application Binary Interface, Interface de Aplicación Binaria
ACK	Acknowledgement, Acuse de Recibo/Reconocimiento
$ADSL$	Asymmetric Digital Subscriber Line, Línea de Abonado Digital Asimétrica
DOS	Disk Operating System, Sistema Operativo de Disco

Lista de Publicaciones

Asociada a esta investigación doctoral, se ha logrado la siguiente producción:

Revistas indizadas por el JCR (aceptado con modificaciones):

> "A Survey of Migration Mechanisms of Virtual Machines"
> V. Medina y Juan Manuel García
> ACM-CSUR, ACM Computing Surveys. CSUR-2012-0101.R1

Revistas indizadas por el JCR (en revisión)

> "Adaptive TCP/IP Replication Scheme with Quality of Service for High Availability"
> V. Medina, Juan Manuel García y Andrei Tchernykh
> ELSEVIER Science. COMputer NETworks. COMNET-D-12-585.

Congresos Internacionales:

> "Live Replication of Virtual Machines"
> V. Medina. y J.M. García
> The 10th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS'11). International conferences at the University of Cambridge, UK in February 20-22, 2011.

> "Virtualization and Cloud Computing: their Trends in Scientific Computing"
> V. Medina y A. Ramos-Paz
> XIII Reunión de Otoño de Potencia, Electrónica Y Computación, ROPEC 2011, Morelia, Mich., 9-11 de Noviembre de 2011.

Congresos Nacionales:

"A Network Isolation Analysis in XEN"
V. Medina y J.M. García.
IX Jornada Nacional de Seguridad Informática. Asociación Colombiana de Ingenieros de Sistemas. Bogotá D.C. 17,17 y 19 de Junio de 2009

"TCP/IP Replication Schemes for High Availability in Data Grid"
V. Medina, J.M. García y A. Tchernykh.
1er Encuentro Nacional de Usuarios de Cómputo de Alto Desempeño (1er ENUHPC). Celaya, Guanajuato. 14-16 de Mayo, 2012

Capítulo 1

Introducción

En los últimos años, la alta disponibilidad de los sistemas ha sido tema de investigación y experimentación. Sin embargo, aún existen varios temas abiertos en este campo. Uno de los grandes retos es el desarrollo de mecanismos de alta disponibilidad para sistemas distribuidos. Los cuales tienen que asegurar un rendimiento operacional predefinido durante un periodo de medición contractual. El Protocolo de Control de Transmisión (en inglés, *Transmission Control Protocol*, TCP) [Forouzan10] complementándose con el Protocolo de Internet (en inglés, *Internet Protocol*, IP) [RFC81], es uno de los protocolos núcleo de la infraestructura distribuida pública compartida. Es utilizado por las aplicaciones de Internet más importantes y proporciona mecanismos confiables de intercambio de datos entre computadoras. Existen otras tecnologías que son ampliamente usadas sobre la infraestructura pública, tal es el caso de las redes privadas virtuales (en inglés, *Virtual Private Networks*, VPNs) [Davis01]. La virtualización es otra tecnología que ha resurgido con fuerza desde los 1960s y ha extendido su uso a los sistemas distribuidos sobre Internet.

Virtualización es una técnica de particionar o dividir los recursos de un solo servidor en múltiples ambientes de ejecución separados [Changarti07]. Cada uno de estos ambientes se ejecuta independientemente de los otros, lo cual hace posible que múltiples sistemas operativos se ejecuten en el mismo hardware; tales unidades lógicas son llamadas Máquinas Virtuales (en inglés, *Virtual Machines*, VMs). Cada ambiente de ejecución es llamado huésped (*guest*) y el servidor sobre el cual se ejecuta es llamado anfitrión (*host*). La

motivación básica para la virtualización es la misma para los sistemas operativos multitarea; las computadoras tienen más poder de procesamiento que el necesario para completar una sola tarea. Eventualmente, el hardware ha sido más rápido y una computadora puede completar una tarea y tener recursos no utilizados. La virtualización toma ventaja de los recursos no utilizados [Chisnall07]. En un sistema no virtualizado, un solo sistema operativo controla todos los recursos de hardware. Un sistema virtualizado incluye una capa de software, el Monitor de Máquina Virtual (en inglés, *Virtual Machine Monitor, VMM*) [Uhlig05]. El papel principal del VMM es arbitrar los accesos a los recursos de la plataforma física del servidor anfitrión de tal manera que los múltiples sistemas operativos (huéspedes del VMM) pueden compartir dichos recursos.

Desde finales de los 90s ha habido un renovado interés en los VMMs, no solamente en las áreas tradicionales de servidores, sino también como una extensión de los ambientes de computación de las computadoras de escritorio [Rosenblum04]. Por ejemplo, en 1999, VMWare introdujo su VMM en los sistemas x86 [VMWare]. El uso de los VMMs se ha extendido a diferentes áreas tales como la industria [Vaughan-Nichols06], la investigación [Smith05] y la educación [Hu08, William I. Bullers06]. Muchas organizaciones encontraron que tenían servidores haciendo una sola tarea o pequeños clusters con trabajos relacionados. La virtualización permite que un conjunto de servidores virtuales sean consolidados dentro de una máquina física única sin perder la seguridad de sus ambientes aislados. Rosemblum y Garfinkel [Rosenblum05] ofrecen una visión general de las características que los VMMs ofrece con el uso de las MVs. Dichas características son clonación, aislamiento, consolidación, migración y replicación de servidores. Pueden ser usadas para proporcionar alta disponibilidad en un sistema virtualizado. Clonación es el proceso de crear una nueva VM a partir de una VM existente. El clonado de VMs implica el copiado o duplicación de los archivos asociados a la VM [Lagar-Cavilla09]. Un VMM puede proveer clonación realizando copias exactas de una VM a un costo muy bajo. Por ejemplo, si un parche es probado, se puede clonar una máquina, aplicar el parche a dicha máquina y observar la manera en que el sistema es afectado. Este procedimiento es más fácil que probar en una máquina en producción. La capacidad de aislamiento [Gupta06] habilita la encapsulación de aplicaciones diferentes dentro de un ambiente de ejecución, lo cual si ocurre una falla de una

VM, ésta no afectará la ejecución de otras VMs hospedadas en el mismo hardware físico. Si una aplicación colapsa el sistema operativo debido a un error de programación, las otras aplicaciones están aisladas y pueden continuar ejecutándose sin problema. Además si una aplicación se ve comprometida, el ataque es contenido en la máquina virtual comprometida, sin afectar a las máquinas que pudieran estar compartiendo el mismo hardware físico. El aislamiento de rendimiento es una característica que garantiza que el consumo de recursos de una VM no degrade el rendimiento de otras VMs en el mismo hardware. La consolidación [Rosenblum05] de servidores permite mover aplicaciones que se ejecutaban en varios servidores físicos a MVs que se encuentran en uno o pocos servidores físicos; incrementando la eficiencia en uso y reduciendo el espacio y los costos de administración. La consolidación permite dar soporte a sistemas heredados [Rodríguez09] (*legacy system*), los cuales son sistemas informáticos que han quedado anticuados pero continúan siendo utilizados por el usuario (empresa); usualmente estos sistemas tienen una misión crítica y es díficil reemplazarlos, contienen mucha información acerca del proceso de negocio que soporta la operación de la empresa. Consolidar muchos servidores en máquinas virtuales, en un número reducido de hosts puede contribuir a reducir, de manera sustancial, el costo de la energía [Torres08]. El uso eficiente de la energía hace atractiva a la virtualización. Otra gran ventaja es la migración [Rosenblum05], la cual permite que una MV sea movida de una máquina física a otra según las necesidades de un centro de datos. Una VM puede ser movida a otro host si el hardware comienza a experimentar fallas o si se programa alguna actualización. La VM migrada puede regresar a su máquina original cuando éste comience a trabajar otra vez. Varios VMMs incluyen un mecanismo de migración para proporcionar tolerancia a fallas en un sistema de alta disponibilidad. La migración es una herramienta para balanceo de carga, que trata problemas como fallas de hardware, escalamiento de sistemas o reasignación de recursos. Un mecanismo de migración debería proporcionar un tiempo de inactividad casi cero. Un mecanismo de migración debe ser transparente para el sistema operativo huésped y clientes remotos de la máquina virtual. Debe de aparentar a todas las partes implicadas que la VM no cambia de lugar. En esta tesis se ha considerado que un mecanismo de migración mueve una VM desde un host a otro y que solo se realiza una copia a la vez. Un mecanismo de replicación mantiene múltiples copias de la imagen de la VM original en

sincronía. Esto es típicamente usado para crear un esquema de recuperación de fallas para un sistema de alta disponibilidad. En esta tesis se presenta un protocolo de replicación de servicios TCP. Este protocolo hace uso de las dos tecnologías mencionadas anteriormente: el diseño e implementación de un protocolo de comunicación TCP/IP sobre un ambiente virtualizado.

1.1. Planteamiento del Problema

Se plantea el uso de dos tecnologías; el Protocolo de Control de Transmisión y la virtualización con el empleo del Monitor de Máquinas Virtuales Xen, para implementar un esquema de Replicación TCP/IP que permita preservar el mismo estado de una máquina original en una máquina virtual replicada para alta disponibilidad.

El Protocolo de Control de Transmisión complementando el Protocolo de Internet es uno de los protocolos más usados para mantener la infraestructura distribuida pública. Es usado en la mayoría de las aplicaciones de Internet y protoporciona un mecanismo confiable de intercambio de datos entre computadoras.

La tecnología de Virtualización habilita a una sola computadora a ejecutar múltiples sistemas operativos simultáneamente. De esta forma, las compañías pueden usar un solo servidor para tareas múltiples que normalmente tendrían que ejecutarse en múltiples servidores, los cuales pueden usar sistemas operativos diferentes. Por lo anterior, esta tecnología reduce el número de servidores que son empleados en las tareas diarias, permitiendo reducir costos y utilizar los recursos de manera eficaz.

Una de las características de utilidad que ofrecen los monitores de máquinas virtuales modernos es la migración de máquinas virtuales. La migración en "vivo"de las Máquinas Virtuales es la capacidad que tiene al Monitor de Máquinas Virtuales de copiar el estado de ejecución de un sistema operativo huésped desde un servidor físico a otro servidor físico mientras la VM se encuentra en ejecución. La migración es una herramienta que permite proporcionar tolerancia a fallas, balanceo de cargas de trabajo y facilidad de programar el mantenimiento de un sistema en el momento requerido.

Se plantea la replicación como una técnica que permite transladar el estado de una

máquina virtual origen (incluyendo memoria, conexiones de red y almacenamiento local) hacia una o más máquinas virtuales, manteniendo en sincronía cada una de estas replicas. La replicación permite crear un sistema de alta disponibilidad [Chidambaram08], es decir, un sistema que garantiza la prestación de sus servicios a pesar de las fallas de sistema que pudieran presentarse.

El objetivo de la presente tesis doctoral es diseñar e implementar un protocolo de replicación de servicios TCP/IP en vivo, lo cual permita mantener un esquema de alta disponibilidad en un ambiente conformado por máquinas virtuales, es decir, si se produce una falla, el sistema debe tener la capacidad seguir proporcionando los servicios TCP/IP que presta. El protocolo de replicación realizará la transferencia de los recursos de memoria, las conexiones de red activas y el almacenamiento de disco duro, a través de un esquema de seguridad que permita proteger los recursos mencionados de un ataque informático. La replicación de los servicios TCP/IP no deberá degradar la ejecución de los servicios activos. Las replicas deberán ser generadas conservando la integridad de los datos originales.

1.2. Revisión del Estado del Arte

En ésta tesis, se presenta un protocolo de replicación de servicios TCP/IP donde se hace uso de dos tecnologías; la virtualización y el manejo del protocolo TCP/IP. A continuación se presenta una revisión del Estado del Arte de las dos tecnologías mencionadas.

1.2.1. La primera máquina virtual

La primera máquina virtual que soportó virtualización completamente fue VM de IBM, la cual comenzó su vida como parte del proyecto Sistema/360. La idea de Sistema/360 (a veces abreviado S/360) fue proporcionar una arquitectura estable y actualización de parches para clientes IBM. Se produjo una variedad de máquinas con la misma arquitectura básica, las compañías pequeñas podían comprar una minicomputadora si era lo que requerían y actualizarla después a una gran mainframe con el mismo software [Chisnall07].

Una oportunidad de mercado identificada por IBM en esa época fue que la gente deseaba consolidar máquinas Sistema/360. Una compañía con pocas minicomputadoras

Sistema/360 podía ahorrar dinero actualizando a una sola S/360, asumiendo que una sola minicomputadora podía proporcionar las mismas características que el conjunto de minicomputadoras. El modelo 67 introdujo la idea del conjunto de instrucciones de auto-virtualización.

El Modelo 67 podía ser particionado fácilmente y parecía tener una serie de versiones de si mismo (menos poderosas). Incluso podía ser virtualizado recursivamente; cada máquina virtual podía ser particionada. Esto hizo muy fácil migrar de tener una colección de minicomputadoras a tener una sola mainframe. Cada minicomputadora sería reemplazada por una máquina virtual, la cual sería administrada exactamente en la misma forma, desde la perspectiva de software. La última versión de VM fue z/VM, la cual se ejecutaba sobre las máquinas zSeries de IBM (llamada System z). Estas podían ejecutar varios sistemas operativos, que iban desde sistemas anteriores para aplicaciones heredadas hasta sistemas nuevos como Linux y AIX en un ambiente completamente virtualizado. También podía ejecutar aplicaciones nativas VM/CMS.

1.2.2. La arquitectura x86

El CPU 80386 fue diseñado con la idea de virtualización en mente. Una de las metas de diseño fue permitir la ejecución de aplicaciones múltiples de DOS a la vez. En esa época, DOS era un sistema operativo de 16 bits ejecutando aplicaciones de 16 bits sobre un CPU de 16 bits. El 80386 incluía un modo virtual 8086, el cual permitía un sistema operativo con un ambiente 8086 aislado para programas anteriores, incluyendo el antiguo modelo de direccionamiento de modo-real ejecutándose sobre el direccionamiento de modo protegido.

Debido a que no existían aplicaciones IA32 y se esperaba que los sistemas operativos del futuro soportaran de manera nativa la multitarea, no fue necesario agregar un modo virtual 80386.

Incluso sin este modo, el procesador sería virtualizable, si de acuerdo a Popek y Goldberg [Popek73], el conjunto de instrucciones sensitivas es un subconjunto del conjunto de instrucciones privilegiadas. Esto significa que cualquier instrucción que modifica la configuración de recursos en el sistema debe ser ejecutada en modo privilegiado o atrapada

si no lo es. Desafortunadamente, hay un conjunto de 17 instrucciones en el conjunto de instrucciones de x86 que no tienen esta propiedad.

Aunque x86 es difícil de virtualizar, es también un objetivo muy atractivo, pues se encuentra muy difundido. Por ejemplo, virtualizar Alpha [Enterprises11] arquitectura de 64 bits perteneciente a las Computadoras con Conjunto de Instrucciones Reducidas (*Reduced Instruction Set Computer*, RISC) diseñada por *Digital Equipment Corporation* (DEC), es mucho más fácil, sin embargo, la base instalada de CPUs Alpha es insignificante comparada con x86, dando un potencial de mercado más pequeño.

1.2.3. Estrategias de Virtualización

Hay tres principales metodologías usadas para proporcionar virtualización [Changarti07, Chaudhary08]:

Emulación del sistema: El ambiente de ejecución es llamado máquina virtual y emula todos los recursos de hardware. Esta capa de emulación usa los recursos de hardware real del servidor anfitrión, lo cual hace posible que el emulador ejecute un sistema operativo huésped sin modificaciones. El VMM atrapa y traduce/ejecuta las instrucciones de CPU privilegiadas que están disponibles en el espacio de usuario. Debido a que atrapar las instrucciones privilegiadas puede traer penalizaciones significativas en rendimiento, se han creado estrategias para mejorar dicho rendimiento, tal es el caso de las traducciones binarias. La virtualización completa es usada por productos tales como VMWare, Microsoft Virtual PC y Parallels.

Paravirtualización: No hay emulación de hardware. El sistema operativo que se ejecuta sobre un huésped necesita ser una versión modificada, lo cual permite que la VM se coordine con el hipervisor. Esto reduce el número de instrucciones privilegiadas de CPU que se ejecutan, y como no hay emulación de hardware involucrada, el rendimiento es mucho mejor y cercano a las velocidades nativas. Esto es una técnica usada por Xen y Linux Modo de Usuario [Dike01].

Virtualización a Nivel de Sistema Operativo: Cada instancia huésped es aislada y se ejecuta en un ambiente seguro. Sin embargo, solamente es posible ejecutar múltiples instancias de huéspedes del mismo sistema operativo que el anfitrión. Si el sis-

tema operativo anfitrión es FreeBSD, solamente se pueden ejecutar múltiples instancias de FreeBSD. La ventaja principal de la virtualización a nivel de sistema operativo es su alto rendimiento. No es necesario un hipervisor o atrapar instrucciones de CPU. Su desventaja mayor es que todas las instancias comparten el mismo kernel, de tal forma que si el kernel se colapsa o se ve comprometido, todas las instancias se ven comprometidas. Este es un método usado por FreeBSD jails [Fre] y zonas Solaris10 [Oracle09].

La emulación de hardware introduce algo de costo debido a las traducciones de los requerimientos entre el hardware y el software. Las soluciones paravirtualizadas logran conseguir mayor velocidad eliminando los pasos de emulación. La virtualización a nivel de sistema operativo probablemente tiene el costo mínimo entre los tres métodos mencionados y es la solución más rápida, pero elimina uno de los grandes beneficios de la virtualización, que es proporcionar a los usuarios la capacidad de ejecutar diferentes sistemas operativos.

1.3. Objetivos de la Tesis

1.3.1. Objetivo general

Diseñar, implementar y estudiar un Protocolo de Replicación de Servicios TCP/IP, que permita conservar un esquema de alta disponibilidad en un entorno de máquinas virtuales localizadas tanto en una Red de Área Local como en una Red de Área Extendida.

El protocolo de replicación mantendrá el mismo estado completo del sistema original hacia una o más replicas donde los datos consistentes serán consistentes.

Se utilizarán mecanismos de transferencia seguros y eficientes que permitan mantener el aislamiento de las máquinas virtuales y un desempeño eficiente del sistema.

1.3.2. Objetivos particulares

- Diseñar e implementar un protocolo de replicación de recursos de memoria eficiente y seguro, que permita la recuperación de fallas en un tiempo satisfactorio de inactividad de la máquina virtual origen.
- Diseño e implementación de un protocolo de replicación de conexiones de red activas entre dos máquinas virtuales localizadas dentro de una red de Área o Local ó a través de una Red de Área Extendida.
- Diseño e implementación de un protocolo que permite la replicación de recursos de almacenamiento local, que no cause contención de recursos, tal como el ancho de banda y que se ejecute en un tiempo de replicación total eficiente.

1.4. Metodología

La metodología asociada con el desarrollo de ésta Tesis se basa en la ejecución de las siguientes actividades:

1. Revisión de las tecnología de virtualización y protocolo TCP/IP. En esta etapa se realiza una revisión a detalle de las mecanismos existentes de migración de VMs y de servicios TCP/IP y sus requerimientos de implementación.

2. Diseño, implementación y operación de un protocolo de replicación de servicios TCP/IP. En ésta etapa se procede a crear el diseño e implementación del algoritmo que permita realizar la copia de los servicios activos de una MV origen hacia una o más MVs conservando la integridad de la información y la alta disponibilidad del sistema. Este esquema considera el diseño el implementación de un mecanismo de tolerancia a fallas, tanto de la MV original como las réplicas.

3. Puesta en operación del protocolo de replicación de servicios TCP/IP. Una vez diseñado e implementado el protocolo se replicación se procederá a verificar su adecuada operación en diferentes ambientes. Tanto en un ambiente de red de área local, como en un ambiente de red de área amplia.

4. Verificar la consistencia de los datos replicados y la disponibilidad de los mismos mediante simulación de fallas.

1.5. Justificación

En base a lo expuesto en la revisión del estado del arte, durante las última décadas la investigación en el área de migración de máquinas virtuales y el desarrollo de aplicaciones sobre protocolo TCP/IP ha tenido gran un auge.

La migración en vivo permite que MVs puedan ser movidas entre servidores físicos en respuesta a cargas de trabajo cambiantes, así los centros de datos pueden tener una infraestructura dinámica, asignando estos recursos sobre demanda o según la utilización de los recursos. Actualmente, la computación ya no se realiza dentro de un centro de datos, ahora los recursos pueden estar dispersos a través de múltiples centros de datos alrededor del mundo. El desarrollo de mecanismos que permitan la migración y/o replicación de recursos computacionales en vivo a través de una WAN es un área de investigación abierta. Esta clase de transferencia de recursos enfrenta retos como la transferencia de imágenes de disco y de memoria a través de un ancho de banda bajo y minimizar la degradación de rendimiento del sistema de la MV en migración. Actualmente, la virtualización y la computación en nube son dos tópicos ligados [Armbrust09, Armbrust10]. Usar virtualización para crear

una red privada es una tendencia importante. Últimamente, una dirección de investigación importante es desarrollar nuevas tecnologías que habilitan la computación en nube para soportar grandes aplicaciones distribuidas a pesar de las limitaciones de ancho de banda y latencia [Sripanidkulchai10]. En la actualidad, cuando los proveedores de cómputo en red necesitan ejecutar algún arreglo o mantenimiento en un servidor físico, estos informan a los usuarios que deben parar y reubicar sus instancias a una ubicación diferente. Los proveedores de computación en nube podrían usar migración en vivo para mover algunas instancias mientras se encuentran en ejecución. Aunque algunos proveedores exponen una interface a los usuarios para usar migración en vivo dentro de su propia nube, la migración en vivo hacia una nube diferente es un área de investigación abierta [Williams12]. Podemos apreciar que la reubicación de recursos es un necesidad de las tecnologías actuales que manejan datos distribuidos. Tomando en consideración los argumentos recién expuestos, en la presente tesis se propone un protocolo de replicación TCP/IP que permita replicar los servicios TCP/IP tanto en una área local como en una red de área extendida, con el objeto de mantener la alta disponibilidad de los recursos y que ofrezca tolerancia a fallas del sistema.

En la presente tesis se reportan resultados que verifican la confiabilidad de los datos replicados en diferentes sitios y la eficiencia del protocolo de replicación. Además se propone un esquema de recuperación de fallas de una manera automática. Por último, se busca que la implementación realizada sea útil con fines académicos y de investigación.

1.6. Aportaciones

En esta tesis, se propone un protocolo de replicación de servicios TCP/IP y se realizan las siguientes aportaciones:

- El protocolo mantiene actualizadas varias replicas de conversaciones TCP/IP de manera simultánea.

- La implementación de un mecanismo que permite reemplazar una máquina original por una máquina réplica, si la máquina original falla.

- Se proporciona la posibilidad de manejar múltiples réplicas de servicios TCP/IP en máquinas virtuales que se encuentran geograficamente separadas.

- El protocolo puede ser ejecutado sin necesidad de realizar modificaciones al núcleo de ningún sistema operativo, ni anfitrión, ni huésped.

- El protocolo puede ser instalado sobre máquinas virtuales o en máquinas reales sin necesidad de modificaciones.

- Para el funcionamiento del protocolo no es necesario hacer una configuración adicional de almacenamiento compartido.

1.7. Descripción de Capítulos

El resto de la tesis esta organizado de la siguiente forma:

En el **Capítulo 2** se presentará una comparación de las ventajas y desventajas que proporcionan los mecanismos de migración y de replicación en máquinas virtuales. Además, se realizará una revisión detallada de diversos mecanismos de migración basados en TCP/IP y de diferentes mecanismos de migración de máquinas virtuales, así como como su clasificación dentro de mecanismos de migración en vivo, memoria, suspender/reasumir, migración a través de WAN, balanceo de carga o grabar/reproducir.

En el **Capítulo 3** se presentarán las características que posee el Monitor de Máquina Virtual elegido *(Xen)* como base del entorno virtualizado en el diseño e implementación del protocolo de replicación.

En el **Capítulo 4** se realizará una descripción detallada del protocolo de replicación de servicios TCP/IP propuesto. Se presentará un esquema general del protocolo y el detalle de implementación del protocolo propuesto.

El **Capítulo 5** se presenta la descripción de cada unos de los escenarios de prueba para los casos de estudio presentados y los resultados obtenidos en cada uno.

En el **Capítulo 6** contiene las conclusiones obtenidas del diseño e implementación del Protocolo de Replicación de servicios TCP/IP y se aportan, en base a la experiencia

recabada por la autora de la investigación doctoral realizada, sugerencias de trabajo de investigación futura en el mismo campo del conocimiento.

Capítulo 2

Introducción a Diferentes Mecanismos de Migración en TCP/IP y en Máquinas Virtuales

2.1. Migración vs. Replicación

Las diferentes estrategias tanto de migración como de replicación mencionadas anteriormente tienen tanto ventajas como desventajas. A continuación se mencionan algunas ventajas y desventajas de ambos mecanismos

2.1.1. Migración

Para el mecanismo de migración se pueden mencionar las ventajas siguientes:

- Los esquemas de migración de memoria han alcanzado un grado de sofisticación y eficiencia suficiente para hacer casi imperceptible al usuario el tiempo de inactividad que puede propiciar el traslado de la información de la máquina fuente hacia la máquina destino.

- El empleo de tecnologías da almacenamiento distribuido como *NAS o SAN* o el empleo de sistemas de archivos distribuidos, permiten que no sea necesario considerar la

migración de almacenamiento local en las máquinas virtuales, ya que, si la migración se realiza en una red local, el sistema de archivos sigue siendo accesible desde cualquier nodo que pertenezca a la red.

La migración de MV puede presentar las desventajas:

- La migración debe ser activada de manera explícita, de tal forma que si la máquina original sufre cambios después del punto en que se realizó la migración, estos cambios no se reflejan en la máquina virtual a donde se migró la información.
- El tiempo total de migración puede ser muy alto si se considera la migración de almacenamiento de disco local de máquinas virtuales.

2.1.2. Replicación

Repecto a la replicación, se pueden mencionar las ventajas

- Las modificaciones que se realizan en la MV original son reflejadas casi de manera inmediata en la máquina destino, sin necesidad de realizar una suspensión de la MV de manera explícita, de tal forma que si existe una falla en el sistema, no habría pérdida de información.
- La réplica de una máquina virtual puede realizarse hacia más de una máquina virtual destino de manera simultánea.
- Las modificaciones al almacenamiento local de las máquinas virtuales se realizan como parte de la replicación, por lo cual el tiempo de replicación no dependerá del tamaño total de los archivos almacenados.

Se pueden mencionar las desventajas:

- Si el número de réplicas crece demasiado, se puede ocasionar una congestión en el tráfico de la red, debido al envio y recepción de varias copias de la misma información a través de la red.
- Creación de complicados protocolos de sincronización de réplicas para mantener copias equivalentes a la original.

2.2. Migración de servicios TCP

La migración y replicación de recursos se ha hecho presente en varías tecnologías tales como en los servicios TCP/IP [Aghdaie09, Shenoy00], bases de datos [Davies06] y en la virtualización [Clark05]. En las siguientes secciones se realiza una revisión del Estado del Arte de los mecanismos de migración (y algunos de replicación) de las áreas relacionadas con la presente Tesis.

Existen varias publicaciones de mecanismos de migración de servicios basados en el protocolo TCP/IP [Shao08, Koch03, Zagorodnov09, Aghdaie09]. A continuación se describen algunos de los más notables.

2.2.1. ER-TCP

En [Shao08], se propone un esquema de tolerancia a fallos para un clúster computacional. Dicho clúster es configurado dentro de una Red de Área Local (*Local Area Network*, LAN), donde los nodos pertenecen a una subred privada. En el clúster, hay un servidor primario, un servidor único de registro y uno o varios servidores de respaldo. Cada nodo contiene un módulo Manejador de Conexión (*Connection Management*) y uno de Control de Respuesta (*Response Control*). Los servidores primario y de registro contienen un módulo de Manejo de Buffer (*Buffer Management*, BM). Para sincronizar nodos, ER-TCP usa un túnel multicast. Durante una conversación TCP, todos los servidores generan paquetes de respuesta. Se asume que todos los paquetes son iguales. El servidor primario espera por la respuesta del servidor de registro y responde a los clientes. Una de las funciones del servidor de registro es mantener la sincronización de las conexiones TCP con los servidores de respaldo, reuniendo las respuestas durante las fases de conexión y terminación. Para conexiones de larga duración, el servidor de registro espera las respuestas de los respaldos cada periodo de tiempo predefinido durante una iteración requerimiento-respuesta denotada por K. El servidor primario y el de registro se encuentran sincronizados entre sí. El servidor de registro se sincroniza con el primario a cada momento, mientras que los servidores de respaldo son sincronizados al principio y al final de cada conversación o bien, cada K iteraciones. Si el servidor primario no espera las respuestas de los servidores de respaldo, no hay overhead. El

servidor primario y el de registro contienen BMs, los cuales registran los requerimientos no confirmados del servidor de respaldo; esto puede llevar a un alto consumo de memoria. Si el servidor primario falla, el servidor de registro se convierte en servidor primario. Entonces, el servidor de respaldo más rápido asume la función del servidor de registro y el buffer de registro del servidor primario se copia del servidor primario al nuevo servidor de registro. Si un servidor de respaldo colapsa, todos los servidores restantes conocen el evento, destruyen la información relacionada con el servidor en falla y sus conexiones. Se asume que los servidores primario y de registro no fallan simultáneamente. Para cargas ligeras, ER-TCP causa una degradación del rendimiento del 15 %.

2.2.2. Conexión TCP tolerante a fallas de manera transparente

En [Koch03], se presenta una conexión TCP tolerante a fallas de manera transparente. La tolerancia a fallas se consigue modificando la pila TCP/IP del servidor. No se necesitan modificaciones a la pila TCP/IP del cliente, la aplicación cliente o al servidor de aplicación. Se implementa una replicación activa, así los servidores primarios y secundarios generan peticiones y respuestas. Existe un servidor primario P y un servidor secundario S, todos se comunican a través de una conexión tolerante a fallas. El esquema de replicación se puede describir con los siguientes pasos:

1. El servidor secundario recibe todos los datagramas de los clientes. Un puente en el servidor secundario descarta todos los datagramas que no contienen el segmento TCP o que no se dirigen hacia P.

2. Para los datagramas interceptados, el puente reemplaza el campo de destino fuente con la dirección del servidor secundario (a_S) y pasa el datagrama a la capa TCP. TCP asume que un cliente (C) envía este segmento directamente a S. La capa TCP extrae el requerimiento original del cliente y lo pasa al servidor de aplicación. Después de que el servidor de aplicación secundario ha procesado el requerimiento del cliente se genera una respuesta.

3. Si el puente del servidor secundario recibe un segmento que tiene como dirección IP a C, se reemplaza el campo de dirección IP destino del segmento con la dirección de P.

En consecuencia, todos los segmentos TCP dirigidos hacia el cliente son direccionados hacia P.

4. Después de que el servidor ha procesado el requerimiento del cliente, se crea una respuesta para el cliente. La carga de datos (*payload*) del segmento, se forma en una cola de salida. Esta información no se envía inmediatamente al cliente. Estos datos son enviados al cliente hasta que el puente del servidor primario recibe el segmento de datos que corresponde con la información guardada en la cola del servidor primario.

5. El nuevo segmento transporta la dirección del servidor primario P en el campo fuente y la dirección del cliente C en el campo destino. El campo de reconocimiento contiene el número de secuencia de reconocimiento del último segmento que el puente ha recibido de P ó S, el que sea menor. Eligiendo el menor de los dos reconocimientos se garantiza que ambos servidores han recibido exitosamente datos de los clientes de todos hasta el número de secuencia del reconocimiento enviado.

Se detectan fallas en los servidores primario o secundario. Cuando se detecta una falla en el servidor primario, el servidor TCP secundario para de enviar segmentos TCP a la capa IP y deshabilita la traducción de direcciones a_P-a-a_S (Primario a Secundario)para paquetes TCP. El servidor secundario asume la dirección IP del servidor primario. Después de esto, el puente comienza a trabajar otra vez. Si el servidor secundario falla, el servidor primario elimina todos los datos de la cola de salida del servidor primario y envía los paquetes correspondientes al cliente y deshabilita la demultiplexación de los datagramas IP de entrada.

2.2.3. FT-TCP

En [Zagorodnov09], se describe un protocolo TCP tolerante a fallas (FT-TCP). Este mecanismo implementa dos tipos de respaldos: el primero es respaldo caliente, donde cada replica ejecuta el procesamiento de los requerimientos de los clientes; cuando todas las replicas han completado el procesamiento, una de ellas (la primaria) responde al cliente. El segundo tipo es llamado respaldo en frío, solo una replica está procesando requerimientos y todos los requerimientos del cliente son guardados en un archivo de registro. FT-TCP se

compone principalmente de tres partes, envoltorio del lado sur, envoltorio del lado norte y un buffer estable. FT-TCP usa un componente llamado envoltorio del lado sur (*south-side wrapper*, SSW) para interceptar, modificar y descartar paquetes entre las pilas TCP/IP y los controladores de red. FT-TCP intercepta y cambia semánticas de las llamadas al sistema (entre una aplicación y el kernel) hechas por el servidor de aplicación a través de un componente llamado envoltorio del lado norte (*north-side wrapper*, NSW). Ambos componentes se comunican con un buffer estable localizado en la memoria física de las máquinas de respaldo. El buffer reconoce los datos recibidos y los regresa al solicitante de manera ordenada. FT-TCP es diseñado para soportar la falla de un servidor primario y de cualquier número de réplicas, si hay al menos un servidor de respaldo trabajando correctamente. El respaldo detecta la falla del primario, si no hay comunicación con el primario por un intervalo de tiempo. En el esquema de respaldo en caliente, después de una falla del servidor primario, el servidor de respaldo alcanza el estado del servidor primario antes de que ocurra la falla. Este proceso usa los registros de segmentos TCP y llamadas al sistema del buffer estable. El proceso de tolerancia a fallas termina cuando el servidor de respaldo toma el lugar del servidor primario. La tolerancia a fallas con respaldos en frio consume memoria por los paquetes almacenados y llamadas al sistema. En el esquema de respaldo en frio, las actividades ejecutadas por la máquina de respaldo son registradas y monitoreadas por la replica primaria. El proceso de recuperación desde un respaldo en frío puede ser manejado por un mecanismo checkpoint. Cuando ocurre una recuperación, el FT-TCP conserva vivas las conexiones de los clientes, respondiendo a los paquetes de entrada con paquetes ACK con una ventana de tamaño cero. Lo que indica que el servidor está trabajando, pero los clientes no pueden enviar datos.

2.2.4. CoRAL

CoRAL fue presentado en [Aghdaie01], [Aghdaie03] como un esquema de servicio web tolerante a fallas basado en replicación de conexión y registro a nivel aplicación (en inglés, *Connection Replication and Application-level Logging*, CoRAL). La idea básica de CoRAL es usar una combinación de replicación activa y de registro [Aghdaie09]. Hay un servidor primario y uno de respaldo. Sin embargo, el servidor de respaldo registra los

requerimientos y respuestas http, pero no los procesa, a menos que el servidor primario falle. Los clientes establecen comunicación con el servicio por una sola dirección de servidor, compuesta de una dirección IP y de un número de puerto TCP, denominada dirección anunciada. A nivel TCP/IP, todos los mensajes enviados por los clientes tienen la dirección anunciada como destino y todos los mensajes recibidos por los clientes tienen las dirección anunciada como dirección fuente. Los servidores primario y de respaldo se conectan en la misma subred, la cual es la misma subred de la dirección anunciada. Este esquema asegura que el respaldo tiene una copia de cada requerimiento antes de que este disponible para el primario. A nivel de aplicación, se registran los mensajes HTTP de requerimiento y respuesta. El respaldo registra cada requerimiento mientras el primario procesa el requerimiento y genera una respuesta. Una vez que se genera una respuesta por el primario, se envía una copia completa al respaldo, antes de que esta respuesta sea enviada al cliente. La clave en este esquema es que el servidor de respaldo obtiene cada paquete TCP (datos y reconocimiento) del cliente antes que el servidor de respaldo. Así, la única manera de que el primario obtenga un paquete del cliente es cuando hay una copia en el servidor de respaldo. En operación normal, los clientes dirigen todos los paquetes hacia la dirección anunciada, los cuales son direccionados al servidor de respaldo. El módulo kernel del respaldo copia cada paquete al servidor primario. Entonces el servidor primario y el de respaldo procesan el paquete recibido. El servidor primario envía los paquetes de salida a los clientes; estos paquetes deben contener la dirección anunciada como dirección IP fuente para ser consistentes con el requerimiento original. Los paquetes de salida son procesados en el servidor de respaldo, y el kernel modifica el estado TCP, pero estos paquetes son interceptados y desechados. Se intercambian mensajes de latido entre los servidores y un monitor de latido que existe en los miembros de un par de servidores (primario y respaldo). Un proceso a nivel de usuario en cada servidor envía periódicamente paquetes UDP a su contraparte. Se detecta una falla, si hay una perdida del consecutivo. CoRAL considera dos modos de operación, modo duplex, cuando se activa la replicación y modo simplex, cuando la información no se duplica. Cuando hay una falla, un módulo del núcleo inicia una transición de modo duplex a modo simplex. Se considera que un servidor primario y uno de respaldo no fallan al mismo tiempo. Si el servidor primario falla, el núcleo del respaldo controlará que los paquetes

entrantes del cliente no sean redirigidos al servidor primario y los paquetes salientes sean enviados a los clientes en vez de ser descartados. Si el respaldo falla, el módulo del núcleo primario toma la dirección anunciada y los paquetes entrantes son recibidos directamente de los clientes en vez de ser recibidos por el respaldo.

2.2.5. HYDRANET-FT

HydraNet-FT [Shenoy00] fue presentado como una infraestructura para replicar servicios dinámicamente a través de una interred. HydraNet-FT modificó el protocolo de comunicación TCP en el lado del servidor para permitir entrega de mensajes uno-a-muchos (el mismo mensaje es difundido hacia varias réplicas), desde un cliente hacia replicas de servicio y entrega de mensajes muchos-a-uno (un mensaje único llegará hasta el cliente), de las replicas al cliente. Hay un canal de comunicación entre las replicas para proporcionar atomicidad y orden en los mensajes. El principal objetivo de HydraNet-FT es proporcionar servicios de tolerancia a fallas a través de una interred. HydraNet-FT proporciona algunas características, como replicación de servicios, a través de servidores geográficamente distribuidos, los cuales podrían estar servidos por diferentes proveedores de red. Soporta multicasting atómico [Mishra98] para asegurar que todos los servidores procesen las mismas operaciones y mantengan el mismo estado. El orden de mensaje, el cual asegura que todos los servidores atiendan los requerimientos de los clientes en el mismo orden. Este protocolo usa un mecanismo de detección de fallas de latencia baja para identificar las disrupciones de servicio, y una transparencia completa para los clientes, para proveer servicios de tolerancia a fallas a una población grande de clientes. HydraNet-FT está conformado por dos partes: servidores y re-directores. Los servidores son hosts configurados para proporcionar servicios replicados y tolerantes a fallas. Un re-director es un enrutador que mantiene la información de los servidores, servicios replicados, número de replicas de cada servidor, etc. HydraNet-FT maneja su esquema de replicación, replicando globalmente direcciones IP. Hay un servidor de origen y este puede contener un servicio o un conjunto de servicios que son replicados a través de uno o más servidores. Un servidor puede contener servicios tolerantes a fallas que operan en modo primario o respaldo. Dentro de una red puede existir más de un re-director. Para cada servicio replicado, el re-director conoce la ubicación del

servidor primario y sus correspondientes respaldos.

HydraNet-FT implementa un mecanismo para conservar atomicidad y orden de mensajes entre los respaldos. Los respaldos son conectados en una cadena margarita (en inglés, Daisy Chain) hacia el servidor primario. En este esquema, todas las réplicas (primaria y respaldos) reciben los paquetes desde el cliente, pero solamente la primaria responde al cliente. Los respaldos pasan la información de control de flujo solo al servidor previo de la cadena y al final el primer respaldo pasa esta información al primario. S_0 denota el servidor primario, y S_1, ..., S_N el servidor de respaldo N dentro de la cadena margarita. Si el servidor falla para recibir un paquete, se detecta que el ciclo de control de flujo se rompe, el cliente retransmite. Si el problema persiste, y el umbral de retransmisiones se alcanza, cualquier servidor puede iniciar una reconfiguración del conjunto de replicas. HydraNet-FT puede manejar la falla de un servidor primario y de un respaldo.

2.2.6. TCP Migratorio

En [Sultan02], se propone el TCP Migratorio (*Migratory TCP*, M-TCP), el cual soporta la migración de conexiones en vivo. Los servidores dentro de un conjunto de servidores similares pueden aceptar conexiones migradas y continuar su servicio. M-TCP transfiere el estado del último checkpoint de la conexión, junto con el estado específico del protocolo, al servidor destino para migrar hasta el punto final de un servidor. El protocolo asegura que el servidor destino reasuma el servicio, preservando el mismo estado, sin interrumpir el tráfico de la conexión. M-TCP proporciona un mecanismo llamado modelo de servicio cooperativo, en el cual un servicio de Internet es representado por un conjunto de servidores geográficamente dispersos. M-TCP tiene algunas limitaciones, tales como la falta de un soporte de tolerancia a fallas. M-TCP asume que el servidor origen de una conexión en migración está viva al tiempo de migración, así el estado de la conexión puede ser extraído por el servidor destino. Si el servidor origen colapsa, la migración no es posible.

2.3. Migración de Máquinas Virtuales

En esta subsección se detallan varios mecanismos de migración y de replicación. Se considera que un mecanismo de migración mueve una sola copia de una VM de un host a otro, a la vez. Un mecanismo de replicación mantiene en sincronía múltiples copias de una de máquina virtual original. Un mecanismo graba/reproduce (*record/replay*) captura y registra eventos no-determinísticos de una VM origen en un archivo de bitácora (*log file*). Durante la fase de reproducción, estos registros log guían a la máquina virtual como si hiciera una re-ejecución desde un checkpoint.

Las primeras estrategias de migración de máquinas virtuales

En la mitad de los 1980s, hubo dos avances tecnológicos importantes. El primero fue el desarrollo de microprocesadores poderosos. Los CPUs se manufacturaron en 64 bits. El segundo desarrollo fue la invención de redes computacionales de alta velocidad. El uso de Redes de Área Local (*Local Area Networks*, LANs) y Redes de Área Amplia (*Wide Area Networks*, WANs) permiten construir sistemas computacionales formados por un gran número de CPUs conectados a una red de alta velocidad. Estos sistemas fueron llamados sistemas distribuidos. Fue necesario desarrollar sistemas operativos para soportar sistemas distribuidos. Un sistema distribuido puede ser definido como una colección de computadoras independientes que dan la apariencia a los usuarios de una sola computadora [Tanenbaum01]. Un sistema puede alcanza este objetivo siendo transparente. Los sistemas distribuidos han sido desarrollados considerando varios conceptos de transparencia: Transparencia de ubicación; los usuarios no saben donde se encuentran ubicados los recursos. Transparencia en migración; los recursos pueden moverse a cualquier lugar sin cambiar sus nombres. Transparencia de replicación; los usuarios no saben cuantas copias de los datos existen. Transparencia en concurrencia; los usuarios pueden compartir recursos automáticamente. Transparencia en paralelismo; algunas actividades pueden ejecutarse en paralelo sin el conocimiento de los usuarios. Bajo esta filosofía, muchos sistemas distribuidos desarrollaron la migración de procesos como una característica inherente. Varios sistemas operativos fueron creados para trabajar en un ambiente distribuido, tales como Amoeba [Mullender90],

Mach [Tanenbaum95] o Chorus [Rozier91]. En los primeros trabajos en migración, se puede considerar la migración de procesos como una estrategia para copiar el estado completo de una VM a otra máquina física.

Zap

Zap [Osman02] introduce una capa de virtualización sobre el sistema operativo llamada Dominio de procesos (*PrOcess Domain*, pod). Un pod incluye un grupo de procesos dentro de un nombre de espacio privado. El grupo de procesos tiene la misma vista del sistema. Un pod permite relacionar identificadores virtuales con recursos del sistema operativo como identificadores de procesos y direcciones de red. Esta abstracción separa un pod de dependencias del sistema operativo y de otros procesos en el sistema.

Zap fue implementado en un módulo de kernel sin modificaciones al kernel, el cual intercepta llamadas al sistema según sea necesario para la virtualización y guarda y restaura el estado del kernel para la migración. El prototipo de Zap muestra que puede proporcionar migración de procesos con baja sobrecarga del sistema. Para soportar migración de procesos transparentes, Zap considera tres requerimientos que incluyen consistencia de recursos, conflicto de recursos y dependencia de recursos. El primer requerimiento es la necesidad de preservar la coherencia de recursos de nombres. Un sistema operativo contiene numerosos identificadores para sus recursos; incluyendo identificadores de procesos (PIDs), nombres de archivo y puertos. El sistema operativo considera que el identificador no cambia durante un ciclo de vida de un proceso. El segundo requerimiento esta relacionado a eliminar el conflicto de nombres cuando los procesos se migran. El tercer requerimiento es para eliminar dependencias residuales [Milojíčić00], las cuales son las dependencias del nodo donde el proceso fue creado (nodo hogar). Un ejemplo es la redirección de llamadas al nodo hogar, como las llamadas a un sistema de archivos u operaciones sobre dispositivos locales.

Los pods son unidades que pueden ser suspendidas para almacenamiento secundario, migrados a otra máquina y reasumidos transparentemente. Un pod puede contener cualquier número de procesos. La principal diferencia entre un pod y un ambiente de sistema operativo tradicional es que cada pod tiene su propio nombre de espacio virtual. Los nombres dentro de un pod son asignados de la misma manera que un sistema operativo

tradicional asigna nombres, pero los nombres están localizados en el pod. Debido a que el nombre de espacio es privado, no hay conflictos de nombrado de recursos en diferentes pods. La virtualización en Zap se basa en un mecanismo de checkpoint-reinicio para suspender, migrar y reasumir pods y sus procesos asociados. El mecanismo checkpoint-reinicio evita dejar componentes de migración residuales migrando el estado completo en una imagen; un pod puede ser migrado sobre la red o en almacenamiento local, el estado del sistema puede ser almacenado. Para migrar un pod, Zap primero suspende el pod, deteniendo todos los procesos en el pod, salvando los mapeos de virtualización y salvando todos los estados de proceso, incluyendo memoria, registros de CPU, manejadores de archivos abiertos, etc. En el host destino, Zap reasume el pod restaurando el ambiente virtualizado y restaurando los procesos detenidos. Finalmente, Zap habilita los procesos a continuar la ejecución en el ambiente pod restaurado. En el proceso de migrar un pod, se considera la migración de memoria. Zap asegura que la misma vista del sistema de archivos esta disponible a un pod en cualquier máquina donde el pod es ejecutado. Por esta razón, Zap no necesita guardar los segmentos de código que pertenecen a un ejecutable, por ejemplo, la página de texto del proceso o librerías ligadas en uso por el proceso. La migración de página de texto se realiza guardando las referencias a los archivos ejecutables y direcciones de memoria virtual a las cuales hacen referencia.

AutoPod

AutoPod [Nieh05] determina si se necesita aplicar un parche en un servidor, basándose en el estado del sistema y preservando la disponibilidad de servicios. La migración de servicios de aplicación hacia otro host es automática. El modelo de Autopod esta basado en una abstracción de máquina virtual llamado pod (*PrOcess Domain*). Para migración, se asume la misma versión de sistema operativo en todos los sistemas. Autopod usa una capa de virtualización para transladar el nombre de espacio del Autopod al nombre de espacio del sistema operativo del host.

AutoPod usa un mecanismo checkpoint-reinicio y un formato intermedio para representar el estado que necesita ser guardado en el checkpoint. Cuando se realiza un checkpoint, la representación en formato intermedio es almacenada y firmada digitalmente.

Durante el proceso de restauración se verifica la integridad de la imagen. De acuerdo a pruebas de rendimiento, iniciar un ambiente de escritorio desde cero puede tomar alrededor de 20 segundos, el checkpoint y reinicio de AutoPod puede tomar 851 ms y 942 ms, respectivamente. AutoPod proporciona un servicio que monitorea los repositorios de seguridad; descarga las actualizaciones de seguridad y a través del mecanismo de checkpoint/reinicio habilita las actualizaciones.

2.3.1. Migración en vivo

Conceptos Importantes

En el área de virtualización, se han propuesto varios mecanismos de migración y replicación para lograr esquemas de tolerancia a fallos y alta disponibilidad. En la copia ideal de una máquina virtual, el estado completo de una VM original debe ser transferido a la copia incluyendo memoria, disco y conexiones de red. La migración de disco local y conexiones de red no es un problema trivial. Para mantener la conectividad después de una migración, es necesario preservar las conexiones abiertas. Los clientes de redes deben ser atendidos con interrupciones mínimas. Si la migración es dentro de la misma LAN, la VM debe retener su dirección IP original después de la migración, generando una respuesta ARP no solicitada avisando de la nueva ubicación de la dirección IP de la VM migrada [Clark05]. En una migración WAN se han incorporado el uso de tecnologías de red tales como redes privadas virtuales (*Virtual Private Networks*, VPNs) y tunneling [Travostino06],[Wood11], servidores DNS [Bradford07]. También se emplea la operación en desconectado [Satyanarayanan05], [Chandra05], [Cáceres05]. En ocasiones, la migración de disco no se considera, porque se asume que se usan estrategias SAN o NAS [Morris03] entre las máquinas virtuales, pero un ambiente WAN debe considerar la migración de disco local. Típicamente, migrar el estado del disco representa el componente más largo del tiempo total de migración, debido a que el disco puede contener decenas o cientos de gigabytes. Se han usado muchas estrategias para migrar almacenamiento tales como grabar y reproducir (record/replay), transmisión por deltas o la implementación de la solución de almacenamiento de Dispositivo de Bloques Replicados Distribuidos (Distributed Replicated

Block Device, DRBD) [LINBIT11].

2.3.2. Migración de Memoria

Hay dos tipos de mecanismos de migración en vivo, pre-copia y post-copia. La migración de post-copia transfiere el contenido de la memoria de una máquina virtual después de que el estado de su procesador ha sido enviado al host destino. Difiere del método de pre-copia, el cual primero copia el estado de la memoria al destino, a través de un proceso repetitivo, entonces el estado del procesador es transferido a la VM destino. La migración de post-copia [Hirofuchi11] básicamente sigue los siguientes pasos:

1. La VM se detiene en el host fuente. El estado del procesador es enviado a la VM destino y el contenido de los registros del CPU virtual y los estados de los dispositivos son copiados al destino.

2. La VM se reasume en el destino sin ningún contenido de memoria.

3. Si la VM trata de acceder a páginas que no han sido aún transferidas, la VM es detenida temporalmente y las páginas faltantes son transferidas por demanda a través de la red desde la fuente. La VM es reasumida.

El mecanismo de pre-copia [Milojíčić00, Clark05] de memoria sigue los siguientes pasos:

1. La VM fuente continua en ejecución mientras todas las páginas de memoria son copiadas a la VM destino en la primera iteración. Las subsecuentes iteraciones copian solo aquellas páginas que fueron modificadas durante la ronda de transferencia previa.

2. Se detiene la VM fuente; se copia el estado de la Unidad Central de Procesamiento (UCP) y las páginas modificadas restantes. Se inicia la VM destino.

3. La nueva VM inicia la operación. Si esta máquina trata de acceder a una página que aún no ha sido transferida, esta página es traída de la VM fuente.

Ambos mecanismos de migración de memoria tienen pros y contras. En el primer paso de la migración de memoria por pre-copia, todas las páginas de memoria se transfieren a la VM destino; por lo tanto, el tiempo de migración se incrementa en proporción al tamaño

de la memoria de la VM. Además, las páginas modificadas deben ser copiadas iterativamente al destino. Si la VM ejecuta una intensa carga de trabajo de escritura accediendo a grandes cantidades de memoria, se generan numerosas páginas, modificadas y transferidas continuamente. En el peor de los casos, la migración en vivo nunca se completa [Hirofuchi11]. El mecanismo de pre-copia minimiza el tiempo de inactividad de la VM y la degradación de las aplicaciones cuando la VM ejecuta cargas de trabajo de lectura intensa. En la migración post-copia, después de que se inicia la VM, las páginas de memoria son transferidas a través de la red en el primer uso. Este mecanismo consume un tiempo de inactividad más corto, pero produce un tiempo total de migración más largo; y el rendimiento durante la migración probablemente sea considerablemente degradado cuando un gran número de páginas de memoria han sido traídas a través de la red sobre demanda. Sin embargo, en la migración de pre-copia, la verificación y envío iterativo de páginas modificadas entre los dos hosts podría consumir el total ancho de banda disponible entre ellos y causar la degradación de los servicios activos. En ambos esquemas, se han implementado mecanismos para aliviar sus inconvenientes. Por ejemplo, en post-copia [Hines09] se incluye la técnica de ballooning [Waldspurger02] para mejorar el rendimiento. En pre-copia, el ancho de banda se limita durante las copias subsecuentes para evitar la degradación de servicios [Clark05]. En la migración post-copia, las páginas faltantes se transfieren bajo demanda a través de la red desde la VM fuente, cada página de memoria se envía a lo sumo una vez, evitando la sobrecarga por transmisión duplicada del mecanismo de pre-copia.

La falla de nodo destino tiene diferentes implicaciones en los dos tipos de migración. Para la pre-copia, no hay problema si se presenta una falla en el nodo destino, pues el nodo fuente aún contiene la copia entera actualizada del estado de la memoria y del procesador de la VM. Sin embargo, si se utiliza el método de post-copia, el nodo destino tiene la copia más actualizada de la VM. Por lo tanto, una falla de la VM destino durante la migración post-copia se convierte en una falla crítica de la VM.

Migración en vivo se refiere a técnicas en las cuales una VM se mueve de un host a otro con tiempo de inactividad casi cero. Usualmente, el ambiente común para la migración en vivo es un servidor con VMs dentro de un centro de datos/cluster donde un estricto control sobre la red habilita la migración transparente de las conexiones. El uso de técnicas

de almacenamiento como una Red de Área de Almacenamiento (*Storage Area Network*, SAN) o Almacenamiento Anexo a una red (*Network Attached Storage*, NAS) elimina la migración de disco.

Xen

Xen [Barham03], propone un mecanismo de migración en vivo [Clark05]. Este método se enfoca en la migración de memoria. La migración en vivo de máquinas virtuales considera la minimización del tiempo de inactividad, el cual es el período mientras los servicios de las máquinas virtuales no están disponibles y el tiempo total de migración; el cual el período que comienza cuando la VM original inicia la migración, y termina cuando se detiene la VM. La migración de la máquina virtual puede describirse en tres fases. En la primera, la VM continua ejecutándose mientras usa el método de pre-copia [Milojíčić00], todas las páginas de memoria se copian a la máquina virtual destino en la primera iteración, las iteraciones subsecuentes copian solo las páginas que fueron modificadas durante la ronda de transferencia previa. En la segunda fase, se ejecuta el proceso de parar-y-copiar, se detiene la VM fuente; se copian las páginas modificadas remanentes y se inicia la VM destino. El tráfico se redirige a la nueva VM. En la tercera fase, la VM destino reasume la operación normal y si aún existen páginas no consistentes, estas son traídas desde la VM fuente. Se considera migrar las conexiones de red abiertas a través de un ARP no solicitado desde el host migrado, notificando que la IP tiene una nueva ubicación. Sin embargo, algunos enrutadores son configurados para no aceptar repuestas ARP (para prevenir ataques a la seguridad), así esta idea puede no funcionar en todos los escenarios. La migración de almacenamiento no es necesaria pues se considera el uso de una forma de almacenamiento compartido.

El *Writable Working Set* (WWS) está conformado por páginas modificadas tan frecuentemente que son candidatas para la fase de pre-copia. Se usan las tablas de páginas sombra de Xen para rastrear las estadísticas de modificación de todas las páginas usadas por un sistema operativo particular en ejecución. A través de estas estadísticas, se puede determinar el WWS. Xen considera dos métodos diferentes para iniciar y manejar la transferencia de estado. En el primero, se ejecuta la migración manejada principalmente fuera de

la VM, la cual está siendo migrada por un demonio de migración ejecutándose en la VM de gestión. En el segundo, la auto migración se ejecuta casi totalmente en la VM que está siendo migrada y una pequeña parte en la maquina destino. La migración manejada se ejecuta por demonios de migración, en las VMs de los hosts fuente y destino. Estos son responsables de crear la VM nueva en la máquina destino y coordinar la transferencia del estado de un sistema en vivo a través de la red. El software de control ejecuta las rondas para transferir la imagen de memoria. En la primera ronda, todas las páginas se transfieren a la máquina destino, en rondas subsecuentes, este copiado es restringido a páginas que fueron modificadas en rondas previas. Para registrar las páginas modificadas, Xen maneja las tablas de páginas sombra bajo el sistema operativo en ejecución. Todas las entradas página-tabla (page-table entries, PTEs) son inicialmente solo-lectura en las tablas sombra. Si el sistema operativo huésped trata de modificar una página de memoria, el fallo de página resultante se captura por Xen. Si el acceso de escritura es por las PTEs del huésped, entonces este permiso es extendido a la PTE sombra, colocando el bit apropiado en el mapa de bits de modificaciones de la máquina virtual. Cuando se determina que la fase de pre-copia ya no es de beneficio, usando métodos heurísticos, se envía un mensaje de control al sistema operativo requiriendo que esa fase sea suspendida. El mapa de bits de modificaciones se recorre una última vez para buscar páginas de memoria inconsistentes y estas son transferidas al destino junto con el estado de CPU-registros obtenidos de un checkpoint de la VM. Una vez que la información final es recibida en el destino, el estado de la VM en la máquina fuente puede ser descartado de manera segura. Entonces, la ejecución es reasumida, iniciando la nueva VM en el punto en el que la antigua VM ejecutó un checkpoint a sí misma. La auto migración maneja un esquema de pre-copia similar a la migración manejada. La dificultad de implementación mayor de este esquema es transferir un checkpoint del sistema operativo consistente. Esta dificultad fue solucionada con una fase final de dos estados parar-y-copiar. La primera fase deshabilita toda actividad de sistema operativo, excepto la migración, y entonces ejecuta un barrido final del mapa de bits de modificaciones, limpiando el bit apropiado cuando cada página es transferida. Las páginas que son modificas durante el barrido final, y que son aún marcadas como modificadas en el mapa de bits, son copiadas a un buffer sombra. La segunda fase transfiere el contenido del buffer sombra. Las actualizaciones de página son

ignoradas durante esta transferencia.

La evaluación de la migración en vivo fue ejecutada con diferentes cargas de trabajo en [Clark05]. Como prueba representativa, se puede mencionar la carga de trabajo de Apache, la cual es presentada por SPECweb99, una aplicación para evaluar servidores web y los sistemas que hospedan. Se usó una máquina virtual con 800MB de memoria y se migró con SPECweb99 en ejecución. La suspensión de la VM fue hecha cuando faltaban por enviarse 18.2MB de memoria. Hasta ese punto, la transmisión tomó 201 ms, después del cual fueron requeridos 9ms adicionales para que el dominio reasumiera la ejecución normal. El tiempo total de inactividad de 210ms que los clientes de SPECweb experimentaron fue suficientemente breve para mantener 350 clientes activos. En el artículo citado anteriormente, se probó un servidor de juegos en línea para multi-jugadores. Se midió el rendimiento de la migración en vivo con una VM de 64MB de memoria ejecutando Quake 3. Seis jugadores se unieron al juego y comenzaron a jugar en una arena compartida. Xen ejecutó la migración en vivo con un tiempo total de inactividad de 60 ms.

Auto-Migration

Hansen et al. establecen las bases para el mecanismo de migración de Xen [Clark05]. En [Hansen04] se implementan dos sistemas prototipo. La primera implementación Nomad-BIOS, es un prototipo en ambiente host para ejecutar varias instancias adaptadas de Linux concurrentemente. Este prototipo migra tales instancias entre servidores sin interrumpir servicios, lo que es llamado migración en vivo. NomadBIOS se ejecuta sobre el microkernel L4. NomadBIOS reduce el tiempo de inactividad usando la migración pre-copia, mantiene el sistema operativo huésped ejecutándose en el host original mientras se está migrando, rastreando los cambios a su espacio de direcciones y enviando las actualizaciones a la VM fuente un número de repeticiones. El tamaño de las actualizaciones decrece a cien kilobytes o menos. Para migrar conexiones de red, se transmite un paquete ARP gratuito a la Ethernet local, para informar a los puntos locales acerca del movimiento a una nueva interface. Los archivos son accedidos vía iSCSI o NFS; por lo cual, la migración de disco no se considera. El segundo método es llamado auto-migration; aquí el sistema operativo huésped ejecuta la migración sin la intervención del hipervisor, el sistema operativo huésped es migrado por si

mismo. Esta migración se basa en Xen [Clark05].

VMWare

VMware desarrolló un mecanismo de migración llamado *VMotion* [Nelson05] como parte del producto VMware Virtual Center que maneja VMware ESX Server. En [Waldspurger02] se presenta el mecanismo de memoria de VMware. ESX Server usa la técnica de ballooning para el manejo de memoria. Un pequeño módulo balloon se carga en un sistema operativo como un manejador pseudo-dispositivo o servicio kernel. Este módulo colabora con el servidor para reclamar páginas que son consideradas de menos valor para el sistema operativo huésped. Cuando el servidor necesita reclamar memoria, se instruye al manejador a "inflar" el balloon asignando páginas de memoria dentro de la VM. Cuando la memoria es escasa, el sistema operativo huésped decide que páginas en particular reclamar y si es necesario, las aloja en su propio disco virtual. El mecanismo de memoria ballooning permite que la memoria que fue asignada para una máquina virtual pueda ser dada a otra máquina virtual sin tener que detener la máquina virtual. Cada sistema operativo huésped que se ejecuta dentro de una máquina virtual direcciona un espacio de direcciones físicas basadas en cero como dentro de un hardware real. El servidor ESX da a cada VM esta ilusión, se virtualiza memoria física agregando un nivel extra de traducción de direcciones. Las direcciones de la máquina se refieren a la memoria de hardware actual, mientras que una dirección física es una abstracción de software usada para proporcionar la ilusión de memoria de hardware a una máquina virtual. Cada memoria virtual tiene un conjunto fijo de rangos de direcciones físicas que acceden a la memoria física. Todos los accesos directos a la memoria física de la máquina virtual son interceptados por el VMM. Entonces, el VMM traduce estas direcciones físicas a direcciones de máquina actual. Se usa el esquema precopia de memoria. Primero, se copia la memoria física desde la VM fuente y se marca como solo-lectura, así cada modificación puede ser detectada por el VMM. Cuando este proceso termina, pueden existir páginas modificadas por la VM en ejecución. Estas páginas se copian a la nueva VM. Este paso se repite hasta que el número restante de páginas es pequeño (16 megabytes) o hay una reducción en páginas modificadas de menos de 1 megabyte. VMware proporciona una Tarjeta de Red Ethernet Virtual (*Virtual Ethernet Network Card*, VNIC),

como parte de su plataforma virtual, la cual tiene una dirección MAC dentro de la red local. Una VNIC puede estar asociada a una o más tarjetas de red físicas. Debido a que la VNIC tiene una dirección MAC independiente de la dirección MAC física, las VMs pueden ser movidas de un host a otro sin detener los servicios y manteniendo las conexiones de red activas. Esto es posible, sólo si la VM nueva es abierta en la misma subred de la máquina virtual original. En almacenamiento, VMware considera que las computadoras se conectan a un servidor SAN o NAS.

KVM

KVM. La Máquina Virtual basada en Kernel (*Kernel-based Virtual Machine*) o KVM [Machine12] es un Monitor de Máquinas Virtuales que permite virtualización completa para Linux sobre hardware x86. Debido al creciente impacto de la virtualización, los vendedores de hardware tales como Intel y AMD han añadido extensiones a la arquitectura x86 que han hecho la virtualización más fácil (Intel VT o AMD-V). KVM consiste en un módulo cargable, kvm.ko, y la funcionalidad de la arquitectura especifica es proporcionada por dos módulos de arquitectura específica, kvm-intel.ko y kvm-amd.ko. Usando KVM, es posible ejecutar múltiples máquinas virtuales de Linux o Windows no modificados.

En [Kivity07] se implementa la migración en vivo de KVM. La migración en vivo usa la estrategia de pre-copia, lo cual significa que si una página del sistema operativo huésped es modificada después de que ha sido copiada, deberá ser copiada otra vez. KVM implementa un registro de páginas modificadas, el cual es usado como un mapa de bits de páginas modificadas desde la última llamada. KVM registra las páginas huésped como solo-lectura y las habilita para escritura después del primer acceso de escritura. La migración en vivo se completa en tres fases [Clark05]. Este es un proceso iterativo, cada iteración copia memoria al host destino. En la primera fase, todas las páginas de memoria son marcadas como modificadas y se inicializa un mecanismo de rastreo de modificaciones. En la segunda fase, se copian las páginas marcadas como modificadas. El proceso iterativo de copiado de páginas continúa mientras la tasa máxima de transferencia hacia la máquina destino no se exceda. Las páginas modificadas, pero no copiadas, son usadas para estimar el tiempo de inactividad si la migración inicia la tercera fase. Si el tiempo de inactividad estimado es alto,

comparado con un valor objetivo, el algoritmo itera hasta que predice un valor menor que el valor objetivo. Cuando se alcanza el tiempo de inactividad objetivo, la migración entra en la tercera fase, donde la máquina virtual fuente (y aplicaciones) se detiene. Las páginas modificadas son transmitidas a la máquina destino, se cargan los registros y la ejecución se re-inicia en el nuevo host [Ibrahim11].

Migración Post-Copia de Máquinas Virtuales

En [Hines09] se presenta un diseño, implemtación y evaluación de una migración en vivo de máquinas virtuales basada en post-copia a través de una LAN Gigabit. La migración post-copia transfiere el contenido de la memoria de la VM después de que el estado del procesador ha sido enviado al host destino. El mecanismo de pre-copia, copia primero el estado de la memoria a través de múltiples iteraciones y después transfiere el estado del procesador hacia el host destino. En este trabajo, se propone y evalúa la estrategia de post-copia para migración en vivo de máquinas virtuales. La eficiencia de la post-copia depende del proceso de minimizar el número de páginas faltantes (o fallas de red), para acelerar la migración, las páginas de memoria se atraen desde la fuente antes de que se requieran en la máquina virtual destino. La manera en la cual las páginas son traídas crea diferentes variantes de post-copia, proporcionando mejoras. La migración post-copia combina cuatro técnicas para traer páginas de memoria desde la fuente: paginado sobre demanda, empuje activo, prepaginado y auto-ballooning dinámico (*Dynamic Self-Ballooning*, DSB). Cuando se usa el paginado sobre demanda, la VM se re-inicia en el destino, cuando se produce la falla de memoria, esta es servida por un requerimiento al nodo fuente a través de la red. El empuje activo reduce la duración de las dependencias residuales del host fuente, empujando las páginas desde la VM fuente hacia la VM destino, la cual continúa en ejecución. La prepaginación usa las fallas de la red como puntos de inicio para predecir la ubicación de acceso de las páginas de la VM y buscarlas en la vecindad de una falla de red antes de que sea accedida por la VM. Las páginas de la VM en la fuente son almacenadas en un dispositivo de psedo-paginación en memoria, el cual reside completamente en memoria. El prepaginado aplica un algoritmo bubbling, el cual comienza un empuje de páginas desde una página pivote dentro del dispositivo de psedo-paginación y transmite simétricamente las páginas

ubicadas alrededor del pivote. Las páginas que han sido transmitidas se omiten. El método de bubbling puede ser ejecutado con uno o múltiples pivotes. Transferir páginas libres puede ser un desperdicio de recursos de red y de CPU e incrementaría el tiempo total de migración sin importar que algoritmo de migración se use. DSB reduce el número de páginas libres transferidas durante la migración, mejorando el rendimiento tanto de la migración pre-copia como de la migración post-copia. La VM ejecuta ballooning [Waldspurger02] continuamente a lo largo de su ejecución en su ciclo de vida. La migración post-copia propuesta se compara contra la migración pre-copia de Xen. Tanto la VM como el Dominio 0 son configurados para usar dos CPUs virtuales en cada experimento; el tamaño por omisión de la VM es de 512 MB. Se analizaron diferentes métricas de rendimiento con cuatro aplicaciones: a) SPECWeb 2005, el sistema es configurado dentro de una VM, y seis clientes externos requieren conexiones; b) Un cliente bit torrent, una aplicación distribuida multi-punto y es ligeramente CPU intensiva; c) Compilación del Kernel Linux: considerada por consistencia; d) Netperf, que es configurado dentro de la VM. Se evalúan cuatro métricas: tiempo de inactividad, tiempo total de migración, páginas transferidas y páginas faltantes. El método de post-copia es efectivo solo cuando la gran mayoría de las páginas alcanzan el destino antes de que sean requeridas en la VM destino, en este caso se convierten en páginas faltantes menores en vez de páginas faltantes mayores. La cantidad de fallas mayores comparadas con páginas faltantes menores es un parámetro de efectividad de post-copia. Para todas las aplicaciones, a excepción de SPECweb, post-copia reduce el total de páginas transferidas a más de la mitad. El resultado más significativo es cuando el algoritmo de prepaginación de post-copia convierte las fallas de página de red en fallas menores para las aplicaciones más grandes como SPECweb y bittorrent en 79 % y 83 %, respectivamente. Post-copia reduce el tiempo de migración total para todas las aplicaciones comparadas con pre-copia, en algunos casos hasta más del 50 %. Pero el tiempo de inactividad es mucho mayor para post-copia que para pre-copia.

2.3.3. Migración Suspender/Reasumir

Conceptos importantes

Chen y Noble [Chen01] sugieren que la tecnología de máquinas virtuales puede ser usada para proporcionar movilidad de usuario de una manera segura. La tecnología de suspender/reasumir está basada en esta idea. La migración suspender/reasumir se refiere al movimiento de una VM desde un host hacia otro cuando la VM está inactiva durante el traslado. Usualmente, este tipo de migración se lleva a cabo a través de una WAN. Las conexiones de red son típicamente desconectadas y deben ser re-establecidas. Para una migración en WAN es crucial no solo transferir las imágenes de las VMs sino también transferir el estado persistente, las conexiones actuales de red y el soporte para operación desconectada. Una técnica para optimizar la transferencia de disco es el uso de deltas. En un host fuente, un proceso intercepta las operaciones de escritura y genera las deltas, las cuales son unidades de comunicación que contienen los datos escritos, la ubicación en disco y el tamaño de los datos escritos. El proceso examina los datos almacenados y localiza los bloques o bytes que han sido modificados desde la última escritura. Los datos cambiados, pueden ser enviados al host destino a través de LAN o WAN, en vez de enviar la información completa.

La operación desconectada [Kistler92] es un modo de operación que habilita a un cliente a continuar accediendo a información crítica de un repositorio de información compartida durante fallas temporales. Mientras se encuentra desconectado, los requerimientos al sistema de archivos se apoyan en el contenido de la caché. Cuando la desconexión termina, las modificaciones se propagan al servidor correspondiente.

Internet Suspender/Reasumir

Internet Suspender/Reasumir (ISR) [Kozuch02, Satyanarayanan07] presenta un proyecto donde el estado completo de una máquina virtual puede ser migrado. ISR está basado en la idea de que un VMM encapsula el estado de ejecución volátil de una VM y el VMM transfiere el estado de sus VMs hacia archivos en el sistema de archivos local dentro de la máquina anfitriona. Cuando una VM es suspendida, el estado volátil es transferido a archivos

hasta un punto de suspensión. Estos archivos, incluyendo el sistema operativo son copiados hacia una máquina remota, donde la VM puede ser reasumida. El tiempo de inactividad depende directamente del tamaño del archivo a ser transferido. ISR usa la tecnología del almacenamiento distribuido. Cada VM encapsula un sistema operativo huésped y las aplicaciones huéspedes en un estado de ejecución y un estado personalizado del cliente y ambos son llamados parcela. La capa de almacenamiento distribuido transporta una parcela a través del espacio (desde un sitio de suspensión hacia un sitio de reanudación) y del tiempo (desde un instante de suspensión hacia un instante de reanudación). Los usuarios pueden poseer múltiples parcelas, así como pueden ser dueños de múltiples máquinas con diferentes sistemas operativos o suites de aplicación. Esta estructura ISR permanece invariante a través de diferentes versiones ISR. La capa ISR fue implementada en dos partes. Una parte fue un módulo de kernel cargable llamado Fauxide que sirve como manejador de dispositivo para un pseudo-dispositivo. VMware fue configurado para usar este pseudo-dispositivo para el estado de la VM. Fauxide redirige los requerimientos de VMware hacia este pseudo-dispositivo a un proceso a nivel de usuario llamado Vulpes, el cual fue el segundo componente de la capa ISR. Vulpes implementa la política de estado-transferencia de la VM, el mapeo del estado de la VM a un directorio de 256-Kbytes en Coda y el control (*hoarding*) de estos archivos. Algunos componentes de cada capa han cambiado con el tiempo. Por ejemplo, el VMM fue VMware para las primeras versiones de ISR, pero en las últimas versiones pueden usarse VMware, KVM o Xen. Estos VMMs suportan un modo en el cual una partición de disco local contiene el estado de la VM. El software cliente de ISR encripta datos desde una parcela antes de entregárselo a la capa de almacenamiento distribuido. Ni los servidores ni los caches cliente persistentes usados por el mecanismo de almacenamiento distribuido contienen algún estado del usuario sin encriptar. ISR ha implementado algunas versiones como ISR-1, ISR-2 e ISR-3. En la última versión Coda ha integrado un mecanismo para usar almacenamiento distribuido. En la primera implementación de ISR se reportaron algunos resultados. La implementación del prototipo ISR es capaz de suspender una VM en el cliente 1, reasumirla en el cliente 2, y viceversa.

Una medida de rendimiento fue reportada en ISR-1. Para las pruebas se consideró que antes de cada operación de reasumir en un cliente, se ejecuta una reinicialización

para asegurar un caché NFS vacío. Se consideran dos eventos de suspensión: Suspensión en caliente, la cual ocurre en una VM en un corto tiempo después de un evento reasumir y una suspensión en frío, la cual ocurre mucho tiempo después de un evento reasumir. Reasumir en frío toma 125 segundos, una suspensión en caliente 114 segundos y una suspensión en frío 146 segundos. Estos resultados son tolerables para algunos usuarios, pero en general, toman más tiempo que el suspender-reasumir en una laptop. ISR-2 incorpora la operación en desconectado, donde los usuarios pueden usar el estado de la caché cuando se encuentran desconectados de la red. El cliente almacena las modificaciones y las reintegra cunado la conectividad de la red es restablecida.

Cápsula

Una cápsula [Sapuntzakis02] es definida como un estado de hardware que incluye el sistema operativo entero, las aplicaciones y los procesos en ejecución. Este estado es movido a través de la red e incluye el estado de sus discos, memoria, registros de CPU y dispositivos de E/S. Este proyecto esta basado en el Colectivo (*the Collective*) [Chandra05] y maneja la arquitectura x86. Las cápsulas pueden ser suspendidas de ejecución, serializadas y después reasumidas. Las cápsulas pueden ser movidas entre máquinas para balancear cargas o para recuperación de errores. Una cápsula inactiva puede contener gigabytes de almacenamiento en disco, mientras una cápsula activa puede incluir cientos de megabytes de datos en memoria, así como también registros de máquina interna y estados de dispositivos de E/S. Para copiar la cápsula entera a otra ubicación física podría tomarse un tiempo largo. Pero este proyecto implementa algunas optimizaciones para reducir los requerimientos de almacenamiento, tiempo de transferencia y tiempo de inicialización a través de la red de una cápsula. Se presentan cuatro optimizaciones:

1. Para reducir el estado de memoria antes de la serialización, se emplea la técnica de ballooning. Un programa balloon requiere al SO un número grande de páginas físicas. Entonces, el programa pone a ceros las páginas, haciéndolas fácilmente compresibles. Ballooning reduce el tamaño del estado de la memoria comprimida y así reduce el tiempo de inicio de las cápsulas. Esta técnica trabaja bien si la memoria tiene muchas

páginas libres cuyo contenido no es comprensible. Esta memoria no es transferida y estas páginas son las primeras en ser limpiadas por el proceso ballooning.

2. Cada vez de que una cápsula inicia, todas las modificaciones hechas a disco son almacenadas en un disco por separado, usado la técnica copia-sobre-escritura (*copy-on-write*, COW). Las cápsulas son creadas en orden jerárquico, así cada cápsula hija puede verse como una herencia de la cápsula padre. Las diferencias en estado del disco entre padre e hijo son capturadas en un disco virtual separado copia-sobre-escritura. Esta estrategia reduce el costo de almacenar un disco cápsula pues solamente las diferencias son capturadas.

3. En vez de enviar el disco completo, las páginas son traídas sobre demanda cuando la cápsula esta en ejecución.

4. El tiempo de transferencia decrece al enviar un hash de un bloque de datos en vez de los datos completos. Los hashes resistentes a colisión son usados para evitar enviar páginas de memoria o datos de disco que ya existen en el destino. Se espera encontrar bloques de datos idénticos entre imágenes de discos y memorias, aún a través de cápsulas de diferentes usuarios. Todo el tráfico de red es comprimido con gzip.

El Colectivo

El Colectivo [Chandra05] es un sistema que proporciona escritorios administrados a usuarios de computadoras personales (*Personal Computer*, PC). Los administradores del sistema crean tales ambientes de escritorio llamados aplicaciones virtuales, los cuales incluyen el sistema operativo y todas sus aplicaciones instaladas. La PC destino ejecuta el software cliente, llamado Transceiver de Aplicación Virtual (*Virtual Appliance Transceiver*, VAT), que atrapa y ejecuta las últimas copias de las aplicaciones localmente y continuamente almacena los cambios de datos de usuario en un repositorio de red. El Colectivo está disponible para máquinas con arquitectura x86 y usa la tecnología de virtualización VMware GSX Server [VMware12]. Las PCs pueden estar unidas a una LAN, a una WAN o aún desconectadas de la red como en una laptop. Los usuarios pueden acceder sus escritorios desde un cliente Colectivo; también pueden traer un dispositivo de arranque que convierte

una PC en un cliente. Un VAT es construido usando un CD de arranque de Knoppix, el cual automáticamente detecta el hardware disponible a tiempo de ejecución y carga los dispositivos de Linux. El Colectivo presenta un manejador de sistema basado en caché, el cual separa el estado de la computadora en dos partes: El estado del sistema, el cual consiste del sistema operativo y todas las aplicaciones instaladas y el estado de usuario, el cual consiste de un perfil de usuario, sus preferencias y archivos de datos. En este modelo, las aplicaciones y el estado de usuario son almacenadas separadamente en repositorios de aplicación y repositorios de datos accesibles por la red. Un usuario puede acceder cualquier PC o VAT y obtener el acceso a cualquier aplicación con su correspondiente autorización. El VAT ejecuta funciones como autentificar usuarios, traer y ejecutar las últimas copias de aplicaciones, almacenar el estado de usuario en un repositorio de datos, manejar la caché para reducir la cantidad de datos que necesita ser traída a través de la red y mejorar el rendimiento. Un VAT puede ejecutar un proceso de manera previa para minimizar las omisiones en el caché, un VAT ejecuta un proceso de manera previa e incluso un cliente puede traer una aplicación completa a caché para trabajar en modo desconectado.

SoulPads

En [Cáceres05] se presenta un sistema llamado SoulPad es presentado. Este sistema permite a un usuario reasumir una sesión de cómputo personal que fue suspendida en otra máquina. El SoulPad divide la máquina del usuario en un cuerpo (despliegue, CPU, RAM, E/S) y un alma (estado de sesión, software, datos y preferencias). El alma es transportada en un dispositivo portable, el SoulPad. El alma puede reasumirse en cualquier computadora personal basada en x86 sin software pre-cargado. Las computadoras que reasumen SoulPad son llamadas EnviroPCs. Las conexiones entre el SoulPad y la EnviroPC son hechas a través de USB 2.0. El usuario puede reasumir un estado suspendido en modo desconectado. Las EnviroPCs no necesitan software precargado. Los usuarios migran de una máquina a otra a través del disco SoulPad. Un disco SoulPad contiene tres elementos. Primero, el SO anfitrión Knoppix que inicia sobre las EnviroPCs y obtiene la habilitación del hardware vía auto-configuración. Segundo, el VMM, VMware Workstation, el cual soporta operaciones de suspender/reasumir en máquinas virtuales y una diversidad de sistemas op-

erativos huéspedes. Tercero, una máquina virtual que ejecuta las aplicaciones de los usuarios sobre un sistema operativo huésped (Windows o Linux). La partición del disco que contiene las imágenes de VM son encriptadas usando el cifrado de bloque AES128.

Cloudlets

La arquitectura donde un usuario móvil explota la tecnología de VM para instanciar servicios de software personalizados en un cloudlet cercano y usa ese servicio a través de una LAN inalámbrica es discutida en [Satyanarayanan09]. El dispositivo móvil típicamente funciona como un cliente ligero con respecto al servicio. Un cloudlet es definido como un cluster de computadoras que esta bien conectado a Internet y está disponible para ser usado por dispositivos móviles cercanos. Es bien conocido que el hardware móvil es pobre en recursos, con respecto a los clientes estáticos y servidores de hardware. Una solución para mejorar esta desventaja es el uso de computación en nube.

CloneCloud

En [Chun11] se presenta CloneCloud. Este particionador de aplicaciones y ejecución hace posible que aplicaciones móviles no modificada se ejecuten en una máquina virtual a nivel de aplicación descargando parte de su ejecución dentro de dispositivos clonados [Chun09] operando en una nube computacional. El sistema transforma la ejecución de una sola máquina de un dispositivo móvil en una ejecución distribuida dentro de una nube. Una VM a nivel de aplicación es una máquina virtual abstracta que permite la independencia del hardware y del sistema operativo. A tiempo de ejecución la VM ejecuta bytecodes de métodos con hilos. Un mecanismo de partición crea una partición, la cual es una elección de los puntos de ejecución donde la aplicación migra parte de su ejecución y estado entre el dispositivo y el clon. Durante la ejecución, si se encuentra un punto de migración, el hilo de ejecución es suspendido y su estado (incluyendo estado virtual, contador de programa, registros y pila) es empacado y enviado a un clon sincronizado. En el clone, el estado del hilo es copiado a un nuevo hilo con la misma pila y objetos del montículo (heap) y después reasumido. Si el hilo migrado alcanza un punto de reintegración, este es suspendido, empaquetado y después enviado de vuelta al dispositivo móvil. Finalmente, el hilo empaquetado

regresado se fusiona con el estado del proceso original. Un optimizador matemático elige los puntos de migración que optimizan el tiempo de ejecución total o el consumo de energía del dispositivo móvil de acuerdo a la aplicación y al modelo de costo. Los resultados de las pruebas han mostrado que algunas aplicaciones alcanzan una aceleración en ejecución de hasta 20x y hasta un decremento de 20 veces la energía gastada por el dispositivo móvil.

2.3.4. Migración en vivo a través de redes de área amplia

Conceptos importantes

En la migración a través de redes de área amplia es esencial la transferencia del estado completo de la máquina virtual incluyendo disco, conexiones de red abiertas y páginas de memoria. La siguiente sección presenta algunos trabajos que mueven el estado completo de la VM de un host a otro por una migración en vivo a través de la WAN.

Migración continúa en vivo de máquinas virtuales a través de una red de área metropolitana (*Metropolitan Area Network*, MAN)/ red de área amplia.

La migración en vivo a través de WANs es un tema interesante que ha atraído la atención en años recientes. Para lograr migración en vivo de máquinas virtuales a través de WANs, se emplea un método usando túneles IP para garantizar conexiones de red en el demostrador "VM turntable" [Travostino06]. El túnel IP entre el host fuente y destino es configurado para enviar paquetes desde y hacia las aplicaciones de cliente. La migración es ejecutada a través de rutas dedicadas, las cuales son circuitos de 1 Gbps de capacidad; creadas entre dos sitios, Amsterdam y San Diego. Las máquinas virtuales fuente y destino ejecutan una aplicación de demostración para detección de rostros, la cual trae imágenes continuamente desde el almacenamiento. Es configurada con 200MB de memoria. La migración en vivo causa un tiempo de inactividad de aplicación de 0-8-1.6 segundos, la cual se compara contra un tiempo de inactividad de configuración intra-LAN que es de 5-10 veces más alto.

Migración en vivo de máquinas virtuales en área amplia

La migración en vivo de máquinas virtuales en redes de área local se ha concentrado en desarrollar métodos eficientes para transferir estados de memoria de una VM. Sin embargo, para la migración de red de área amplia es importante transferir recursos adicionales de una VM como estado persistente local (su sistema de archivos) y las conexiones de red en curso. En [Bradford07] los autores combinan una solución a nivel de bloques con pre-copiado y el estrangulamiento de la escritura (*write throttling*) para tranferir un servidor web en ejecución incluyendo su estado persistente local con mínima disrupción. La combinación de dynDNS con tunneling transfiere las conexiones existentes. Las nuevas son redirigidas a la nueva ubicación de red. La VM original permanece en operación en el host destino. Durante la migración de almacenamiento se utiliza un dispositivo de bloques a nivel de usuario para registrar y enviar los accesos de escritura al destino. El manejo de las conexiones de red se realiza por medio de un esquema de redirección y la notificación de las direcciones nuevas a través de dyn DNS. El sistema es implementado como parte de la plataforma XenoServer y usa la migración de memoria de Xen. El mecanismo de migración es coordinado por un cliente de migración, ejecutándose en la fuente, en continúa comunicación con el demonio de migración, ejecutándose en el destino. El sistema sigue algunas fases. La fase de inicialización configura el cliente y los procesos del servidor, los cuales manejan la transferencia de la imagen de disco al destino. El estado de transferencia de datos pre-copia la imagen de disco de la VM destino y la registra en archivos. Finalmente, la VM fuente es pausada, las páginas modificadas restantes son copiadas al destino y la VM es reasumida. Como la VM continúa la ejecución en la fuente durante las etapas de transferencia de datos de disco y migración de memoria. Durante la transferencia de datos de disco y migración en vivo, las operaciones de escritura en la VM fuente son interceptadas. Estas operaciones son empaquetadas como deltas, las cuales son unidades de comunicación que consisten de los datos escritos, la ubicación de la escritura en disco y el tamaño de los datos escritos. Las deltas son enviadas y encoladas en la VM destino para su aplicación al disco imagen. Si la velocidad a la cual la VM ejecuta accesos de escritura es muy alta, se aplica un estrangulamiento de escritura para minimizar la congestión de ancho de banda. Esto es en base a un

umbral y un retraso; cuando la VM alcanza el número de escrituras definidas como umbral, un se retrasa un nuevo intento de escribir a través de un parámetro de retraso. Después de que la transferencia de datos finaliza, la migración en vivo de Xen inicia y se aplican ordenadamente las deltas que se encuentran encoladas en el disco imagen destino. Cualquier nueva escritura que ocurre durante esta etapa es encolada en el disco imagen destino y de nueva cuenta aplicada en orden. Después de que la migración en vivo de Xen se completa, la VM original se pausa y comienza un esquema de redirección si la VM se migra a través de Internet. Iproute2 crea un túnel IP entre la dirección IP fuente y la dirección IP destino. Cuando la migración termina, el DNS dinámico se actualiza, y las nuevas conexiones se redirigen a la nueva dirección IP de la VM. Los autores obtuvieron algunos resultados notables. Se ejecutó un servidor web con un tablón de anuncios. Se evaluó la migración desde fuente a destino tanto en LAN como en WAN. En una LAN, la transferencia de datos de disco comienza a los 25 segundos, la aplicación de deltas ocurre después de 62 segundos. La migración en vivo se inicia después de 125 segundos y el experimento finaliza después de 225 segundos. La migración de un servidor web en ejecución conteniendo una base de datos de 250 clientes tiene un tiempo de disrupción de 3.09 segundos. Para emular una red de área extendida, se usó la interface de configuración del tráfico de red de Linux a 5Mbps y una latencia de 100ms entre los servidores A y B. Esto es representativo de la conectividad observada entre un servidor en Londres y otro en la costa este de Estados Unidos. En un ambiente WAN, la transferencia de datos de disco comenzó después de 260 segundos y la migración en vivo de Xen después de 2250 segundos. La migración terminó después de 3600 segundos con una disrupción de 68 segundos.

Un mecanismo de migración de almacenamiento en vivo a través de la WAN

En [Hirofuchi09] se propone un mecanismo de acceso a almacenamiento que soporta la migración de máquinas virtuales en vivo a través de la red. De manera rápida reubica discos de máquinas virtuales entre sitios fuente y destino con mínimo impacto sobre el rendimiento de E/S. El mecanismo propuesto trabaja como un servidor de almacenamiento de un protocolo de E/S de almacenamiento a nivel de bloque (por ejemplo, iSCSI y NBD). Para mover discos virtuales en línea entre los sitios se aplican técnicas de copiado sobre

demanda y en fondo (background). Este mecanismo trabaja para Xen y KVM sin ninguna modificación para sistemas operativos. Los experimentos emulan un ambiente de WAN para servidores de datos remotos a través del Océano Pacífico (por ejemplo, Tokio y San Francisco). Los experimentos muestran que el mecanismo propuesto mejora el rendimiento de E/S del acceso a almacenamiento remoto, minimizando la degradación de rendimiento durante la migración de disco. El mecanismo de migración de almacenamiento está compuesto principalmente de un servidor destino y un servidor proxy de un protocolo de E/S a nivel de bloques. La tecnología de Dispositivo de Bloques de Red (*Network Block Device*, NBD) es usada para construir este prototipo llamado xNBD. NBD conecta los servidores fuente y destino a un objetivo y servidor proxy, respectivamente, usando TCP/IP. Los discos virtuales son accedidos por una VM a través de archivos de dispositivos de bloques (por ejemplo, /dev/nbd0) en un sistema operativo anfitrión. Antes, la migración en vivo trabaja de la misma forma que en un servidor objetivo NBD, el cual redirige los requerimientos de E/S desde la VM hacia un archivo de imagen del disco. Después de que se inicia la migración en vivo [Clark05], el mecanismo propuesto trabaja junto con la migración de memoria con un VMM y el servidor destino continúa modificando los bloques de disco del archivo imagen. En la última parte de la migración de memoria, la VM es reiniciada en el sitio destino, entonces la operaciones de E/S son ejecutadas en el sitio destino a través del servidor proxy. En este momento, el servidor proxy comienza la migración de disco. El servidor proxy continúa el copiado de bloques remotos a través de la conexión NBD, hasta que todos los bloques requeridos se encuentran en el sitio destino. Después de esto, el servidor proxy termina la conexión NBD; la VM no depende del servidor objetivo en el sitio fuente. El servidor proxy copia los bloques de disco remoto usando dos métodos, trabajando en paralelo el copiado sobre demanda y en fondo. El método sobre demanda trabaja si la VM destino trata de leer un bloque que no se encuentra en la VM destino, el servidor proxy recupera el bloque de datos del servidor fuente. El mecanismo de copiado en fondo copia los bloques que aún permanecen en el sitio fuente. Los bloques son traídos de manera pro-activa, antes de que la VM los acceda. El módulo netem del Kernel de Linux el ambiente WAN experimental. Los parámetros de configuración corresponden a una red entre Tokio y la costa este de los Estados Unidos. Los resultados experimentales muestran que

el mecanismo propuesto reubica discos entre los sitios fuente y destino con un rendimiento de E/S comparable a las operaciones realizadas en un ambiente LAN.

CloudNet

La tecnología emergente de computación en nube [Marston09, Mel09], la cual ofrece el uso de servicios a través de la red, ha cambiado el alcance del manejo de los recursos. Antes se asignaban recursos en un solo servidor, con computación en nube se pueden manejar grupos de recursos dentro de un centro de datos. La arquitectura CloudNet [Wood11] como una estructura de nube consiste de plataformas de computación en red enlazadas con una red privada virtual para proporcionar conectividad segura y sin interrupciones entre las empresas y los sitios de centros de datos en red. Como una contribución, los autores presentan algunas optimizaciones para minimizar el costo de transferencia del almacenamiento y memoria durante las migraciones a través de un ancho de banda bajo y enlaces internet de alta latencia. CloudNet incluye la abstracción de Conjunto de Nube Virtual (*Virtual Cloud Pool*, VCP), la cual permite una conexión entre servidores dentro de la WAN que parece como un solo conjunto lógico de recursos conectados a través de la LAN. CloudNet usa las tecnologías existentes de VPN para construir un VCP y mover el estado de la memoria y disco usando esta estructura. CloudNet implementa varias optimizaciones de WAN para habilitar la migración en enlaces de ancho de banda bajo. Se implementa un algoritmo de migración en vivo adaptativo que ajusta dinámicamente el estado de la migración de memoria basado en el comportamiento de la migración. También implementa mecanismos tales como la eliminación de redundancia basada en contenido y páginas delta dentro del hipervisor para reducir el volumen de datos enviados durante el proceso de migración. Colectivamente, estas optimizaciones minimizan el tiempo total de migración, el tiempo de inactividad de la aplicación y el volumen de datos transferidos.

Cloudnet se implementa usando la plataforma Xen y la implementación comercial layer-2 VPN. Otra herramienta empleada es Servicios LAN Privados Virtuales (*Virtual Private LAN Services*, VPLS) que conecta múltiples puntos en un solo segmento LAN. Esto permite que los recursos de la nube parezca que se encuentran dentro de la LAN de la propia empresa. Se asume que hay una relación de confianza entre la empresa, el proveedor

de red y el proveedor de computación en nube. CloudNet sigue los siguientes pasos para la migración en vivo de VMs:

1. Establecer la conectividad virtual entre los puntos terminales de VCP.
2. Si el almacenamiento no es compartido, se transfiere todo el estado de disco.
3. La transferencia del estado de memoria de la VM a un centro de datos destino es ejecutado mientras la VM fuente continúa ejecutándose.
4. Después de que el estado de disco y memoria han sido transferidos, la VM fuente es pausada para la transición remanente del estado de la memoria y del procesador al host destino. Este proceso no interrumpe las conexiones de red activas entre la aplicación y sus clientes.

CloudNet usa el almacenamiento distribuido de Dispositivo de Bloques Replicados Distribuidos (*Distributed Replicated Block Device*, DRBD) para migrar almacenamiento a un centro de datos destino. Para reducir el impacto al rendimiento de esta sincronización, CloudNet usa el modo de replicación asíncrona de DRBD durante este paso. Una vez que el disco remoto se encuentra en un estado consistente, CloudNet cambia al esquema de replicación síncrona y la migración en vivo del estado de la memoria de la VM inicia. Cuando la migración se completa, el disco del nuevo host se convierte en primario y el disco del original es deshabilitado. CloudNet usa el código de Xen para lograr la migración de memoria e implementa un paro inteligente y una optimización de copia para reducir el número innecesario de iteraciones y minimizar el tiempo de pausa. Este punto fue detectado donde el número de páginas enviadas es igual al número de páginas modificadas. La técnica de eliminación de redundancia basada en contenido es otra mejora. Es usada para eliminar la redundancia de datos mientras se transfiere el estado de la memoria y el disco de la VM. Después de esta primera iteración, muchas páginas se transmiten porque han sido modificadas. Otro mecanismo para reducir el consumo de ancho de banda es conservar un caché de las páginas que se transmitieron antes, y entonces enviar solo la diferencia entre las páginas en caché y las páginas modificadas. Este tipo de comunicación en delta es combinada con la optimización CBR. La evaluación experimental fue obtenida usando tres centros de datos esparcidos por

los Estados Unidos. Los resultados muestran que las optimizaciones de CloudNet decrementan el tiempo de migración de memoria y de pausa de un 30 a 70 %; en un conjunto de migraciones de VMs en una distancia de 1200 Km, CloudNet ahorra 20GB de ancho de banda, una reducción del 50 %.

2.3.5. Balanceo de Carga

La utilización baja de los servidores incrementa costos por consumo de energía y sistemas de enfriamiento. Además, más máquinas físicas requieren más espacio físico e incrementa la mano de obra en mantenimiento y administración. Actualmente, la computación verde contribuye a desarrollar tecnologías que reducen el consumo de energía y las emisiones de CO2 de los sistemas de enfriamiento. La virtualización pertenece a esas tecnologías y proporciona mecanismos para optimizar la asignación de recursos y también la capa de aislamiento que consolida las aplicaciones en ejecución en varios servidores subutilizados en un número reducido de servidores altamente utilizados. Además, es posible migrar recursos desde una máquina física a otra con mínima disrupción. Existen varias propuestas que permiten la reasignación de recursos para obtener un balanceo de cargas y la distribución de recursos de una manera más óptima. En [Bobroff07] se introduce la migración dinámica de servidores junto con un algoritmo de consolidación. El objetivo del algoritmo es minimizar el costo ejecución de un centro de datos. El costo penaliza la sobrecapacidad (la utilización baja) y la sobrecarga, la cual causa que el desempeño de la aplicación sea pobre y viola los acuerdos de nivel de servicio (*Service Level Agreements*, SLAs). Los SLAs son típicamente expresados como garantías CPU o tiempo de respuesta. El algoritmo propuesto está basado en la medida de datos históricos, la previsión de demandas futuras, y la reasignación de VMs a máquinas físicas (physical machines PMs) y es llamado Medida-Previsión-Reasignación (*Measure-Forecast-Remap*, MFR). Este método considera el análisis y clasificación de recursos de la carga de trabajo. Las series de tiempo de las demandas de recursos fueron analizadas con una técnica de previsión y se encontró que los servidores que se benefician más con la migración dinámica son aquellos que denotan una fuerte variabilidad y autocorrelación en sus distribuciones de recursos. Después de averiguar las VMs candidatas para manejo dinámico y su demanda de recursos futura (para un intervalo de tiempo T), el algo-

ritmo de manejo trata de reubicar cada VM candidata a PM tales que los requerimientos de los recursos de las máquinas virtuales deben ser menores que la capacidad de las máquinas físicas destino.

Manejo automático de la migración en vivo

La migración en vivo automática puede ser causada por la detección de "puntos calientes" (*hotspot*). Un hotspot puede ser definido como un punto de acceso con alta demanda de recursos, tales como un servidor sobrecargado, si algunos recursos como el procesador, la red o la memoria exceden un umbral o si son ejecutadas algunas violaciones de los acuerdos de nivel de servicios. Una característica importante de un centro de datos bien administrado es su capacidad de evitar hotspots. Los nodos sobrecargados (servidores, almacenamiento o conmutadores de red) con frecuencia llevan a una degradación de rendimiento y son vulnerables a las fallas. Para aliviar los hotspots, la carga podría ser migrada desde el recurso sobrecargado hacia uno subutilizado. En [Wood07] se presenta un mecanismo para migrar automáticamente máquinas virtuales. El sistema es llamado Sandpiper e implementa un algoritmo de detección de hotspot que determina cuando, cuáles y a donde migrar las máquinas virtuales. Sandpiper usa el algoritmo de migración de Xen [Clark05]. El mecanismo de VirtualPower [Nathuji07] opera en una plataforma virtualizada. Este hace posible controlar y coordinar la aplicación de varias políticas de manejo de energía dentro de VMs. Se limita el uso del hardware por huésped a través de técnicas de software y hardware por medio del soporte de hardware subyacente como el escalamiento de frecuencia del procesador.

2.3.6. Grabar/Reproducir para Replicación

Conceptos importantes

Grabar/Reproducir es una técnica que es capaz de mantener el estado completo de una VM en más de una ubicación de manera simultánea.

Remus

Remus [Cully08] migra VMs en ejecución entre servidores físicos y replica el estado entero de una instancia de SO en ejecución a altas frecuencias (tan frecuentemente como cada 25 ms) entre un par de máquinas físicas. Remus se ejecuta en servidores emparejados en una configuración activo-pasiva. Remus utiliza tres técnicas en su implementación: Primera, la virtualización es usada para ejecutar un par de sistemas en sincronía, los eventos externos son inyectados cuidadosamente en ambas máquinas, la primaria y el respaldo, las cuales tienen estados idénticos. Segunda, Remus no intenta hacer computación determinística. El estado de la réplica necesita ser sincronizado con la VM primaria solamente cuando la salida de la primaria es externamente visible. En vez de dejar que el flujo de salida normal dicte cuando debe ocurrir la sincronización, Remus guarda información en un buffer de salida (red y disco) hasta un momento más conveniente. Y tercera, se implementa una replicación asíncrona; guardar la información de salida del servidor primario permite que la replicación se ejecute asincrónicamente. El host primario puede reasumir la ejecución en el momento en que el estado de la máquina ha sido capturado, sin esperar por el reconocimiento del extremo remoto. El servidor primario permanece productivo, mientras que la sincronización con servidor replica se ejecuta asincrónicamente. El mecanismo de alta disponibilidad de Remus está basado en checkpoints frecuentes de la VM activa a una VM de respaldo. La VM respaldo está residente en memoria y podría entrar en ejecución inmediatamente, si el sistema activo sufre una falla. La VM de respaldo no es siempre consistente con la VM primaria, la salida de red es grabada a un buffer hasta que el estado es sincronizado en el respaldo. El ciclo que incluye checkpoint, buffer y liberación del buffer ocurre frecuentemente (hasta 40 veces por segundo). La replicación de disco se mantiene asincrónicamente a través de escrituras al respaldo. Las escrituras a disco primario son contenidas en un buffer RAM hasta que el checkpoint correspondiente llega. Después de esto, el checkpoint es reconocido por el servidor primario, el cual libera el tráfico de red saliente y las escrituras a disco dentro de un buffer son aplicadas al servidor de respaldo. Pueden hacerse múltiples respaldos de manera simultánea. Remus implementa un modelo de falla que considera el manejo de una falla en cualquier host. Si el host primario y de respaldo fallan al mismo tiempo, Remus

dejará el sistema en un estado consistente. Así, las salidas solo pueden ser visibles hasta que el estado del sistema asociado a ha sido reconocido (commit) por la réplica. Se ha implemntado un detector de fallas dentro de la estrategia de checkpoint. Si se agota el tiempo de espera mientras que el servidor primario está esperando los requerimientos de reconocimiento (commit), el servidor primario asume que la VM de respaldo ha colapsado y deshabilitará la protección. Por otro lado, si el tiempo de espera se agota durante un checkpoint, el respaldo asumirá que el primario ha colapsado y reasumirá la ejecución desde el checkpoint más reciente. En la evaluación de rendimiento [Cully08], se encontró que en una tarea de propósito general, Remus incurre en una penalización de rendimiento del 50 % cuando realiza un checkpoint de 20 veces por segundo. Cully et al. aplicaron varias pruebas a Remus, dentro de las cuales pueden ser mencionado que: La prueba de compilación del núcleo mide el tiempo requerido para construir una versión de Linux 2.6.18 usando la configuración por omisión y un destino bzImage. Esta es una prueba para CPU, memoria y rendimiento de disco. El checkpoint fue configurado para velocidades de 10, 20, 30 y 40 veces por segundo, comparado contra una base de compilación de una máquina virtual no protegida. El costo medido de cada una de estas frecuencias fue de 31 %, 52 %, 80 % y 103 % respectivamente.

Migración en vivo ligera

La alta disponibilidad es una característica apreciada para conglomerados y computación en red y algunas veces más apreciada que el rendimiento. La migración en vivo de máquinas virtuales es un mecanismo para lograr alta disponibilidad. Pero el uso de checkpoint que es la operación que forza a que los cambios en los datos que están registrados en memoria, sean escritos a disco, puede introducir un costo significativo. En el trabajo [Jiang10], se presenta una migración en vivo ligera (Lightweight Live Migration, LLM). Este trabajo es desarrollado sobre Xen y comparado con Remus [Cully08]. Los casos de prueba presentan que LLM supera a Remus en términos de retraso de red y costo de red. En este trabajo, la máquina que proporciona los servicios regulares es llamada la máquina primaria y la máquina réplica es llamada máquina de respaldo. LLM integra dos ideas: checkpointing y la reproducción de entrada con el objetivo de mejorar la migración para

aplicaciones con carga de red intensiva. La máquina primaria migra las actualizaciones de estado de CPU, memoria y disco a la máquina respaldo a baja frecuencia y los servicios requeridos por los clientes de red a alta frecuencia. En este trabajo, el modelo de falla de red considera las siguientes suposiciones: Cualquier servidor puede detectar si otro servidor ha fallado y se usa un almacenamiento estable que contiene el último estado correcto del servidor colapsado. Remus usa el caso block/commit donde los paquetes de salida son bloqueados hasta que un checkpoint es reconocido. LLM libera los paquetes de respuesta inmediatamente. Durante una transferencia con checkpoint en baja frecuencia, los clientes de red podrían experimentar largas esperas con el mecanismo block/commit de Remus. La secuencia para migrar los recursos de checkpoint se describe a continuación:

- La máquina virtual se pausa. Durante este periodo de suspensión, todas las actualizaciones del estado de CPU/memoria/disco son recolectadas y almacenadas en un buffer de migración.

- Una vez que la VM huésped es reasumida, el contenido almacenado en el buffer de migración es migrado primeramente a la máquina réplica.

- Después, la migración del buffer de red comienza a alta frecuencia hasta que la VM huésped es suspendida otra vez. Al final de cada ciclo de migración de buffer de red, se conoce cuales paquetes necesitan ser reproducidos en la máquina respaldo y cuales necesitan ser respondidos a los clientes.

En cualquier momento que ocurre una falla a la máquina primaria, la máquina de respaldo continuará la ejecución de la VM desde el último estado de checkpoint. Entonces, esta máquina reproducirá los requerimientos después del primer límite para lograr consistencia y responderá a aquellas peticiones no respondidas después del segundo límite. LLM fue evaluado y comparado contra Remus en términos de exactitud, tiempo de inactividad, espera de clientes y costo de varios periodos de checkpoint [Jiang10]; LLM probó algunos casos de estudio. Por ejemplo, se lanza un ping flood sobre la máquina primaria. Este experimento es llamado “HighNet” por su intensidad de carga de red. Con HighNet, LLM obtiene tiempos de inactividad más cortos que Remus. Esto es debido a que hay muchos

paquetes duplicados por ser servidos por la máquina de respaldo en Remus desde el lado del cliente. En este experimento, también se estimó el retraso de red. Los resultados mostraron que la mayoría del tiempo, LLM tiene retrasos de red más cortos que Remus.

2.4. Conclusiones

En este capítulo se presentó una descripción de las ventajas y desventajas de los mecanismos de migración y de replicación. Además se realizó un presentación detallada de diversos trabajos publicados en el área de migración de servicios TCP/IP. También se describieron varios trabajos relacionados con el área de migración/replicación de máquinas virtuales y su clasificación dentro según el recurso migrado y/o medio de migración; identificándose la migración en vivo, memoria, migración suspender/reasumir, migración através de WAN, migración para balanceo de carga, mecanismo grabar/reproducir. Se realiza una descripción detallada con el ánimo de introducir al lector en las áreas de virtualización, migración de servicios TCP/IP y migración de máquinas virtuales, dándole un panorama general y pueda reconocer las áreas de oportunidad que aún se encuentran abiertas a la investigación.

Capítulo 3

Xen

3.1. Diseño de Xen

Xen es un Monitor de Máquinas Virtuales paravirtualizado de código abierto, desarrollado por la Universidad de Cambridge [Xen.org11].

Una idea clave en un buen diseño de sistema es la separación de política y mecanismo. El hipervisor implementa mecanismos, pero deja la política hasta el dominio 0 del huésped [Chisnall07].

Xen no soporta ningún dispositivo de manera nativa. En vez de esto, proporciona un mecanismo por el cual un sistema operativo huésped puede dar acceso directo a un dispositivo físico. Entonces, el sistema operativo huésped puede usar un manejador de dispositivo existente.

También se necesita que exista una manera de proporcionar acceso al dispositivo a más de un huésped. Xen proporciona solamente un mecanismo. La interfaz de tabla de derechos permite a los desarrolladores otorgar acceso a las páginas de memoria de otros huéspedes, en mucho de la misma forma como la memoria compartida de POSIX, siempre que el XenStore proporcione una jeraquía como un sistema de archivos (completa con control de acceso) que pueda ser usado para implementar el descubrimiento de las páginas compartidas.

El hipervisor Xen solamente implementa estos mecanismos, pero los huéspedes

requieren cooperar si desean usarlos; si un dispositivo advierte su presencia en una parte del árbol XenStore, otros huéspedes deben saber para mirar allí si desean encontrar un dispositivo de este tipo. Como tal, existen varias convenciones y algunos mecanismos de alto nivel, tales como los buffers de anillo, que son usados para los requerimientos de paso y respuestas entre dominios para soportar E/S. Estos son definidos por especificaciones y documentación, sin embargo, no es forzoso en el código, lo que hace a Xen un sistema muy flexible.

En contraste con la mayoría de los paquetes de software, cada nueva versión de Xen intenta hacer menos que la versión previa. La razón para esto es que Xen se ejecuta a un nivel alto de privilegios, incluso por arriba del sistema operativo. Un defecto en un programa podría comprometer los datos que accede, un defecto en el kernel podría comprometer el sistema entero, pero un defecto en Xen podría comprometer cada máquina virtual ejecutándose en la máquina. Por esta razón, es importante que el código de Xen sea seguro y lo más libre de defectos que sea posible.

Para mantener flexibilidad, Xen no forza mecanismos para comunicación entre dominios. En vez de esto, proporciona mecanismos simples, tales como memoria compartida y permite a los sistemas operativos huéspedes usar esto como requieran. Esto significa que agregar soporte a una categoría de dispositivo no requiere modificar Xen.

Las últimas versiones de Xen hacen más en el hipervisor. La multiplexación de red, por ejemplo, fue parte de Xen 1.0, pero después se movió al Dominio 0. La mayoría de los sistemas operativos ya incluyen características muy flexibles para puenteo y tunelling de interfaces virtuales de red, así que tiene más sentido usar estas que implementar unas nuevas.

3.2. La arquitectura de Xen

Xen se establece entre el SO y el hardware, y proporciona un ambiente virtual en el cual un kernel puede ejecutarse. Los tres componentes principales de cualquier sistema relacionado con Xen son el hipervisor, el kernel y las aplicaciones de espacio de usuario.

3.2.1. El hipervisor, el SO y las aplicaciones

El objetivo principal de los anillos de protección es la organización jerárquica del sistema operativo, mediante capas o anillos concéntricos, cada una de las capas esta construida sobre la anterior. Los sistemas operativos proporcionan diferentes niveles de acceso a los recursos. Un anillo de protección es un nivel jerárquico o capa de privilegios dentro de la arquitectura de un sistema de computación. Esto es generalmente impuesto por el hardware por algunas arquitecturas de CPU que ofrecen diferentes modos de CPU en el hardware o a nivel de microcódigo. Los anillos están dispuestos en una jerarquía desde los más privilegiados (de más confianza), usualmente numerado cero, hasta el menos privilegiado (de menos confianza), usualmente con el mayor número de anillo. En la mayoría de sistemas operativos, el anillo 0 es el nivel con la mayoría de los privilegios e interactúa más directamente con el hardware físico, como el CPU y la memoria [Karger84].

Uno de los cambios mayores para un kernel ejecutándose bajo Xen es que ha sido movido del anillo 0. El anillo dentro del cual se ejecuta varía de plataforma a plataforma. En los sistemas IA32, es movido al anillo 1, como se muestra en la Figura 3.1. Esto le permite acceder la memoria asignada a las aplicaciones que se ejecutan en el anillo 3, pero protegido de aplicaciones y de otros kernels. El hipervisor, en anillo 0, es protegido de los kernels en anillo 1 y aplicaciones en anillo 3.

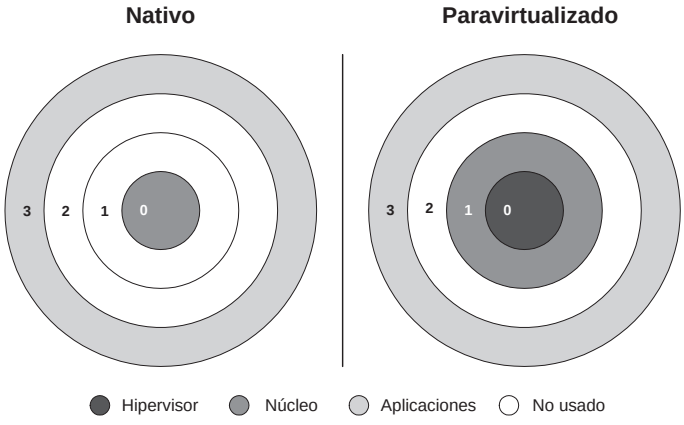


Figura 3.1: Uso de los anillos en sistemas nativos y paravirtualizados

Cuando AMD creó x86-64 como parte de la arquitectura IA32, se redujo el número

de anillos. Con la excepción de OS/2 y (opcionalmente) NetWare, ninguno hacía mucho uso de los anillos 1 y 2, así que fueron omitidos. Desafortunadamente, la comunidad de virtualización fue entre las más afectadas.

En ausencia de anillos 1 y 2, fue necesario modificar Xen para colocar el sistema operativo en el anillo 3, junto con las aplicaciones. La Figura 3.2 muestra la diferencia entre los dos métodos. Xen toma este método en otras plataformas, tales como IA64, la cual solamente tiene dos anillos de protección. x86-64, también removió la protección de memoria basada en segmento. Esto significa que Xen tiene que confiar en los mecanismos de protección de paginación para aislarse a sí mismo de los huéspedes. Desde la perspectiva de

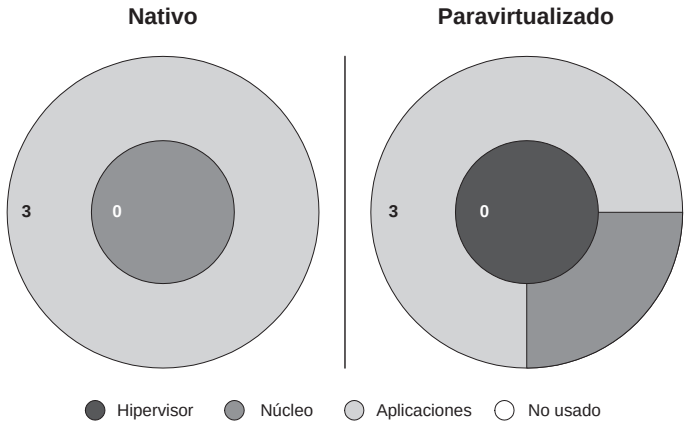


Figura 3.2: Uso de los anillos en sistemas x86-64 nativos y paravirtualizados

un kernel paravirtualizado, hay bastantes diferencias entre ejecutarse en Xen y ejecutarse en el hardware. La primera es el modo de CPU al tiempo de inicio. Todos los procesadores x86 desde el 8086 han iniciado en modo real. Para el 8086 y 8088, este fue el único modo disponible; un modo de 16-bits con acceso a un espacio de direcciones de 20-bits y no manejo de memoria. Debido a que se esperó que todas las máquinas subsecuentes x86 fueran capaces de ejecutar software heredado, incluyendo los sistemas operativos, todas las PCs compatibles con IBM han iniciado con el CPU en modo real. Una de las primeras tareas para un sistema operativo moderno es cambiar el CPU a un modo protegido, el cual proporciona algunas facilidades para aislar estados de procesos de memoria y permitir la ejecución de instrucciones de 32-bits. Debido a que Xen es responsable del inicio del sistema,

el mismo ejecuta esta transición. Si no lo hace, no sería capaz de aislarse a sí mismo de la interferencia de los sistemas operativos huéspedes. Esto significa que el kernel huésped inicia en un ambiente bastante diferente. Los nuevos sistemas x86 vienen con Extended Firmware Interface (EFI), el cual es reemplazado por el antiguo PC BIOS. Cualquier sistema con EFI puede iniciar en modo protegido, aunque la mayoría tiende a reusar el código de inicio anterior y requiere que sea cargado un modulo de compatibilidad BIOS-EFI.

El siguiente cambio es que las instrucciones privilegiadas deben ser reemplazadas con hiperllamadas. Otro cambio es como la conservación del tiempo es manejada. Un sistema operativo necesita conservar la pista del tiempo de dos formas: Necesita conocer la cantidad de tiempo actual que ha transcurrido y la cantidad de tiempo CPU. La primera es requerida para la interfaz de usuario, así se le da al usuario un reloj real para despliegue y para programas tales como cron y para sincronizar eventos a través de la red. El segundo es requerido para multitarea; cada proceso debe obtener una parte justa del CPU.

Cuando se esta ejecutando fuera del hipervisor, el tiempo real y el tiempo de CPU son la misma cosa. Todo lo que el kernel tiene que hacer es conservar la pista de que tanto tiempo asigna a los procesos y a sus propios hilos. Cuando se está ejecutando en Xen, sin embargo, tiene que compartir los CPUs disponibles con otros sistemas operativos. Esto probablemente significa que solamente recibirá alguna porción de segundo de tiempo de CPU por cada segundo de tiempo real. Como tal, debe resincronizar continuamente su reloj interno con las facilidades de cronometraje proporcionadas por Xen.

3.2.2. El papel del Dominio 0

El propósito de un hipervisor es permitir a los huéspedes que se ejecuten. Xen ejecuta los huéspedes en ambientes conocidos como dominios, los cuales encapsulan un ambiente virtual completo ejecutándose. Cuando Xen se inicia, una de las primeras acciones que hace es cargar el kernel huésped Dominio 0 (dom0). Esto es típicamente especificado en el cargador de inicio como un módulo, y puede ser cargado sin que ningún manejador de archivos de sistema esté disponible. El dominio 0 es el primer huésped en ejecutarse, y tiene privilegios elevados. En contraste, otros dominios son referenciados como Dominio

U (domU), "U" de no privilegiados (*unprivileged*). Sin embargo, ahora es posible delegar algunas de las responsabilidades del dom0 a los huéspedes U.

El dominio 0 es muy importante para un sistema Xen. Xen no incluye manejadores de dispositivos, ni interfaces de usuario. Todos estos son proporcionados por el sistema operativo y las herramientas de espacio de usuario ejecutándose en el huésped dom0. El huésped Dominio 0 es típicamente Linux, aunque NetBSD y Solaris puede también ser usado y probablemente otros sistemas tales como FreeBSD sean agregados en el futuro. Linux es usado por la mayoría de los desarrolladores de Xen, y ambos son distribuidos bajo las mismas condiciones, la Licencia Pública General (en inglés, *General Public License*, GNU).

Una tarea de dom0 es manejar dispositivos. Este huésped ser ejecuta a más alto nivel de privilegios que los otros, así que puede acceder el hardware. Por esta razón, es vital que los huéspedes privilegiados sean asegurados de manera apropiada.

Parte de la responsabilidad de manejar dispositivos es el multiplexamiento de ellos para las máquinas virtuales. Debido a que la mayoría del hardware no soporta nativamente ser accedido por múltiples sistemas operativos, es necesario para alguna parte del sistema proporcionar a cada huésped su propio dispositivo virtual.

La Figura 3.3 presenta que le ocurre a un paquete cuando es enviado por una aplicación ejecutada en un dominio huésped domU. Primero, viaja a través de la pila TCP/IP como lo haría de manera normal. El final de la pila, sin embargo, no es un manejador de interfaz de red normal. Es una pieza de código que coloca el paquete dentro de la memoria compartida. El segmento de memoria ha sido compartido previamente usando las tablas de derechos de Xen y anunciado vía XenStore. La otra mitad de manejador de dispositivo dividido, ejecutándose en el huésped dom0, lee el paquete del buffer y lo inserta en los componentes de firewall del sistema operativo, típicamente algo como iptables o pf, el cual lo rutea como si fuera un paquete entrante desde la interfaz real. Una vez que el paquete ha pasado a través de las reglas del firewall, éste toma su camino al manejador de dispositivo real. Este es capaz de escribir a ciertas áreas de memoria reservadas para E/S, y podría requerir acceso a IRQs vía Xen. Entonces, el dispositivo de red físico envía el paquete.

Notar que el dispositivo de red dividido aquí es el mismo independientemente de

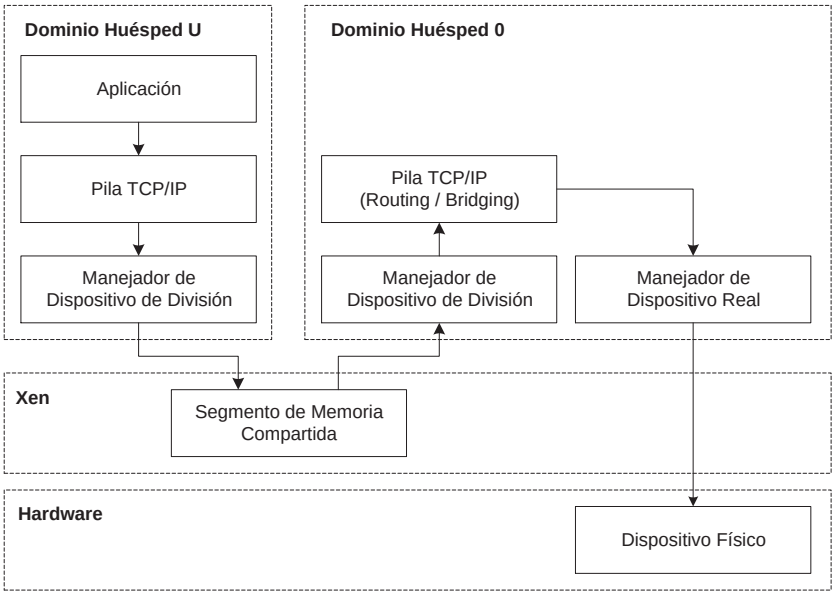


Figura 3.3: El camino de un paquete que se envía desde un huésped no provilegiado a través del sistema

la tarjeta de red real. Xen proporciona una interfaz simplificada de estos dispositivos, la cual es fácil de implementar para gente portando sistemas a Xen. Hay tres componentes de cualquier driver:

1. El manejador dividido
2. El multiplexor
3. El manejador real

El manejador dividido es típicamente tan simple como puede ser. Está diseñado para mover datos de los huéspedes domU al huésped dom0, usualmente usando buffers de anillo en memoria compartida.

El dispositivo real debería existir en el sistema operativo dom0, y así, no puede ser considerado parte de Xen. El multiplexor podría o no podría. En el ejemplo de red, los componentes de firewall de la pila de red ya proporcionan esta funcionalidad. En otros, podría no existir un componente del sistema operativo que pueda ser usado.

El huésped dom0 es también responsable de manejar tareas administrativas. Mientras que Xen por sí mismo crea nuevos huéspedes domU, lo hace en respuesta a una hiper-

llamada desde el huésped dom0. Esto es típicamente hecho vía un conjunto de herramientas Python (*scripts*) que manejan toda la política relacionada a la creación de huéspedes y emiten las hiperllamadas relevantes.

El Dominio 0 proporciona una interfaz de usuario al hipervisor. Los dos demonios xend y xenstored ejecutándose en este dominio proveen características importantes para el sistema. El primero es responsable de proporcionar una interfaz administrativa al hipervisor, permitiendo al usuario definir política. El segundo proporciona almacenamiento back-end para el XenStore.

3.2.3. Dominios No Privilegiados

Un dominio no privilegiado (domU) está más restringido. A un huésped domU típicamente no se le permite ejecutar ninguna hiperllamada que acceda directamente al hardware, aunque en algunas situaciones le podrían ser otorgados accesos a uno o más dispositivos.

En vez de acceder directamente al hardware, un huésped domU típicamente implementa el front end de algunos manejadores de dispositivo dividido. En un mínimo, es probable que necesite de XenStore y de los manejadores de dispositivo de consola. Debido a que estos son genéricos, abstractos, dispositivos, un huésped de dominio U solamente necesita implementar un manejador por cada categoría de dispositivo. Por esta razón, ha habido varios sistemas operativos portados para ejecutarse como huéspedes Xen dominio U, los cuales tienen soporte relativamente pobre de hardware ejecutándose directamente en hardware real. Xen permite a estos huéspedes tomar ventaja del soporte para hardware del huésped de Dominio 0.

A diferencia de los huéspedes dom0, se puede tener un número arbitrario de huéspedes domU en una sola máquina, y éstos pueden ser migrados. Si la migración es posible depende en gran medida de la configuración; los huéspedes configurados con acoplamiento fuerte al hardware no pueden ser movidos. Un cliente que usa algo como NFS o Isasi para sus requerimientos de almacenamiento puede ser migrado en vivo, mientras que uno que usa un manejador de dispositivo de bloques debe ser suspendido a un dispositivo flash

y movido a una máquina diferente.

Por razones de seguridad, es recomendable hacer lo menos posible en el Dominio 0. Un exploit root en Dominio 0 podría comprometer potencialmente al sistema entero. Como tal, la mayoría del trabajo debería ser hecho en los huéspedes paravirtualizados de dominio U o en huéspedes HVM.

La línea entre el dom0 y domU es algunas veces tenue. Es posible permitir a los huéspedes domU acceder directamente a algún hardware, e incluso a los manejadores de dispositivo divididos del host. Por ejemplo, una laptop podría usar Linux como el huésped dom0, pero ejecutar una VM NetBSD domU para soportar una tarjeta particular WiFi. En plataformas sin un IOMMU, hacer esto puede comprometer la seguridad, porque permite al huésped domU acceder al espacio de direcciones completo.

3.2.4. Dominios HVM

Cuando Xen fue creado, la arquitectura x86 no reunió los requerimientos de Popek and Goldberg para virtualización. Un monitor de máquina virtual para x86 necesitaba emular la arquitectura, aunque podría hacerlo muy rápidamente para un gran subconjunto de instrucciones. Xen implementó paravirtualización para evitar este problema.

Los chips más recientes x86 no sufrieron esta limitación, y así hace sentido extender Xen para soportar los huéspedes no modificados. Las versiones más recientes de Xen, permiten que sean ejecutados huéspedes de Máquina Virtual de Hardware (en inglés, *Hardware Virtual Machine*, HVM). Ejecutar huéspedes no modificados en Xen requiere soporte de hardware, lo cual no es una opción para máquinas viejas. Algunos sistemas adquiridos en 2007 o después deberían soportar HVM y algunas de 2006.

Los huéspedes HVM difieren de los huéspedes paravirtualizados en varios sentidos. Esto es evidente desde el tiempo de autoarranque. Un huésped paravirtualizado inicia en modo protegido, con algunas páginas de memoria que contienen información de autoarranque mapeada por el hipervisor, mientras un huésped HVM comienza en modo real y obtiene la información de configuración de un BIOS emulado.

Si un huésped HVM quiere tomar ventaja de las características específicas de Xen,

necesita usar la instrucción CPUID para acceder a un registro específico de la máquina virtual y acceder la página de hiperllamadas. Entonces puede emitir hiperllamadas de la misma manera que un huésped paravirtualizado, llamando un desplazamiento en la página de hiperllamadas. Esto entonces usa la instrucción correcta (por ejemplo, VMCALL) para una transición rápida al hipervisor.

3.3. La virtualización de los recursos en Xen

Se presenta una descripción de la interfaz x86 paravirtualizada, tomando en cuenta tres aspectos principales del sistema: Manejo de memoria, el CPU, y dispositivos de E/S. Notar que aunque ciertos aspectos de implementación tales como manejo de memoria, son específicos de x86, muchos aspectos (CPU y dispositivos de E/S) pueden ser fácilmente aplicados a otras arquitecturas de máquina. Además, x86 representa el peor caso donde difiere significativamente de los procesadores de estilo RISC, por ejemplo, paravirtualizar tablas de páginas de hardware es más difícil que virtualizar TLB (en inglés, *Translation Lookaside Buffer*) manejadas por software [Barham03].

3.3.1. Manejo de memoria

Dentro de una arquitectura, una tarea complicada es la virtualización de la memoria, en términos de los mecanismos requeridos en el hipervisor y las modificaciones requeridas para portar cada SO huésped. La tarea es más fácil si las arquitecturas proporcionan un TLB manejado por software.

Desafortunadamente, x86 no tiene un TLB manejada por software; TLB es manejado automáticamente por el procesador recorriendo la estructura de tabla de páginas en hardware. Así para lograr el mejor desempeño posible, todas las traducciones de página válidas para el espacio de direcciones actual deberían estar presentes en la tabla de páginas accesible por hardware. Además, debido a que TLB no está etiquetado, los switches de espacio de direcciones típicamente requieren un vaciado completo de TLB. Dado estas limitaciones, se tomaron dos decisiones: 1) los SOs huéspedes son responsables de asignar y

manejar las tablas de páginas de hardware, con participación mínima de Xen para asegurar aislamiento y seguridad; y 2) Xen existe en una sección de 64MB encima del todo espacio de direcciones, evitando un vaciado TLB cuando se entra o se deja el hipervisor.

Cada vez que un SO huésped requiere de una tabla de páginas nueva, quizá porque un proceso nuevo está siendo creado, asigna e inicializa una página desde su propia reservación de memoria y la registra con Xen. En este punto el SO debe renunciar a los privilegios de escritura directa a la memoria de tabla de páginas: todas las actualizaciones siguientes deben ser validadas por Xen. Esto restringe las actualizaciones en varios sentidos, incluyendo solamente permitir a un SO mapear páginas que le pertenecen y deshabilitando mapeos de escritura de las tablas de páginas. Los SO huéspedes podrían requerir actualizaciones por lote para amortizar el costo de entrar al hipervisor. La región más alta de 64MB de cada espacio de direcciones, la cual esta reservada para Xen, no es accesible o maleable por los SOs huéspedes. Esta región de direcciones no es usada por ninguna de las ABIs x86 comunes, sin embargo, esta restricción no rompe compatibilidad de aplicación.

3.3.2. CPU

Virtualizar el CPU tiene varias implicaciones para los SOs huéspedes. Principalmente, la inserción de un hipervisor bajo el sistema operativo viola el estatuto de que el SO es la entidad más privilegiada del sistema. Para proteger el hipervisor de un mal comportamiento de SO (y dominios de uno a otro), los SOs huéspedes deben ser modificados para ejecutarse a un nivel de privilegios más bajo.

Muchas arquitecturas de procesador solamente proporcionan dos niveles de privilegio. En estos casos el SO huésped debería protegerse a sí mismo ejecutándose en un espacio de direcciones separado de las aplicaciones vía el hipervisor para colocar el nivel de privilegio virtual y cambiar el espacio de direcciones actual. Si el TLB del procesador soporta etiquetas de espacio de direcciones, entonces los costosos vaciados de TLB pueden evitarse.

La virtualización eficiente de niveles de privilegios es posible en x86 debido a que soporta cuatro niveles de privilegio diferentes en hardware. Los niveles de privilegio x86 son generalmente descritos como anillos, y son numerados de cero (el más privilegiado) a tres

(el menos privilegiado). El código del sistema operativo típicamente se ejecuta en el anillo 0 porque ningún otro anillo puede ejecutar instrucciones privilegiadas, mientras que el anillo 3 es generalmente usado para código de aplicación. Los anillos 1 y 2 no han sido usados por algún SO x86 conocido desde OS/2. Cualquier SO que sigue este arreglo común puede ser portado a Xen modificándolo para ejecutarse en el anillo 1. Esto previene al SO huésped de ejecutar instrucciones privilegiadas directamente, permanece aislado seguramente de las aplicaciones ejecutándose en el anillo 3.

Las instrucciones privilegiadas son paravirtualizadas, requiriéndoles que sean validadas y ejecutadas dentro de Xen, esto aplica a operaciones tales como instalar una tabla de páginas nueva o ceder el procesador cuando esta ocioso (en vez de intentar un hlt). Cualquier SO huésped que intenta ejecutar una instrucción privilegiada directamente es rechazado por el procesador, bien de manera silenciosa o con una falla.

3.3.3. Dispositivos E/S

En vez de emular los dispositivos existentes de hardware, como se hace típicamente en los ambientes de virtualización completa, Xen expone un conjunto de abstracciones limpias y simples. Esto permite diseñar una interface que es eficiente y satisface los requerimientos de protección y aislamiento. Para este fin, la E/S de datos es transferida a y desde cada dominio vía Xen, usando la memoria compartida y anillos descriptores de buffer asíncronos. Estos proporcionan un mecanismo de comunicación de alto rendimiento para el paso de información de buffer verticalmente a través del sistema, permitiendo a Xen realizar verificaciones de validación de rendimiento eficientemente (por ejemplo, verificar que los buffers están contenidos dentro de una reservación de memoria del dominio).

De manera similar a las interrupciones de hardware, Xen soporta un mecanismo de entrega de eventos de peso ligero el cual es usado para envío de notificaciones asíncronas al dominio. Estas notificaciones son hechas actualizando un mapa de bits de tipos de eventos pendientes y opcionalmente, llamando un manejador de evento especificado por el SO huésped.

3.4. Conclusiones

En este capítulo se presentó la arquitectura del VMM usado en esta tesis, Xen. Se describió el uso de los anillos de los sistemas operativos tanto en modo nativo como en modo paravitualizado y los diferentes niveles de acceso a los recursos que proporcionan dichos anillos. Se describe al Dominio 0 de un núcleo modificado de Linux como la única máquina virtual en ejecución sobre Xen que tiene derechos especiales de acceder a los recursos físicos de Entrada/Salida y de interactuar con otras máquinas virtuales en ejecución en el sistema (Dominio U). Todos los ambientes de virtualización Xen requieren un Dominio 0 en ejecución antes de que cualquier otra máquina virtual pueda ser iniciada. El Dominio 0 contiene herramientas para el control de las MVs.

Se describe la virtualización de los recursos en Xen en una arquitectura x86, tomando en cuenta tres aspectos del sistema; manejo de memoria, el CPU y los dispositivos de E/S. Se explica de manera general como son atendidas las solicitudes de manejo de memoria, CPU y dispositivos de E/S de los dominios no privilegiados.

Capítulo 4

Propuesta de Diseño e Implementación de un Sistema de Replicación de Servicios TCP/IP

4.1. Descripción del protocolo de replicación de servicios TCP/IP propuesto

Esta contribución propone el diseño e implementación de un protocolo de replicación de máquinas virtuales basado en TCP/IP. El cual tiene la capacidad de trabajar tanto en un ambiente LAN como WAN proporcionando alta disponibilidad con un nivel de desempeño satisfactorio para el usuario. En las siguientes subsecciones se describe la operación de este protocolo, como se ejecuta la inyección de paquetes y se explica de manera detallada su implementación.

4.1.1. Esquema del protocolo

Este protocolo está basado en la idea de que una VM puede propagar su estado completo hacia una VM réplica, si se aplican los mismos comandos TCP. Para la siguiente explicación se contempla un escenario de tres computadoras, donde se contempla solo una réplica. El número de réplicas puede incrementar, pero para la explicación solo se considera

una.

Una máquina llamada Máquina Real (*Real Machine*, RM) intercambia datos con una VM llamada Original (*Original Virtual Machine*, OVM). La RM puede modificar el estado de la OVM a través de comandos remotos. Por ejemplo, la RM puede crear, borrar o modificar archivos en la OVM a través de comandos remotos. RM puede instalar aplicaciones en OVM; puede habilitar un servicio en OVM, etc. Otra máquina llamada Máquina Virtual Replica (*Replica Virtual Machine*, RVM) se crea en una ubicación física diferente con el mismo estado que la OVM. Cada comando que tiene una acción sobre la OVM se replica sobre la RVM. El comando se ejecuta sobre la RVM de manera transparente, sin requerir la atención del usuario, como si la RM trabajara directamente con la RVM. En la Figura 4.1, se presenta un esquema general del protocolo.

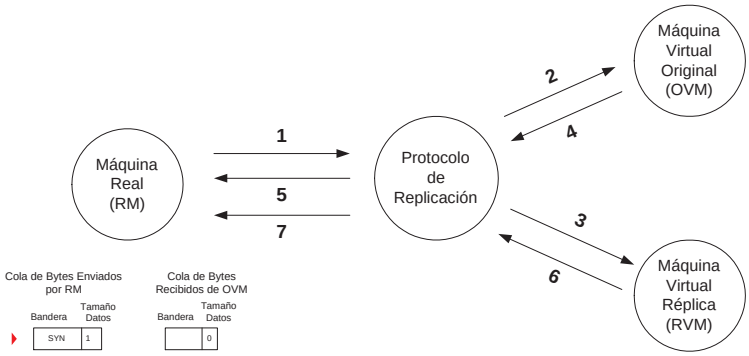


Figura 4.1: Esquema de replicación

El protocolo que se propone puede describirse la manera siguiente:

1. La *RM* lanza un comando remoto, el cual tiene como IP fuente la dirección IP de *RM* y como destino la dirección IP de *OVM*.

2. Si el programa intermedio replicador detecta que un paquete TCP tiene como destino a *OVM*, se atrapa el paquete, se duplica y se envía a una cola. A estos paquetes se les cambiará la dirección IP destino por la dirección IP de RVM. Los paquetes que son atrapados se forman ordenadamente en una cola, de acuerdo al lugar que el paquete tiene en la conversación entre *RM* y *OVM*. El paquete espera en la cola hasta que se su turno de ser enviado, en el lugar correcto como parte de una nueva conversación

entre *RM* y *RVM*.

3. Se extrae el primer paquete de la cola. Este paquete sufre algunas modificaciones en los siguientes campos: la IP destino de la cabecera IP, el número de reconocimiento (ACK) de la cabecera TCP, el subcampo TSVal y Tsecr del campo opciones TCP, y el campo *checksum* TCP es recalculado. Después, se envía el paquete modificado a *RVM* a través de una VPN.

4. La *OVM* regresa una respuesta a *RM* después de la recepción del paquete. Este evento es registrado en una ventana de deslizamiento.

5. *RM* recibe el paquete enviado por *OVM*.

6. La *RVM* envía un paquete de respuesta, lo cual causa la modificación de los registros de la correspondiente ventana de deslizamiento.

7. *RM* captura la respuesta de *RVM*, pero el paquete es desechado para evitar que la conversación entre *RM* y *RVM* sea invalidada.

El protocolo aprovecha los mecanismos de recuperación del protocolo TCP/IP para establecer una comunicación confiable entre servidores. TCP [Institute81] fue seleccionado como base de este protocolo debido a que proporciona la facilidad de recuperación de datos corruptos, pérdida de datos, detecta datos duplicados y fuera de orden. En nuestra implementación, se considera el mecanismo de confiabilidad. Se asigna un número de secuencia a cada octeto transmitido y es necesaria la recepción de un número de reconocimiento positivo (en inglés, *acknowledgment*, ACK) para cada paquete que el receptor aceptó y procesó. Si no se recibe un ACK durante un tiempo de espera, se retransmiten los datos. El receptor usa los números de secuencia para ordenar correctamente los segmentos que pudiera haber recibido fuera de orden y para eliminar duplicados. La corrupción de los datos es verificada mediante un campo checksum que se agrega a cada segmento transmitido, el receptor verifica este campo y descarta los segmentos dañados. Este protocolo, al igual que TCP asigna un número de secuencia único a cada paquete que se transmite hacia la RVM y requiere que la RVM responda con un un ACK. Si no se detecta la recepción de un ACK, el protocolo TCP/IP retransmite el paquete. Cuando se establece la conexión entre la máquina que

maneja el protocolo y la VM réplica, se crea un mecanismo llamado "ventana de deslizamiento" [Jacobson92]. Esta ventana registra el número de secuencia inicial tanto del emisor como del receptor. La ventana contiene los registros que permiten conocer cuál es el número de secuencia que debe ser enviado y cuál es el número de secuencia que debe ser recibido en ambos sentidos, emisor y receptor. En otras palabras, la ventana almacena el número de bytes que deben ser reconocidos y el número de bytes que han sido enviados y los que aún no se reconocen. Como TCP, el protocolo propuesto incluye los cálculos del checksum basados en una pseudocabecera, este campo se incluye como parte de la cabecera TCP.

4.1.2. Implementación del protocolo

El protocolo puede ejecutar una copia de cada comando remoto basado en TCP que se dirige de la RM hacia una OVM y replicar estas acciones en una o varias RVMs. El objetivo final de este protocolo es mantener estados iguales en todas las VMs, incluyendo memoria, almacenamiento local y conexiones de red. Las interacciones con la OVM que se basan en TCP/IP son replicadas a la RVM. El protocolo se instala en una máquina real que controla la replicación de paquetes. La implementación total de este protocolo se realizó con código abierto: Xen [Xen.org11], *Libnet* [Libnet] y *Libpcap* [Libpcap]. Se usa Xen como hipervisor para todos los sitios, soportando tanto la OVM como las RVMs. Libnet es una interface de programación de aplicaciones (en inglés, *Application Programming Interface*, API) que permite la construcción, modificación e inyección de paquetes de red. Libnet puede trabajar con IP, TCP, UDP, IGMP, etc. Libpcap es una librería que puede capturar paquetes en la capa de red y se usa para definir reglas de filtrado de paquetes, el nodo de captura puede ser normal o promiscuo. Cuando se captura un paquete se ejecuta una retrollamada (en inglés, *callback*).

Los paquetes de red asociados a los comandos ejecutados se capturan y se almacenan en una cola. Cuando alguno de los paquetes de la cola es seleccionado para ser enviado como parte de una nueva conversación con una RVM, se cambian algunos campos, tanto en la cabecera IP como en la cabecera TCP. En la cabecera IP, se modifica el campo de IP destino; la dirección IP de la OVM se reemplaza por la dirección IP de la RVM. La cabecera TCP también sufre algunos cambios. El número de secuencia y el número de

reconocimiento se calcula de acuerdo con el acuerdo RFC793 [Institute81]. El número de secuencia inicial para la conversación replica es igual al número inicial de la conversación original. Los números de ACK son generados cuando se establece la nueva conversación. La conversación réplica se establece de acuerdo con un saludo de tres vías:

1. La RM envía un paquete SYN a la OVM. Este paquete se duplica conservando el mismo número de secuencia y se envía a la RVM con un número de ACK igual a cero.

2. La OVM y la RVM responden con paquetes que tienen las banderas SYN+ACK activadas. En este paso, el receptor genera su propio número de secuencia y llena el campo de número ACK con el número identificado en el paso 1, que es el número de secuencia más 1.

3. La RM envía un paquete ACK. Este se duplica y se envía a la RVM. En ambas conversaciones, el número se secuencia es igual al número de ACK de paquete generado en el paso 2. El número de ACK es igual al número se secuencia generado en el paso 2 más 1.

Los datos que van desde la RM hacia la OVM se envían en el mismo orden durante la conversación entre la RM y la RVM. La RM inicia la conversación enviando un segmento SYN hacia la OVM. La OVM responde con un SYN+ACK; la RM responde con un segmento que contiene un ACK. Al final de este saludo, el cliente y el servidor pueden comenzar a intercambiar información. Cada vez que el cliente o el servidor envían datos, su contraparte debe enviar una señal de que la información ha sido recibida y reconocer cuantos bytes se reciben. Cuando se captura el primer paquete del saludo de tres vías que corresponde a la conversación entre la RM y la OVM, se almacena en una *Cola de Bytes Enviados*. El paquete se modifica y se envía a la RVM. Cuando la OVM responde con un paquete SYN+ACK, éste se almacena en una *Cola de Bytes Recibidos*. El saludo termina cuando la RM envía un ACK con destino OVM. El ACK se captura, duplica y almacena en la *Cola de Bytes Enviados*. Cuando la RM envía un paquete TCP, éste se almacena en la *Cola de Bytes Enviados*. Si el inicio de la *Cola de Bytes Recibidos* se encuentra marcado con 0, significa que la réplica ya ha enviado un paquete de confirmación y se encuentra

lista para recibir un paquete, por lo tanto, se puede extraer el siguiente paquete de la *Cola de Byte Enviados* para ser modificado y enviado a la RVM. De manera contraria, si el número de bytes en el inicio de la *Cola de Bytes Recibidos* es diferente de 0, los paquetes de la Cola de Bytes Enviados esperaran su turno para ser enviados hacia la RVM. Debido a que la cabecera TCP se modifica, el campo checksum es recalculado de acuerdo a las especificaciones en RFC 793. Además, el subcampo timestap se modifica en base al RFC 1323. La frecuencia de reloj timestap se configura en un rango de un milisegundo a 1 segundo por tick. Cuando se detecta la replicación de un paquete SYN, el campo TSVal se inicializa con el valor del paquete original. El campo TSecr se inicializa con un valor de 0. Cuando se envía el último TSVal, TSVal se calcula como el tiempo del último paquete duplicado que se registró más las centésimas de segundo transcurridas. El valor del campo TSecr de un paquete duplicado corresponde al valor de TSVal del último paquete capturado con dirección RVM a OVM. Para evitar que la RM pueda detectar paquetes no solicitados, es necesaria la configuración de reglas de cortafuegos que desechen los paquetes que provienen de la RVM. El filtro de tráfico de red del protocolo de replicación sólo analiza los paquetes TCP/IP que pertenecen a las conversación RM-OVM o RM-RVM. La Figura 4.2 muestra un ejemplo de una conversación TCP/IP y su conversación réplica. Se puede observar que el saludo de 3 vías se lleva a cabo tanto en la conversación original como en la réplica del paquete 1 al paquete 3. En este fragmento de conversación, de acuerdo con el protocolo propuesto, se puede apreciar que el campo de secuencia generado por el cliente se conserva igual para la conversación réplica que en la conversación original. Se observa que el servidor de la conversación réplica genera su propio número de secuencia, esta numeración será la base del campo ACK del cliente réplica. El protocolo propuesto conserva la numeración de secuencia y de ACK de la réplica de acuerdo con el RFC 793.

4.2. Conclusiones

En el presente Capítulo se explicó la operación del protocolo de alta disponibilidad de máquinas virtuales basado en un esquema de replicación de servicios TCP/IP. Se describió el proceso de replicación de paquetes y se listaron los campos TCP/IP que re-

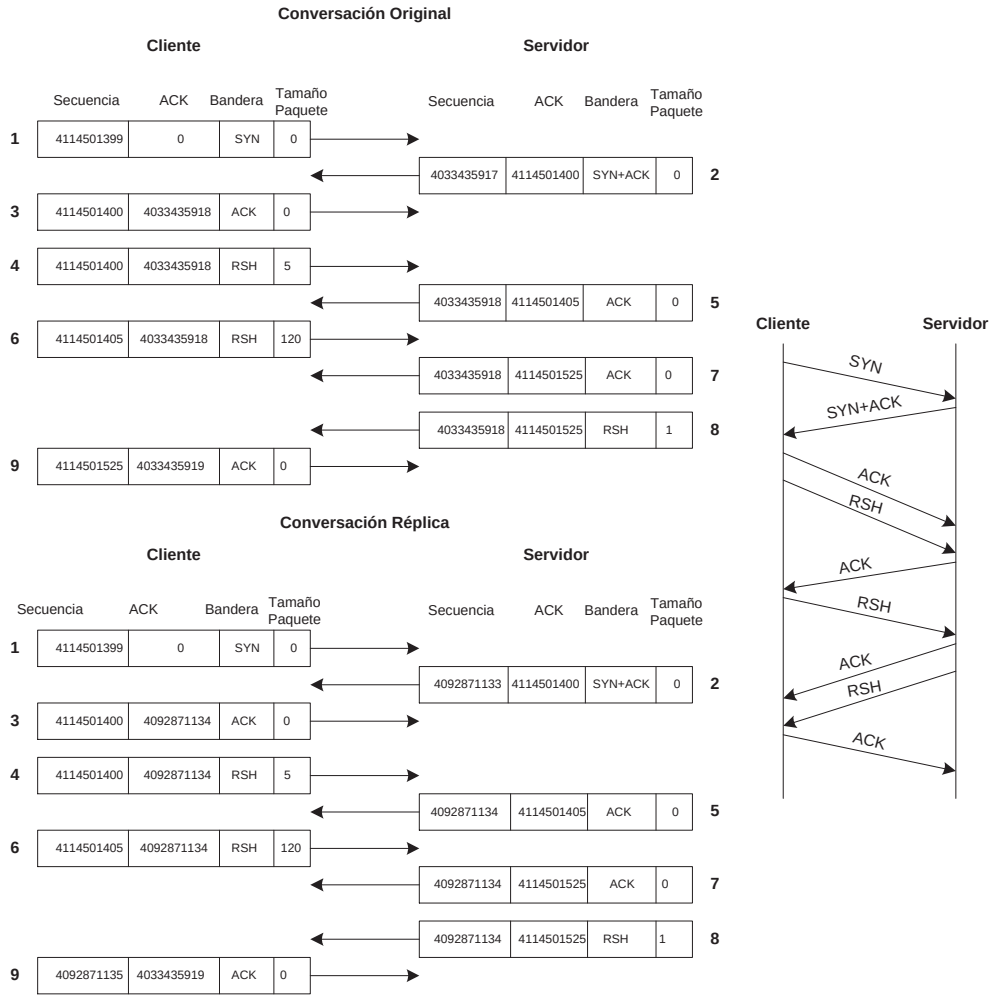


Figura 4.2: Un fragmento de una conversación TCP/IP y su replica

quieren de modificación durante el proceso. Se señalaron las aplicaciones de código abierto que fueron necesarios para la implementación del protocolo; así como la función que cada herramienta tiene durante el proceso de replicación.

Capítulo 5

Casos de Estudio

En las siguientes secciones se describen los tres casos de estudio en los cuales se muestra la aplicación del protocolo de replicación de servicios TCP/IP propuesto.

5.1. Caso de prueba 1: Consistencia de los datos replicados

5.1.1. Escenario de pruebas

Las pruebas de consistencia de datos se realizaron con el siguiente equipo: Un servidor con dos procesadores Xeon quad-core, cada uno de 2.0 GHz, 3GB de RAM y un disco duro de 72GB. El sistema operativo base de este servidor es Debian 4.0, sobre el cual se encuentra instalado el hipervisor Xen 3.1. En este sistema fueron configuradas dos máquinas virtuales de iguales características; sistema operativo huésped es Ubuntu 8.04, 512 MB de memoria RAM, 2 GB como espacio de almacenamiento. La instalación de software para las máquinas virtuales fue idéntica. Además se contó con una laptop con procesador Pentium M a 2.0 GHz, 2 GB de RAM, y un disco duro de 80 GB, donde el sistema operativo base es Debian 4.0.

Dentro de una red de área local, se ejecutan comandos desde una máquina real hacia una máquina virtual. Dicho comando es ejecutado con rsh *(en inglés, Remote Shell)*. El sistema remoto al cual se conecta rsh debe tener en ejecución al demonio rshd. Este demonio usa el puerto TCP número 514.

Tabla 5.1: Comandos rsh

Comandos remotos
rsh 192.168.1.2 /root/configure rsh 192.168.1.2 make -C /root rsh 192.168.1.2 make install -C /root

Desde la máquina real pueden ejecutarse comandos bajo el protocolo TCP/IP como copia de archivos, cambio de permisos de acceso y de ejecución, creación y borrado de archivos, etc.

5.1.2. Resultados

Para verificar la integridad de los datos después de la replicación de varias transacciones TCP, se realizaron algunas pruebas como la instalación de Apache bajo la secuencia de comandos mostrada en la Tabla 5.1:

Previamente se instaló en cada MV la aplicación Open Source Tripwire[SourceForge] que es una herramienta útil en el monitoreo de la integridad y seguridad de datos, ya que alerta sobre los cambios sufridos por archivos específicos dentro del sistema. Tripwire puede funcionar como un sistema de detección de intrusos. En vez de detectar intrusiones al nivel de la interface de red, tripwire detecta los cambios en los objetos del sistema de archivos.

Cuando *tripwire* se inicializa, la aplicación realiza un barrido de todo el sistema de archivos y almacena la información de cada archivo revisado en una base de datos. En el futuro, los mismos archivos son revisados y los resultados que se obtienen son comparados contra los valores almacenados en la base de datos. Los cambios son reportados al usuario. Se emplean hashes criptográficos para detectar cambios en un archivo sin almacenar el contenido completo de un archivo en la base de datos.

Antes de la ejecución del comando remoto sobre la VM, se registró el estado del sistema de archivos de la VM en las bases de datos de *tripwire*, de igual forma se registró el estado de la VM con ayuda de *tripwire* después de ejecutar el comando remoto sobre la VM original y de que el protocolo de réplica duplicara los comandos sobre la VM copia. La toma inicial de registros de *tripwire* se realiza desde un mismo estado ambas máquinas, por

lo cual sus registros son inicializados en cero como se muestra en los reportes siguientes de *tripwire*.

En las Figuras 5.1 y 5.2 se muestra el estado de la máquina virtual original y de la máquina virtual réplica reportado por *tripwire*, respectivamente, antes de la ejecución del protocolo de réplica.

```
Note: Report is not encrypted.
Tripwire(R) 2.3.0 Integrity Check Report

Report generated by:          root
Report created on:            Thu Feb  4 01:45:14 2010
Database last updated on:     Thu Feb  4 01:41:52 2010

===============================================================================
Report Summary:
===============================================================================

Host name:                    ubuntu-1
Host IP address:              192.168.1.2
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/ubuntu-1.twd
Command line used:            tripwire -m c

===============================================================================
Rule Summary:
===============================================================================

-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                       Severity Level    Added    Removed  Modified
  ---------                       --------------    -----    -------  --------
  Invariant Directories           66                0        0        0
  Tripwire Data Files             100               0        0        0
  Other binaries                  66                0        0        0
  Tripwire Binaries               100               0        0        0
  Other libraries                 66                0        0        0
  Root file-system executables    100               0        0        0
  System boot changes             100               0        0        0
  Root file-system libraries      100               0        0        0
  (/lib)
  Critical system boot files      100               0        0        0
  Other configuration files       66                0        0        0
  (/etc)
  Boot Scripts                    100               0        0        0
  Security Control                66                0        0        0
* Root config files               100               0        0        2
* Devices & Kernel information    100               62       62       0

Total objects scanned:  14090
Total violations found:  126
```

Figura 5.1: Estado de la máquina virtual original antes del comando remoto

```
Note: Report is not encrypted.
Tripwire(R) 2.3.0 Integrity Check Report

Report generated by:          root
Report created on:            Thu Feb  4 01:46:39 2010
Database last updated on:     Thu Feb  4 01:43:18 2010

===============================================================================
Report Summary:
===============================================================================

Host name:                    ubuntu-2
Host IP address:              192.168.1.3
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/ubuntu-2.twd
Command line used:            tripwire -m c

===============================================================================
Rule Summary:
===============================================================================

-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                     Severity Level    Added    Removed  Modified
  ---------                     --------------    -----    -------  --------
  Invariant Directories         66                0        0        0
  Tripwire Data Files           100               0        0        0
  Other binaries                66                0        0        0
  Tripwire Binaries             100               0        0        0
  Other libraries               66                0        0        0
  Root file-system executables  100               0        0        0
  System boot changes           100               0        0        0
  Root file-system libraries    100               0        0        0
  (/lib)
  Critical system boot files    100               0        0        0
  Other configuration files     66                0        0        0
  (/etc)
  Boot Scripts                  100               0        0        0
  Security Control              66                0        0        0
* Root config files             100               0        0        2
* Devices & Kernel information  100               62       62       0

Total objects scanned:  14223
Total violations found:  126
```

Figura 5.2: Estado de la máquina virtual réplica antes del comando remoto

Mientras que en las Figuras 5.3 y 5.4 se muestra el estado que reporta *tripwire* después de ejecutar el comando remoto sobre una máquina virtual original y la máquina virtual réplica respectivamente.

```
Note: Report is not encrypted.
Tripwire(R) 2.3.0 Integrity Check Report

Report generated by:          root
Report created on:            Thu Feb  4 02:03:08 2010
Database last updated on:     Thu Feb  4 01:46:00 2010

===============================================================================
Report Summary:
===============================================================================

Host name:                    ubuntu-1
Host IP address:              192.168.1.2
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/ubuntu-1.twd
Command line used:            tripwire -m c

===============================================================================
Rule Summary:
===============================================================================

-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                     Severity Level    Added    Removed  Modified
  ---------                     --------------    -----    -------  --------
  Invariant Directories         66                0        0        0
  Tripwire Data Files           100               0        0        0
  Other binaries                66                0        0        0
  Tripwire Binaries             100               0        0        0
  Other libraries               66                0        0        0
  Root file-system executables  100               0        0        0
* System boot changes           100               0        0        26
  Root file-system libraries    100               0        0        0
  (/lib)
  Critical system boot files    100               0        0        0
* Other configuration files     66                0        0        2
  (/etc)
  Boot Scripts                  100               0        0        0
  Security Control              66                0        0        0
* Root config files             100               779      0        213
* Devices & Kernel information  100               960      960      0

Total objects scanned:  14869
Total violations found:  2940
```

Figura 5.3: Estado de la máquina virtual original después del comando remoto

```
Note: Report is not encrypted.
Tripwire(R) 2.3.0 Integrity Check Report

Report generated by:          root
Report created on:            Thu Feb  4 02:03:17 2010
Database last updated on:     Thu Feb  4 01:47:01 2010

===============================================================================
Report Summary:
===============================================================================

Host name:                    ubuntu-2
Host IP address:              192.168.1.3
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/ubuntu-2.twd
Command line used:            tripwire -m c

===============================================================================
Rule Summary:
===============================================================================

-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                     Severity Level    Added    Removed  Modified
  ---------                     --------------    -----    -------  --------
  Invariant Directories         66                0        0        0
  Tripwire Data Files           100               0        0        0
  Other binaries                66                0        0        0
  Tripwire Binaries             100               0        0        0
  Other libraries               66                0        0        0
  Root file-system executables  100               0        0        0
* System boot changes           100               0        0        26
  Root file-system libraries    100               0        0        0
  (/lib)
  Critical system boot files    100               0        0        0
* Other configuration files     66                0        0        2
  (/etc)
  Boot Scripts                  100               0        0        0
  Security Control              66                0        0        0
* Root config files             100               779      0        213
* Devices & Kernel information  100               957      957      0

Total objects scanned:  15002
Total violations found:  2934
```

Figura 5.4: Estado de la máquina virtual réplica después del comando remoto

De la comparación de ambos registros se puede observar que después de la ejecución de los comandos remotos sobre la VM original hubo cambios en los registros de configuración de Root, lo cual indica la adición de archivos de configuración debido a la instalación de Apache.

En la Tabla 5.2 se puede observar, en el renglón donde se reportan las modificaciones a los archivos de configuración de Root *(* Root config files)*, que la copia VM sufrió varias modificaciones en comparación con el reporte inicial, lo que indica que la copia VM también ejecutó los cambios en el sistema de archivos debido a la ejecución de los comandos remotos replicados.

Tabla 5.2: Modificaciones en archivos de configuración de la VM réplica, según reporte de tripwire

Rule name	Added	Removed	Modified
* Root config files (inicial)	0	0	2
* Root config files (final)	779	0	213

5.1.3. Resumen

El protocolo de replicación implementado duplica comados remotos bajo el protocolo rsh que se dirigen a una máquina virtual hacia otra máquina virtual de respaldo, sin causar la interrupción de los servicios de las máquinas virtuales activas. Además, se ha verificado que al final de un proceso de replicación, el estado de la máquina replica es igual al de la máquina original.

5.2. Caso de prueba 2: Replicación en máquinas remotas

En este caso de estudio, se usaron tres servidores ubicados en diferentes lugares dentro de la ciudad. En la Figura 5.5, se muestran dichas ubicaciones. Se usaron 3 servidores Xeon quad-core 2.0 GHz, con 3GB de RAM y 72GB DD. En el sitio A, se instala el programa de replicación y se configura para replicar en los sitios B y C. Para que la información viaje de manera segura, se configura una VPN entre el sitio A y cada uno de los sitios remotos B y C. El sitio A se conecta a un servicio de Internet ADSL (*Asymmetric Digital Subscriber Line*) de 1MB, el sitio B a un servicio ADSL de 1MB y el sitio C a un servicio ADSL de 2MB.

5.2.1. Escenario de pruebas

En cada uno de los sitios se usa como sistema operativo Debian 4.0 y el hipervisor Xen 3.1.3. En el sitio A (sitio local), se instalan dos VMs con las mismas características. Ambas máquinas tiene como sistema operativo Ubuntu 8.04, con 512MB de memoria RAM y 2 GB de almacenamiento. El software instalado en todas las VMs de los sitios es el mismo. Los servidores remotos se configuran con VMs con sistema operativo 8.04, 512

Figura 5.5: Ubicación de los servidores de réplica en la ciudad

MB de RAM y 2 GB de almacenamiento. El sistema de manejador de bases de datos MySQL versión 5.0.51 se instala en todas las MVs. Se crea una base de datos en una VM del sitio y se copia a los servidores remotos B y C. Un script del sistema operativo ejecuta operaciones aleatorias sobre los registros de la base de datos, dichas transacciones pueden ser de inserción, borrado o modificación. Para que fuera posible realizar la inserción y la modificación de campos aleatorios; los valores de los campos fueron limitados a un conjunto de valores determinado y los campos fueron definidos de tipo enumerado (ENUM) para MySQL [MySQL]. El programa de replicación se instala en la RM ubicada en el sitio A (dirección IP, `192.168.2.2`) y configurada para ejecutar una operación aleatoria cada dos minutos sobre la base de datos localizada en la OVM en sitio A, con dirección IP `192.168.2.3`. Las operaciones son replicadas a las VMs remotas en los sitios B (RVM1) y C (RVM2). El sniffer Wireshark registra el tráfico de red. La conversación original se establece entre la RM local y la RVM1 remota con IP `192.168.3.2` y con la RVM2 con dirección `192.168.4.7`. El script se sistema operativo se ejecuta durante 72 horas continuas. La base de datos se modifica cada 2 minutos. Al final del periodo de prueba se verifica la consistencia de la información a través de la verificación del checksum de la base de datos

y los registros binarios de MySQL.

5.2.2. Resultados

En esta sección, se presentan los tiempos promedio de RTT (*Round Trip-Time*) para transacciones sin réplica, con una y dos réplicas activas. RTT [Forouzan10] es el tiempo que toma en enviar un segmento y recibir un ACK de este segmento. Se presentan las medidas promedio desde la RM hacia la OVM, después los RTTs de la RM hacia las RVM1 y la RVM2 en los sitios B y C respectivamente. Los resultados se presentan en la Tabla 5.3. En la primer columna, de izquierda a derecha se indica el sentido en el cual fue medido el RTT, puede ser medido en el sentido cliente a servidor (RM-Otra MV) o bien en el sentido servidor a cliente (Otra MV-RM). La segunda columna identifica el número de réplicas activas durante la medición del RTT. La tercera columna muestra el RTT promedio de las mediciones expresado en ms. La cuarta columna indica la desviación estándar promedio del experimiento correspondiente y la última columna muestra el porcentaje de degradación que sufre el RTT variando el número de réplicas. Se puede observar que la conversación

Tabla 5.3: RTT

	Réplicas	**RTT Promedio (ms)**	**Desviación estándar**	**%Degradación RTT**
RM-OVM	0	0.03320	0.00000287	—
	1	0.03720	0.00000439	12.05 *vs* 0 réplicas
	2	0.03870	0.00000522	16.57 *vs* 0 réplicas
OVM-RM	0	0.03690	0.00000425	—
	1	0.03660	0.00000682	-0.81 *vs* 0 réplicas
	2	0.03763	0.00000646	1.98 *vs* 0 réplicas
RM-RVM1	1	24.4000	0.00149000	—
	2	24.1000	0.00850000	1.24 *vs* 1 réplica
RVM1-RM	1	0.06270	0.00005740	—
	2	0.06810	0.00002390	8.61 *vs* 1 réplica
RM-RVM2	2	22.3120	0.00148200	92.58 *vs* RM-RVM1 2 réplicas
RVM2-RM	2	0.12740	0.00032200	87.08 *vs* RVM1-RM 2 réplicas

RM-OVM con una réplica es 12 % más lenta que el proceso sin réplicas. Con dos réplicas, la misma conversación es 16 % más lenta. El proceso con una réplica es 4 % más rápido que el proceso con dos replicas. Las conversaciones entre la OVM y la RM experimentan una sobrecarga de menos del 2 % aun cuando existen dos replicas activas. Los RTTs del cliente al servidor (RM-RVM1) con dos replicas tienen una sobrecarga de menos del 1.25 % en comparación con una réplica, y del servidor al cliente (RVM1-RM) de menos del 9 %. Este

resultado muestra que el incremento de número de réplicas no causa retrasos que pudieran interrumpir la operación del sistema. Las conversaciones entre RM y RVM2 son ligeramente más rápidas que las conversaciones entre RM-RVM1 cuando hay dos replicas activas. Esto es debido a que el servicio de Internet en el sitio C es dos veces más rápido que el servicio en el sitio B. Se debe mencionar que los resultados fueron obtenidos en un escenario real donde el proceso de replicación se encuentra ejecutándose en fondo sin intervención de usuarios. En la Table 5.3 también se puede observar uno de los efectos de la conexión doméstica ADSL, la cual está diseñada para que la capacidad de bajada de datos (descarga) sea mayor que la de subida. Esta característica corresponde a la demanda del uso de Internet por la mayoría de los usuarios finales, que reciben más información de la que envían [Bathrick99]. Se considera que el flujo de subida se refiere a enviar datos desde un cliente o computadora local hacia un servidor o host remoto. El flujo de datos de bajada se refiere a los datos transferidos desde un servidor remoto a una máquina local. Las típicas velocidades de acceso a internet son de una capacidad de bajada de 6 Mbps y de subida una capacidad de 756 Kbps, lo cual es altamente asimétrico [Podlesny12]. Varios investigadores han explorado la dinámica del tráfico TCP en dos sentidos realizando el análisis en lineas ADSL. En [Kalampoukas98] se asume que la degradación en las líneas asimétricas es debido a la compresión ACK. En [Heusse11] se fundamenta que la razón de la degradación TCP en líneas asimétricas es debido al efecto péndulo en el cual, el cuello de botella de una línea ADSL se encuentra en una sola dirección a la vez. Los resultados observados incluyen la sobrecarga en el sistema por realizar la replicación, los accesos a internet de los usuarios de los sitios designados, las actividades de los usuarios durante las pruebas, el tráfico de internet, etc.

5.2.3. Resumen

Nuestro protocolo está basado en TCP/IP y puede ser implementado en distribuciones de Linux. No está limitado a trabajar dentro de una Red de Área Local y puede ser configurado para crear y mantener réplicas locales y remotas. Además, el proceso de replicación es ejecutado en una forma segura. Las comunicaciones pueden ser enviadas a través de un túnel IPSEC entre la máquina que controla la replicación y la máquina que contiene la réplica. Este es compatible con los servicios de acceso a internet y no requiere de

un servicio dedicado de internet para realizar el proceso de replicación. Las pruebas fueron llevadas a cabo sin interrumpir las actividades de los usuarios. Para la implementación de este protocolo no es necesario hacer modificaciones a hardware o software. No se realizaron cambios al código fuente de ningún sistema operativo, ni al sistema operativo anfitrión ni al sistema operativo huésped de las VMs. El programa de replicación puede ser ejecutado dentro de un ambiente no virtualizado. Se debe mencionar que en la implementación de este caso de estudio, un nodo centralizado controla las réplicas y los paquetes replicados son almacenados en memoria. Esto es útil cuando ocurre alguna falla, pero con un número creciente de réplicas, podría ocurrir una saturación de memoria.

5.3. Caso de prueba 3: Tolerancia a fallas

Una característica que distingue a los sistemas distribuidos de los sistemas de una sola máquina es la noción de la falla parcial. En un sistema distribuido, una falla parcial puede acontecer cunado falla un componente. Una falla puede afectar la operación de algunos componentes, al tiempo que otros más no se ven afectados en absoluto. En contraste, en un sistema no distribuido, una falla a menudo es total en sentido de que afecta a todos los componentes y puede dejar inactivo a un sistema. Un objetivo importante en el diseño de sistemas distribuidos es la construcción de mecanismos que permitan la recuperación automática a fallas parciales sin que se afecte seriamente el desempeño total. Siempre que una falla ocurra, el sistema distribuido deberá continuar operando de modo aceptable mientras se realizan las reparaciones, dicho de otra manera, deberá tolerar las fallas y continuar operando hasta cierto grado incluso en presencia de la falla. Un concepto importante que se toma en cuenta para la tolerancia a fallas es la atomicidad. Por ejemplo, en transacciones distribuidas es necesario garantizar que se realicen todas las operaciones involucradas en un proceso o que no se realice ninguna. Un sistema tolerante a fallas es un sistema fiable [Forouzan10]. La fiabilidad comprende varios conceptos como son la disponibilidad, confiabilidad, seguridad y mantenimiento. La disponibilidad es la propiedad de que un sistema está listo para ser utilizado de inmediato. Se refiere a la probabilidad de que el sistema esté operando correctamente en un momento dado y disponible para realizar las

operaciones que requieren los usuarios. Un sistema altamente disponible tiene muy altas probabilidades de que esté en funcionamiento en un momento dado. La confiabilidad se refiere a la propiedad de que un sistema sea capaz de funcionar de manera continua sin fallar. La seguridad se refiere a la situación en que no acontece nada catastrófico cuando un sistema deja de funcionar correctamente durante un tiempo. Y el mantenimiento se refiere a cuán fácil puede ser reparado un sistema que falló. La recuperación automática de fallas es fácil de expresar, pero difícil de realizar.

5.3.1. Protocolo para tolerancia a fallas

Para este caso de estudio se hizo una modificación a la configuración inicial del protocolo. En este caso, el protocolo de replicación ya no se encuentra de la misma máquina que realiza las peticiones remotas, llamada RM. En esta configuración el cliente envía peticiones a una única dirección IP. Esta dirección es una dirección virtual de la máquina que donde se encuentra instalado el protocolo replicador. El cliente envía peticiones a esta IP virtual y el protocolo redirige estás peticiones hacia la OVM y a la vez, replica estas peticiones hacia las RVMs que existan. De manera general, el protocolo realiza los siguientes pasos:

1. El cliente envía una petición hacia la Conexión del Servidor Principal (en inglés, *Principal Server Connection*, PSC).

2. Los paquetes con dirección IP origen del cliente y con dirección IP destino PSC son atrapados por el protocolo.

3. El protocolo realiza unos cambios a los paquetes atrapados. La dirección IP origen se cambia por la dirección origen del PSC. La dirección IP destino se cambia por la dirección de la OVM.

4. Además los paquetes son replicados un número de veces igual al número de máquinas réplica existentes. Estos paquetes también cambiaran la IP fuente por la dirección IP del PSC y la dirección IP destino por la dirección de la correspondiente RVM.

5. La OVM recibe los paquetes con la dirección IP origen de la PSC, los procesa y responde.

6. Cuando el protocolo detecta un paquete de respuesta con dirección IP de origen de la OVM y destino la PSC, el paquete es atrapado y modificado para ser redirigido hacia la máquina cliente. Para lo cual, el paquete se modifica, la dirección IP fuente será la dirección IP del PSC y la dirección destino será la dirección IP del cliente (RM).

7. El cliente recibe las respuestas como si estuviera interactuando directamente con el PSC.

8. Las respuestas enviadas por las réplicas son procesadas en la máquina intermedia que tiene instalado el protocolo.

La configuración del protocolo es esquematizada en la Figura 5.6. En esta configuración del

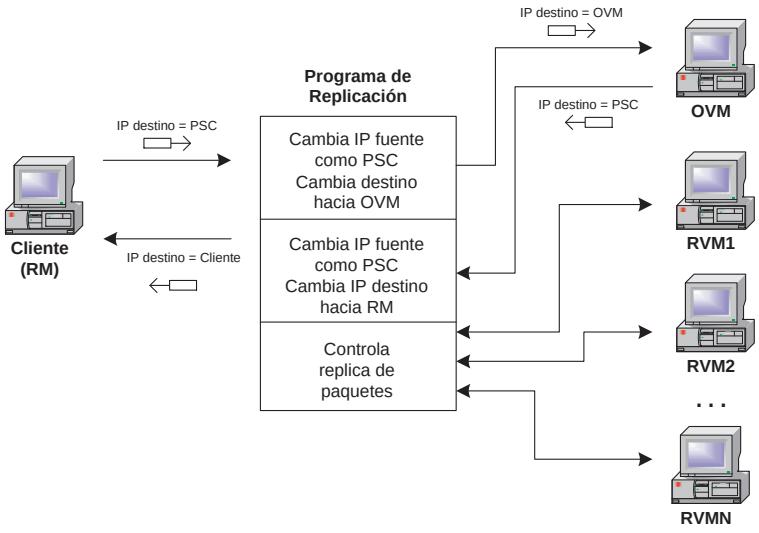


Figura 5.6: Configuración del replicación considerando tolerancia a fallas

sistema existe un cliente que envía comandos TCP/IP remotos hacia la dirección IP del PSC estos comandos son redirigidos y atendidos por la OVM. El protocolo almacena una lista de los servidores disponibles y sus direcciones IP, incluyendo la dirección IP única del servidor PSC. La dirección PSC es diferente a la dirección IP de la OVM. También existe una lista de RVM hacia donde dichos comandos son replicados y que permiten que todas

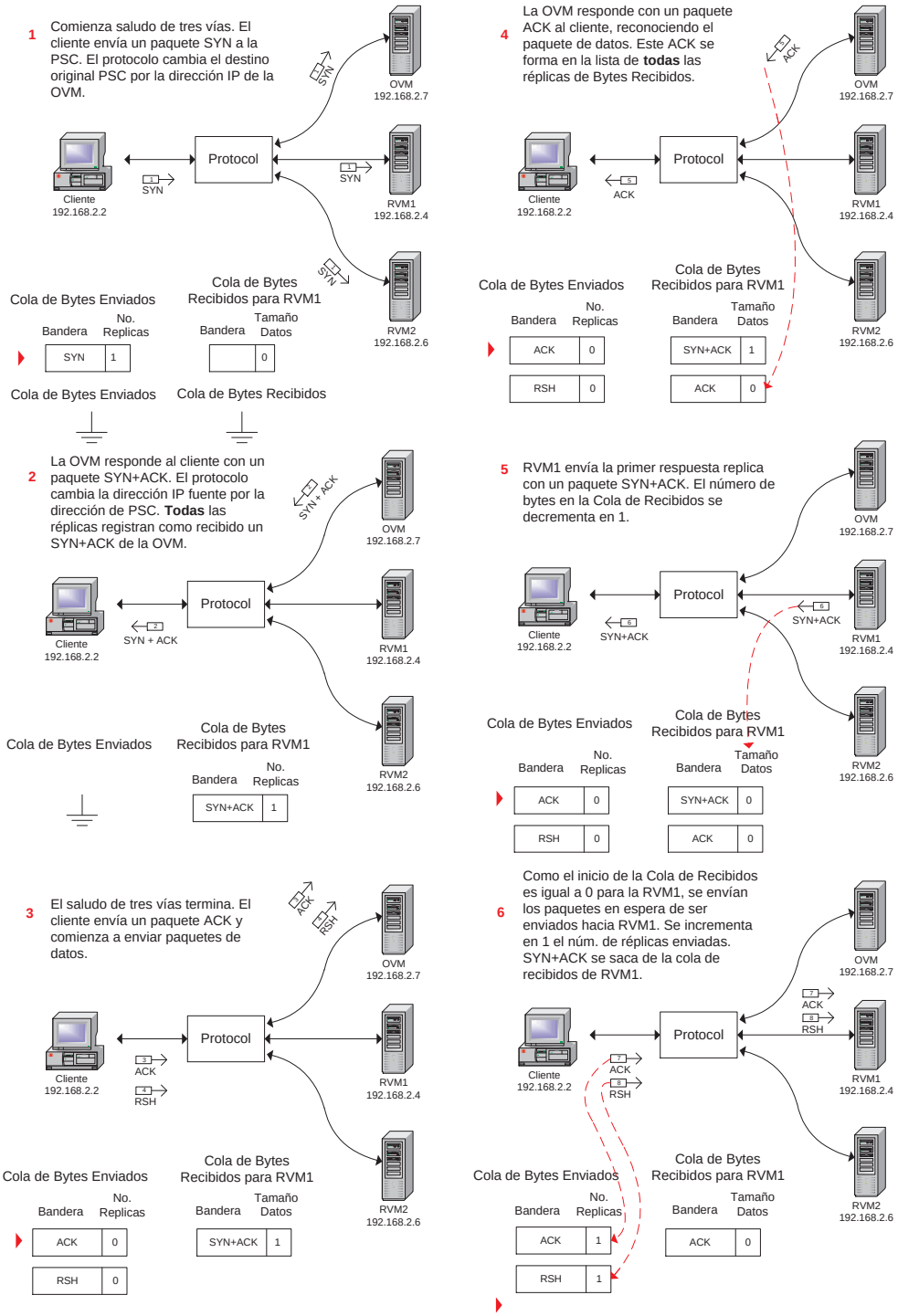
la RVMs conserven el mismo estado de la OVM. Para que la redirección de las peticiones hechas por el cliente sea posible, en la máquina cliente se habilitó una IP virtual por medio de comandos ARP para la máquina donde se instala en protocolo de replicación, esta IP virtual es la dirección IP del PSC. En cada una de las RVMs también se asocia la dirección MAC de la máquina que controla el protocolo con la IP del PSC. Esta configuración permite que el cliente se comunique siempre a una dirección fija sin conocer de manera directa cual es la dirección de la OVM. Así la OVM podría ser sustituida de una manera más transparente por una RMV en caso de existir una falla en el sistema. En la Figura 5.7 ilustra un fragmento de una conversación TCP/IP y su correspondiente réplica de paquetes hacia diferentes máquinas. El protocolo considera que pueden existir dos tipos de la fallas: Falla de la OVM y falla en alguna RMV. Se considera que la OVM y las réplicas no fallan de manera simultánea.

5.3.2. Falla de la OVM

Se considera que cuando la OVM falla, la primera RVM que se encuentra en lista de réplicas disponibles toma el lugar de la OVM. Para el sistema existe una lista de máquinas disponibles donde se guarda la dirección IP asociada con cada máquina física y la función que realiza, por ejemplo:

- 0, identifica la dirección IP del PSC
- 1, identifica una dirección IP cliente
- 2, pertenece a la OVM
- 3, identifica una RVM
- 4, una máquina que ha tomado el lugar de la OVM y se encuentra en recuperación

Cuando se detecta que la OVM ha dejado de responder, la primer RVM toma su lugar y para lo cual se siguen los siguientes pasos: Se elimina la OVM de la lista de máquinas disponibles dentro del sistema. La primer RVM es identificada en el estado 4 que indica que toma el lugar de la OVM. Antes de que la RVM sea identificada plenamente como OVM,



la conversación entre el PSC y la RVM tiene que alcanzar el mismo estado que la tenía la conversación entre el cliente y el PSC antes de que ocurriera la falla. Para ello se verifica

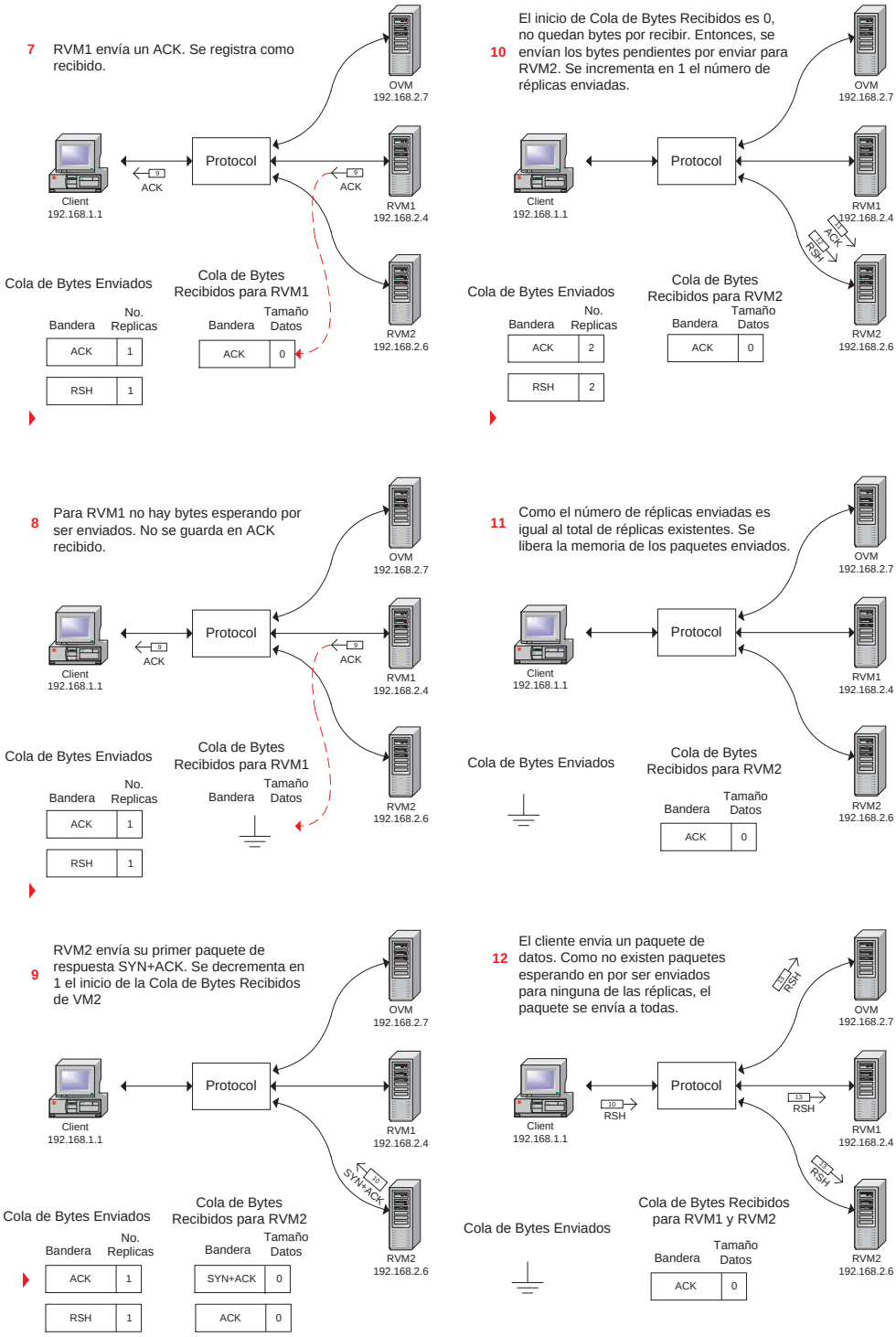


Figura 5.7: Fragmento de una conversación replicada

que el número de bytes enviados y recibidos en la conversación réplica llegue al mismo número de bytes enviados y recibidos por la conversación original antes de la falla. En el momento que ocurre la falla puede darse la situación de que existan paquetes en tránsito; ya sean paquetes enviados por la máquina cliente (RM) o paquetes pendientes de recibir antes de enviar los paquetes formados en la lista de pendientes por enviar. Cuando la RVM queda en el mismo estado de paquetes enviados y recibidos que la OVM antes de la falla, la RVM entra en un estado de recuperación hasta que el comando en ejecución sea finalizado. Durante la recuperación, si el protocolo identifica un paquete que tiene como dirección IP fuente la que pertenece a la máquina réplica que tomará el lugar de la OVM, se captura el paquete y se modifican algunos campos:

1. La dirección IP fuente cambia de la dirección en recuperación por la del PSC.

2. La dirección IP destino cambia de la dirección del PSC a la del cliente.

3. Los campos de la cabecera TCP como el número de secuencia y el número ACK deben ser reemplazados por la numeración que correspondería al siguiente paquete recibido por el cliente durante la conversación original.

4. También los campos que corresponden a los subcampos *timestamp* de las opciones TCP deben ser cambiados para conservar la numeración de la conversación ya iniciada entre el cliente y el PSC.

Además, cuando se identifica un paquete de estas características, se deben alterar los registros de la ventana de deslizamiento de la conversación original para preservar las numeraciones de secuencia y ACK en correcto estado.

Durante la recuperación, la conversación ya establecida entre el PSC y la RVM que tomará el lugar de la OVM se debe mantener activa. Por lo tanto, esta conversación se sigue manejando como si fuera una réplica normal activa. Cuando termina el proceso de recuperación la máquina que se encuentra en estado de recuperación cambia su estado a OVM.

5.3.3. Falla de RVM

Cuando se detecta la falla de una RVM, el proceso de recuperación sigue los siguientes pasos:

1. Se busca la réplica en falla en la lista de réplicas reconocidas por el protocolo.
2. Se elimina de memoria la información que corresponde la réplica en falla, tal como la Cola de Bytes Recibidos por la réplica hasta el momento y todas las variables internas que mantienen su estado.
3. Se libera la memoria ocupada por las variables que mantienen el estado de la réplica.
4. Se decrementa en uno el número de réplicas activas.

5.3.4. Descripción del escenario de pruebas

Para probar la configuración de tolerancia a fallas se tiene como cliente una computadora Latitud D410 con procesador Pentium M de 2.0 GHz, 2GB RAM, 80 GB DD. La computadora intermedia Xeon *quad-core* 2.0 GHz, con 3GB RAM, 72 GB DD, donde se instala el protocolo de replicación. La OVM se instala dentro del servidor Xeon y en la computadora que funge como cliente se instalan hasta 3 RVMs. El sistema operativo base es Debian 4.0 y el hipervisor Xen 3.1.3. Tanto la OVM como las réplicas tienen sistema operativo Ubuntu 8.04, con 512 de RAM y 2GB de almacenamiento. Para realizar las pruebas de rendimiento y de tolerancia a fallas se usa el *script* mencionado en la sección 5.2 del caso de estudio de replicación remota.

La máquina cliente tiene la dirección IP `192.168.2.2`. Existe una dirección única de PSC que es la `192.168.2.1` y por medio de un comando arp -s en el cliente se asocia esta dirección virtual con la dirección MAC (00:21:5A:44:DA:BA) de la máquina que tiene el protocolo activo. Este mismo comando es ejecutado en cada una de las RVMs que se encuentran registradas como replicas activas. Todas las VMs se encuentran en el mismo segmento de red. La Figura 5.8 muestra esta configuración. El *script* fue ejecutado para realizar 500 transacciones aleatorias cada 8 segundos sobre una base de datos instalada en la OVM y sus réplicas.

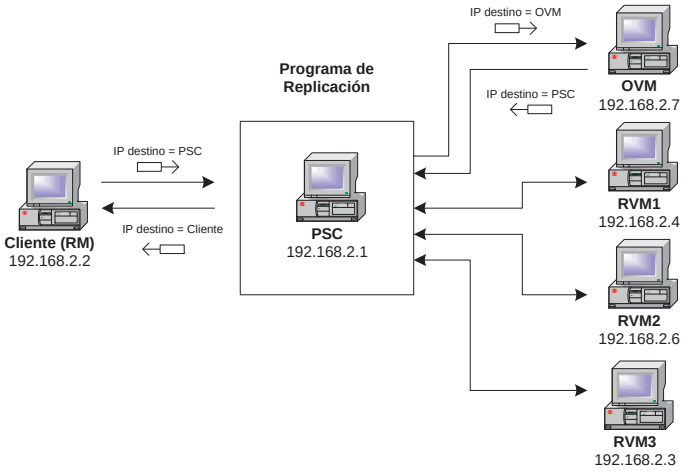


Figura 5.8: Configuración para caso de estudio

5.3.5. Resultados de Eficiencia

A continuación se presentan los RTTs promedio para las transacciones realizadas entre el cliente y la OVM cuando no hay replicas, con una, dos y tres replicas activas. La Tabla 5.4 muestra los resultados obtenidos.

Tabla 5.4: RTT promedio configuración con PSC, sentido cliente-servidor

	Réplicas	**RTT Promedio(ms)**	**%Degradación RTT**
RM-OVM	0	0.026990	—
	1	0.048890	81.16 *vs* 0 réplicas
	2	0.048750	80.64 *vs* 0 réplicas
	3	0.050710	87.92 *vs* 0 réplicas
PSC-RVM1	1	6.238400	—
	2	6.418100	2.88 *vs* 1 réplica
	3	7.519844	20.54 *vs* 1 réplica
PSC-RVM2	2	9.778470	65.63 *vs* PSC-RVM1 2 réplicas
	3	13.68177	39.92 *vs* 2 réplicas
PSC-RVM3	3	13.56890	0.82 +eficiente *vs* PSC-RMV2 3 rep.

Los resultados de esta tabla estan medidos en el sentido cliente-servidor. En la columna uno se observa que el RTT fue medido desde la RM (cliente) hacia alguna máquina virtual, que puede ser la OVM o alguna de las tres réplicas. La columna dos nos indica el número de réplicas activas al tomar la medición. La columna 3 indica el RTT promedio en milisegundos y la columna 4 el porcentaje de degradación RTT. En la Tabla 5.4 pode-

mos observar que el mecanismo de replicación introduce una sobrecarga en el sistema y el RTT promedio sufren degradación en comparación con los RTT promedio de un sistema TCP sin replicación de comandos remotos. Se puede observar que la conversación entre el cliente y la OVM sufre una degradación del 81.16 % con una réplica en comparación de una conversación sin replicas. El RTT es degradado un 80.64 % con dos réplicas activas y un 82.92 % con tres replicas activas. Observando los resultados para la réplica PSC-RVM1, se tiene que en comparación con el RTT de una réplica, se sufre una degradación del 2.88 % cuando hay dos replicas activas y un 20.54 % si hay tres replicas activas. En la misma tabla, se puede observar que la conversación PSC-VM2, que se crea cuando hay dos o más replicas activas, es un 39.91 % más lenta cuando hay tres replicas activas que el RTT medido con dos replicas. También se puede notar que el RTT promedio de las conversaciones PSC-RVM2 y PSC-RVM3 se encuentra alrededor de los 13.5 ms para ambos casos. Se puede apreciar que el RTT promedio sufre una degradación cuando al sistema se incorpora la operación del protocolo de replicación. Se introduce un retraso con el procesamiento intermedio del paquete original. El paquete original se procesa el la máquina intermedia PSC, se cambia de dirección IP fuente y destino y se envia a la OVM. El paquete procesado por la OVM, es captado nuevamente por la PSC y procesado. A este paquete se le cambia la IP origen y destino, finalmente regresando a la RM. Todo este proceso introduce que el RTT promedio se eleve. Cuando hay una réplica activa, el RTT medido también incluye un tiempo adicional de procesamiento del paquete que va hacia la RVM1, ya que este paquete debe ser sacado de la cola donde espera su turno de ser enviado, modificado en los campos necesarios, enviado hacia la RVM1 y esperar su paquete ACK de reconocimiento. Cuando se incrementa una réplicas también se incrementa el tiempo requerido para hacer las modificaciones necesarias al paquete que será replicado, enviarlo a la réplica correspodiente y esperar su paquete de reconocimiento. Debe mencionarse que para las pruebas realizadas la OVM se encontraba ubicada en un servidor con mejores características de desempeño que las RMVs. Sin embargo, a pesar de que el RTT desde el punto de vista del cliente (RM) sufre hasta un 87.92 % de degradación con tres réplicas activas, el RTT promedio se encuentra en 0.050710 ms lo cual es imperceptible para el usuario. En la literatura relacionada no se encuentran experimentos que sean comparables con los realizados en la presente tesis, pero en [Aghdaie09] se men-

ciona que su esquema de replicación duplex, que realiza la replicación de un servidor web primario con su respectivo servidor secundario. El cliente lanza constantes requerimientos de aproximadamente 50 bytes. Las respuestas generadas se encuentran también en el rango de 50 bytes. Se reporta un tiempo de latencia promedio de 6.40 ms. Tiempo que es similar al RTT promedio en los experimentos del protocolo de replicación cuando se duplican las transacciones de la OVM en la RVM1 el cual es de 6.2384 ms y aún cuando existen dos réplicas activas, el tiempo obtenido con el protocolo de replicación es de 6.4181 ms. En [Shao08] se menciona que la latencia incrementa de un 15 % a un 90 % cuando se introduce la replicación de un servidor secundario. El protocolo de replicación propuesto se encuntra dentro de estos parámetros. En la Tabla 5.5 se muestran los RTTs promedio en un sentido servidor-cliente que corresponden al sistema sin replicas, con una, dos y tres replicas activas.

Tabla 5.5: RTT promedio configuración con PSC, sentido servidor-cliente

	Réplicas	**RTT Promedio(ms)**	**%Degradación RTT**
OVM-RM	0	6.908490	—
	1	26.86444	388.86 *vs* 0 réplicas
	2	32.29798	467.51 *vs* 0 réplicas
	3	43.43767	628.76 *vs* 0 réplicas
RVM1-RM	1	43.37405	—
	2	49.71785	114.63 *vs* 1 réplica
	3	68.00150	156.78 *vs* 1 réplica
RVM2-RM	2	43.50354	0.12 + eficiente *vs* RVM1-RM réplicas
	3	57.12202	131.30 *vs* 2 réplicas
RVM3-RM	3	58.53689	2.48 *vs* RMV2-RM 3 rep.

La Tabla 5.5 muestra los RTTs promedio en el sentido servidor-cliente, también muestra que el protocolo introduce un sobrecarga y produce un incremento en el RTT promedio. Para el sentido OVM-Cliente, el RTT promedio de dos replicas es 3.89 veces mayor el RTT promedio sin replicas; el RTT promedio con dos replicas es 4.68 veces mayor y el de tres réplicas es 6.29 veces mayor. Para la conversación en sentido RVM1-PSC, el RTT promedio medido con dos replicas crece 1.15 veces en comparación con el RTT medido con una réplica y 1.57 veces con dos replicas activas. El RTT promedio en sentido RVM2-PSC aumenta un 31 % cuando hay tres replicas a comparación del tiempo medido cuando hay dos replicas en operación. Nuevamente, los tiempos RVM2-PSC y RVM3-PSC tiene RTTs alrededor de un mismo número, que es 58 ms. Se observa que el RTT en sentido servidor-

cliente sufre un porcentaje de degradación mayor que los resultados presentados en el sentido cliente-servidor. Esto es debido a que el reconocimiento de los paquetes que es esperado para completar la medición del RTT llega después de que los paquetes varios procesos de replicación se han completado de forma intermedia. En esta configuración tolerante a fallas las réplicas se encuentran en una máquina con menor capacidad de procesamiento, lo cual también influye en la medición de los tiempos promedio de RTT.

La replicación hacia otras máquinas produce un incremento en el RTT promedio comparando el RTT del sistema sin replicación, el esquema manejado permite la implementación de un mecanismo de recuperación de fallas automático y el manejo de cambio de ubicación física de la OVM de manera transparente para el usuario.

5.3.6. Falla en la OVM

En este experimento, se simularon fallas de congelación de la máquina OVM, donde este servidor se encontraba trabajando correctamente hasta que se detuvo. Se midió el intervalo de tiempo que inicia cuando se detecta que la OVM deja de responder a las peticiones del PSC y termina cuando la primera máquina de la lista entra en estado de recuperación y comienza a redirigir los paquetes que envía como respuesta a las peticiones del PSC. En este momento, la máquina intermedia con el protocolo en funcionamiento reconoce que RVM1 será la máquina que atienda las peticiones del cliente. Se simuló la falla de la OVM cuando está en funcionamiento una, dos o tres replicas. Se tomaron 30 muestras en cada caso y los tiempos promedio en iniciar la recuperación estando en actividad una, dos y tres muestras. Los resultados obtenidos se muestran en la Tabla 5.6:

No. Réplicas	**Tiempo redirigir OVM-nueva OVM (ms)**
1	5.05
2	5.07
3	5.64

Tabla 5.6: Tiempos para redirigir tráfico de OVM-a-nueva OVM

Se puede observar que el tiempo que transcurre entre que el protocolo reconoce la falla y la RVM1 alcanza el estado que tenía la OVM hasta antes de dejar de respon-

der permanece alrededor de los 5 ms, lo cual es un tiempo imperceptible dentro de las actividades de replicación. Estos tiempos de inicio de recuperación permiten que la comunicación permanezca interrumpida entre el cliente y la nueva OVM. Tampoco las réplicas que permanecen en funcionamiento se ven afectadas por el cambio de OVM.

5.3.7. Falla de RVM

Se realizaron simulaciones de fallas en la RVM1. Si el protocolo detecta que una réplica ha dejado de responder a las peticiones hechas por el PSC. La réplica es eliminada de la lista de réplicas disponibles y se libera la memoria donde se registra el estado de su conexión. Los tiempos que toma liberación de esta memoria son despreciables. Para el peor de los tiempos registrados cuando se encuentran en funcionamiento tres réplicas y se elimina una, la liberación de memoria se realizó en un tiempo de 0.003526 ms.

5.3.8. Resumen

El protocolo de replicación propuesto permite la recuperación automática a fallas parciales dentro del sistema. Permite la recuperación de la falla de una OVM y de una RVM garantizando la disponibilidad del sistema y la integridad de los datos. También se pudo observar que la disponibilidad del sistema conlleva a una penalización en cuanto a desempeño del sistema. De acuerdo a los datos reportados, la conversación entre el cliente y el PSC sufre una degradación del 81.16 % cuando el sistema funciona con una réplica activa. Este incremento en el RTT promedio es debido a que el RTT entre el cliente y el PSC considera el tiempo de envío de un paquete cuando sale del cliente, llega a la máquina intermedia y el mismo paquete es reenviado hacia la OVM, entonces la OVM regresa la respuesta al PSC y esta respuesta es reenviada al cliente. Por lo tanto, el RTT se incrementa por lo menos el doble cuando se aplica el esquema de tolerancia fallas. La alta disponibilidad en los sistemas puede acarrear una degradación en el rendimiento del sistema. Sin embargo, en el protocolo propuesto la degradación presentada es aún imperceptible desde el punto de vista del cliente la cual llaga ser de 0.050710 ms, en el peor de los casos con tres réplicas activas. En referencias a esquemas de replicación sobre TCP/IP se mencionan casos de replicación con máximo una replica activa donde el tiempo promedio de latencia es de

6.40 ms. [Aghdaie09], lo cual es un tiempo aún mayor de lo obtenido con el protocolo de replicación propuesto, donde se los resultados reportaron que con una réplica activa se tiene un tiempo RTT promedio de 6.2384 ms y aún con dos réplicas activas se tiene un tiempo de 6.4181 ms. En [Shao08] se reportó que la degradación de los esquemas de replicación con una réplica activa puede estar en 15 %-80 % y el protocolo propuesto se encuentra entre los limites de penalización sufridos por los esquemas actuales de replicación bajo TCP/IP. El protocolo prouesto se basa en una replicación activa donde los paquetes los paquetes dirigidos a la OVM son inmediatamente replicados hacias las RVMs existentes, dicho esquema coloca todas las estructuras en memoria y podría causar degradación del sistema en medida que el número de réplicas crece.

5.4. Conclusiones

En el caso de estudio número uno se verificó la integridad de los datos replicados mediante el uso de la aplicación de código abierto *tripwire*. Se instaló en la máquina original y en la replica la aplicación Apache. Se verificó que estado del sistema fuera igual en ambas máquinas antes de iniciar la instalación y después de la instalación se verificó que el número de el número de objetos agregados y modificados en la configuración de los archivos root era igual. Lo cual verifica que en ambas máquinas ocurrieron los mismos cambios durante la instalación de la aplicación. En el segundo caso de estudio se realizaron replicaciones en un ambiente WAN con hasta dos réplicas activas. Se observó que hubo una degradación de hasta 16 % del RTT promedio cuando hay dos réplicas activas. Sin embargo, este incremento en RTT no interrumpe las actividades de los usuarios. Estas pruebas se registraron en un ambiente no exclusivo, es decir los usuarios de cada sitio llevaron a cabo sus actividades cotidianas. En este caso de estudio se vieron reflejadas las características de las conexiones ADSL, ya que el RRT promedio medido en sentido cliente-servidor en el medio WAN es mayor que el RTT promedio medido en sentido servidor-cliente. De manera similar a las condiciones que brinda el servicio ADSL, donde la capacidad de bajada de datos es mayor que la de subida. Las pruebas de replicación en el caso de prueba tres se llevaron a cabo en un ambiente LAN donde se habilitaron hasta tres réplicas de manera simultánea. En este

caso de prueba se configuró un esquema tolerante a fallas; se considera que tanto la máquina OVM como una RVM puede fallar. En este esquema de replicación el cliente se comunica con una dirección única PSC y no tiene conocimiento de cual es la dirección de la máquina original ni de las máquinas replicas. El esquema de replicación sufre una sobrecarga debido a el procesamiento adicional que recibe cada paquete que lleva como destino la dirección única PSC, ya que ésta es una dirección intermedia y el paquete debe de viajar hasta su destino final que es la OVM y realizar tantas copias del paquete como número de réplicas exista. El RRT promedio llega a incrementar hasta un 87.92 % cuando hay tres réplicas activas, el RTT promedio asociado a estas condiciones es de 0.050710, lo cual es imperceptible para el usuario. La sustición de una máquina OVM en falla por una réplica se lleva en hasta 5.64 ms., lo cual permite al usuario desarrollar sus actividades sin interrupciones.

Capítulo 6

Conclusiones Generales y Trabajo Futuro

6.1. Conclusiones Generales

Como uno de los resultados obtenidos en esta investigación doctoral, se propone el diseño e implementación de un sistema de replicación TCP/IP que proporciona alta disponibilidad de los servicios. Un mecanismo de replicación de VMs mantiene el mismo estado de una VM original en varias copias de la misma máquina, de manera simultánea. El protocolo replica comandos de servicios TCP/IP sin interrumpir las actividades de los usuarios. No requiere modificaciones al código fuente del protocolo TCP/IP o de los sistemas operativos anfitrión y/o huésped. Este protocolo se basa en la replicación de comandos bajo rsh, los cuales son aplicados a una o varias VMs. La sincronización de paquetes durante el envío/recepción de las conversaciones replica se realiza sin pérdida de información y se verifica que los datos replicados sean consistentes en todas las RVMs. El protocolo puede ser empleado através de redes de área amplia, en cuyo caso se usó una comunicación segura configurando VPNs IPSEC. El protocolo de replicación multipunto propuesto permite que se ejecute un mecanismo de recuperación a fallas y que los datos se encuentren disponibles aún en las situaciones de falla. Este protocolo puede ser instalado en cualquier ubicación donde TCP/IP esté disponible. Los experimentos realizados mostraron que el protocolo im-

plementado es un mecanismo de aplicación factible tanto en redes locales como en redes de área amplia. En los experimentos realizados en ambiente de red ampliada se verificó que no se realiza un gran consumo del ancho de banda disponible, aún cuando existen dos réplicas remotas, el rendimiento de la red local no sufre una degradación significativa. Las conversaciones en sentido cliente-servidor y servidor-cliente no presentan un variación significativa del RTT promedio para una y dos replicas. La comunicación sufre una degradación promedio del 16 % cuando hay dos replicas activas. Sin embargo, debe tomarse en consideración que este retraso adicional en la transmisión no solamente es propiciado por el protocolo de replicación. Este cálculo se realiza en un ambiente real con un tráfico de red cambiando dinámicamente, uso continúo de las computadoras durante los experimentos, limites de ancho de banda impuesto por los proveedores de internet, retraso en el servicio entregado de internet, etc. A pesar, de que el protocolo es ejecutado en un ambiente con un servicio de internet común y no dedicado; muestra un buen desempeño que permite la ejecución de dos replicas remotas sin tener un servicio completamente exclusivo para ser usado por la conexión donde se encuentran viajando las conversaciones replicadas.

Se ha propuesto un esquema de tolerancia fallas y se han reportado los resultados obtenidos. Este mecanismo permite la recuperación automática de fallas durante el proceso de ejecución. Siendo posible que el protocolo se recupere si falla la máquina original o bien alguna de las réplicas. Según el esquema de replicación propuesto, al considerar recuperación a fallas el RTT promedio puede sufrir una degradación desde el punto de vista del cliente de hasta un 87 %, comparando el RTT promedio calculado sin replicas y el RTT cuando se encuentran activas tres VMs réplica. En los resultados podemos observar que las mediciones obtenidas de RTT promedio del cliente (RM) hacia la OVM sufrieron un incremento de 0.02699 ms, sin réplicas, hasta 0.05071 ms con tres réplicas. Sin embargo, estos tiempos aún proporcionan al sistema un adecuado nivel de desempeño y hacen posible la replicación de datos sin que se presente retransmisión de paquetes por parte del protocolo TCP/IP. De los casos de estudio realizados y su observación, se aprecia un aumento del RTT promedio debido a que el paquete emitido por el cliente, primero debe viajar hasta la máquina identifiacada como PSC y después hasta la OVM. Para que el paquete emitido por el cliente pueda ser reconocido debe emprender el camino de vuelta regresando desde la

OVM a la máquina PSC y por último al cliente, donde finalmente se reconoce el paquete. En la sección de trabajos futuros se propone una mejora al mecanismo de recuperación de fallas actual con el objetivo de mejorar el redimiento del sistema.

El protocolo de replicación hace uso de un mecanismo de recuperación a fallas que permite que una VM réplica tome el lugar de una máquina original en un RTT máximo de 5.64 ms. Lo cual, hace posible que las transacciones entre cliente y la máquina original sean ininterrumpidas. También, el mecanismo de tolerancia a fallas puede detectar la falla de una réplica y realizar los ajustes necesarios en la configuración del protocolo para que esto no afecte la operación del sistema; en un tiempo máximo de 0.003526 ms.

La implementación de mecanismos que permiten alta confiabilidad y disponibilidad puede implicar un costo en el rendimiento general del sistema debido al reprocesamiento que se genera en la replicación de datos o en los mecanismos de recuperación de fallas.

La implementación de este protocolo fue realizada en su totalidad con software libre y con equipo de hardware con características físicas estándar, mostrándose la posibilidad de obtener resultados eficientes bajo estas condiciones.

6.2. Trabajo Futuro

Tomando como referencia la investigación reportada en esta tesis, se recomienda encauzar los siguientes trabajos de investigación relacionados con los temas tratados en esta tesis:

1. Optimización del esquema de replicación para lograr un desempeño más eficiente del sistema de alta disponibilidad. Decremento del del RTT promedio durante el proceso de replicación. Se propone la reprogramación del código de netfilter para reconocer los paquetes que provienen del cliente con destino PSC y cambiar la IP fuente y destino sin realizar el procesamiento del paquete. De igual forma los paquetes detectados en la máquina intermedia con origen OVM y destino PSC, cambien a su origen por PSC y destino cliente, sin necesidad de que exista un procesamiento del paquete en la máquina intermedia. De manera similar a como iptables realiza una regla FORWARD.

2. Implementación del algoritmo de recuperación de falla de la OVM donde la VM que reemplace a la OVM sea la máquina con el menor RTT promedio registrado y no la primer RVM de la lista de réplicas.

3. Implementación de un algortimo de replicación donde no exista una máquina OVM fija. El rol de OVM será tomado por la máquina que responda más rápido.

4. Implementación del modelo matemático de comportamiento del protocolo de replicación de servicios TCP/IP.

Referencias

[Aghdaie01] Aghdaie, N. y Tamir, Y. Client-transparent fault-tolerant web service. *En 20th IEEE International Performance, Computing, and Communications Conference*, págs. 209–216. 2001.

[Aghdaie03] Aghdaie, N. y Tamir, Y. Fast Transparent Failover for Reliable Web Service. *En International Conference on Parallel and Distributed Computing and Systems*, págs. 757–762. nov. 2003.

[Aghdaie09] Aghdaie, N. y Tamir, Y. Coral: A transparent fault-tolerant web service. *J. Syst. Softw.*, 82(1):131–143, ene. 2009. ISSN 0164-1212. doi: 10.1016/j.jss.2008.06.036.
URL `http://dx.doi.org/10.1016/j.jss.2008.06.036`

[Armbrust09] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., y Zaharia, M. Above the clouds: A berkeley view of cloud computing. Inf. Téc. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.
URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`

[Armbrust10] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., y Zaharia, M. A view of cloud computing. *Commun. ACM*, 53:50–58, April 2010. ISSN 0001-0782. doi:http://doi.acm.org/10.1145/1721654.1721672.
URL `http://doi.acm.org/10.1145/1721654.1721672`

[Barham03] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., y Warfield, A. Xen and the art of virtualization. *En SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, págs. 164–177. ACM, New York, NY, USA, 2003. ISBN 1-58113-757-5. doi:http://doi.acm.org/10.1145/945445.945462.

[Bathrick99] Bathrick, G. Definitions of managed objects for the adsl lines, August 1999. URL `http://tools.ietf.org/html/rfc2662`

[Bobroff07] Bobroff, N., Kochut, A., y Beaty, K. Dynamic Placement of Virtual Machines for Managing SLA Violations. *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, págs. 119–128, mayo 2007. doi: 10.1109/INM.2007.374776.
URL `http://dx.doi.org/10.1109/INM.2007.374776`

[Bradford07] Bradford, R., Kotsovinos, E., Feldmann, A., y Schiöberg, H. Live wide-area migration of virtual machines including local persistent state. *En Proceedings of the 3rd international conference on Virtual execution environments*, VEE '07, págs. 169–179. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-630-1. doi:10.1145/1254810.1254834.
URL `http://doi.acm.org/10.1145/1254810.1254834`

[Cáceres05] Cáceres, R., Carter, C., Narayanaswami, C., y Raghunath, M. Reincarnating pcs with portable soulpads. *En Proceedings of the 3rd international conference on Mobile systems, applications, and services*, MobiSys '05, págs. 65–78. ACM, New York, NY, USA, June 2005. ISBN 1-931971-31-5. doi: 10.1145/1067170.1067179.
URL `http://doi.acm.org/10.1145/1067170.1067179`

[Chandra05] Chandra, R., Zeldovich, N., Sapuntzakis, C., y Lam, M. S. The collective: a cache-based system management architecture. *En Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, págs. 259–272. USENIX Association, Berkeley, CA, USA,

May 2005.
URL http://dl.acm.org/citation.cfm?id=1251203.1251222

[Changarti07] Changarti, P. *Xen Virtualization: A Practical Handbook*. Packt Publishing Ltd., Birmingham, B27 6PA, UK., 1ª ed$^{\text{ón}}$., 2007.

[Chaudhary08] Chaudhary, V., Cha, M., Walters, J., Guercio, S., y Gallo, S. A comparison of virtualization technologies for hpc. *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*, págs. 861–868, March 2008. ISSN 1550-445X. doi:10.1109/AINA.2008.45.

[Chen01] Chen, P. M. y Noble, B. D. When virtual is better than real. *En Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, HOTOS '01, págs. 133–. IEEE Computer Society, Washington, DC, USA, 2001.
URL http://dl.acm.org/citation.cfm?id=874075.876409

[Chidambaram08] Chidambaram, J., Prabhu, C., Narasimha Rao, P., Wankar, R., Aneesh, C., y Agarwal, A. A methodology for high availability of data for business continuity planning / disaster recovery in a grid using replication in a distributed database. *En TENCON 2008 - 2008 IEEE Region 10 Conference*, págs. 1–6. Nov. 2008. doi:10.1109/TENCON.2008.4766862.

[Chisnall07] Chisnall, D. *The definitive guide to the xen hypervisor*. Prentice Hall Press, Upper Saddle River, NJ, USA, 2007. ISBN 9780132349710.

[Chun09] Chun, B.-G. y Maniatis, P. Augmented smartphone applications through clone cloud execution. *En Proceedings of the 12th conference on Hot topics in operating systems*, HotOS'09, págs. 8–8. USENIX Association, Berkeley, CA, USA, May 2009.
URL http://dl.acm.org/citation.cfm?id=1855568.1855576

[Chun11] Chun, B.-G., Ihm, S., Maniatis, P., Naik, M., y Patti, A. Clonecloud: elastic execution between mobile device and cloud. *En Proceedings of the sixth conference on Computer systems*, EuroSys '11, págs. 301–314. ACM, New York,

NY, USA, April 2011. ISBN 978-1-4503-0634-8. doi:10.1145/1966445.1966473. URL `http://doi.acm.org/10.1145/1966445.1966473`

[Clark05] Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I., y Warfield, A. Live migration of virtual machines. *En NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, págs. 273–286. USENIX Association, Berkeley, CA, USA, 2005.

[Cully08] Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchinson, N., y Warfield, A. Remus: high availability via asynchronous virtual machine replication. *En NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, págs. 161–174. USENIX Association, Berkeley, CA, USA, 2008. ISBN 111-999-5555-22-1.

[Davies06] Davies, A. y Fisk, H. *MySQL Clustering*. MySQL Press, 2006. ISBN 0672328550.

[Davis01] Davis, C. R. *Ipsec: Securing Vpns*. McGraw-Hill Professional, 2001. ISBN 0072127570.

[Dike01] Dike, J. User-mode linux. *En Proceedings of the 5th annual Linux Showcase & Conference - Volume 5*, ALS '01, págs. 2–2. USENIX Association, Berkeley, CA, USA, 2001.
URL `http://dl.acm.org/citation.cfm?id=1268488.1268490`

[Enterprises11] Enterprises, A. Alasir enterprises. http://alasir.com/index.html, 1999-2011.

[Forouzan10] Forouzan, B. *TCP/IP Protocol Suite*. McGraw-Hill Forouzan Networking. McGraw-Hill Education, 2010. ISBN 9780073376042.
URL `http://books.google.com.mx/books?id=yNthPwAACAAJ`

[Fre]

[Gupta06] Gupta, D., Cherkasova, L., Gardner, R., y Vahdat, A. Enforcing performance isolation across virtual machines in xen. *En Proceedings of the*

ACM/IFIP/USENIX 2006 International Conference on Middleware, Middleware '06, págs. 342–362. Springer-Verlag New York, Inc., New York, NY, USA, 2006.
URL `http://dl.acm.org/citation.cfm?id=1515984.1516011`

[Hansen04] Hansen, J. G. y Jul, E. Self-migration of operating systems. *En Proceedings of the 11th workshop on ACM SIGOPS European workshop*, EW 11. ACM, New York, NY, USA, September 2004. doi: http://doi.acm.org/10.1145/1133572.1133616.
URL `http://doi.acm.org/10.1145/1133572.1133616`

[Heusse11] Heusse, M., Merritt, S. A., Brown, T. X., y Duda, A. Two-way tcp connections: old problem, new insight. *SIGCOMM Comput. Commun. Rev.*, 41(2):5–15, abr. 2011. ISSN 0146-4833. doi:10.1145/1971162.1971164.
URL `http://doi.acm.org/10.1145/1971162.1971164`

[Hines09] Hines, M. R., Deshpande, U., y Gopalan, K. Post-copy live migration of virtual machines. *SIGOPS Oper. Syst. Rev.*, 43(3):14–26, jul. 2009. ISSN 0163-5980. doi:10.1145/1618525.1618528.
URL `http://doi.acm.org/10.1145/1618525.1618528`

[Hirofuchi09] Hirofuchi, T., Nakada, H., Ogawa, H., Itoh, S., y Sekiguchi, S. A live storage migration mechanism over wan and its performance evaluation. *En Proceedings of the 3rd international workshop on Virtualization technologies in distributed computing*, VTDC '09, págs. 67–74. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-580-2. doi:10.1145/1555336.1555348.
URL `http://doi.acm.org/10.1145/1555336.1555348`

[Hirofuchi11] Hirofuchi, T., Nakada, H., Itoh, S., y Sekiguchi, S. Reactive consolidation of virtual machines enabled by postcopy live migration. *En Proceedings of the 5th international workshop on Virtualization technologies in distributed computing*, VTDC '11, págs. 11–18. ACM, New York, NY, USA, 2011. ISBN

978-1-4503-0701-7. doi:10.1145/1996121.1996125.
URL http://doi.acm.org/10.1145/1996121.1996125

[Hu08] Hu, D. y Wang, Y. Y. Teaching computer security using xen in a virtual environment. *Information Security and Assurance, 2008. ISA 2008. International Conference on*, págs. 389–392, April 2008. doi:10.1109/ISA.2008.18.

[Ibrahim11] Ibrahim, K. Z., Hofmeyr, S., Iancu, C., y Roman, E. Optimized pre-copy live migration for memory intensive applications. *En Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, págs. 40:1–40:11. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0771-0. doi:10.1145/2063384.2063437.
URL http://doi.acm.org/10.1145/2063384.2063437

[Institute81] Institute, I. S. Rfc 793-transmission control protocol, sep 1981. Edited by Jon Postel. Available at http://www.rfc-es.org/rfc/rfc0793-es.txt.
URL http://www.rfc-es.org/rfc/rfc0793-es.txt

[Jacobson92] Jacobson, V., Braden, R., y Borman, D. Tcp extensions for high performance, 1992.

[Jiang10] Jiang, B., Ravindran, B., y Kim, C. Lightweight live migration for high availability cluster service. *En Proceedings of the 12th international conference on Stabilization, safety, and security of distributed systems*, SSS'10, págs. 420–434. Springer-Verlag, Berlin, Heidelberg, 2010. ISBN 3-642-16022-0, 978-3-642-16022-6.
URL http://dl.acm.org/citation.cfm?id=1926829.1926865

[Kalampoukas98] Kalampoukas, L., Varma, A., y Ramakrishnan, K. K. Improving tcp throughput over two-way asymmetric links: analysis and solutions. *SIGMETRICS Perform. Eval. Rev.*, 26(1):78–89, jun. 1998. ISSN 0163-5999. doi:10.1145/277858.277877.
URL http://doi.acm.org/10.1145/277858.277877

[Karger84] Karger, P. A. y Herbert, A. J. An augmented capability architecture to support lattice security and traceability of access. *Security and Privacy, IEEE Symposium on*, 0:2, 1984. ISSN 1540-7993. doi: http://doi.ieeecomputersociety.org/10.1109/SP.1984.10001.

[Kistler92] Kistler, J. J. y Satyanarayanan, M. Disconnected operation in the coda file system. *ACM Trans. Comput. Syst.*, 10(1):3–25, feb. 1992. ISSN 0734-2071. doi:10.1145/146941.146942.
URL `http://doi.acm.org/10.1145/146941.146942`

[Kivity07] Kivity, A., Kamay, Y., Laor, D., Lublin, U., y Liguori, A. kvm: the Linux virtual machine monitor. *En Ottawa Linux Symposium*, págs. 225–230. jul. 2007.
URL `http://www.kernel.org/doc/ols/2007/ols2007v1-pages-225-230.pdf`

[Koch03] Koch, R. R., Hortikar, S., Moser, L. E., y Melliar-Smith, P. M. Transparent tcp connection failover. *Dependable Systems and Networks, International Conference on*, 0:383, 2003. doi: http://doi.ieeecomputersociety.org/10.1109/DSN.2003.1209949.

[Kozuch02] Kozuch, M. y Satyanarayanan, M. Internet suspend/resume. *En WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, pág. 40. IEEE Computer Society, Washington, DC, USA, 2002. ISBN 0-7695-1647-5.

[Lagar-Cavilla09] Lagar-Cavilla, H. A., Whitney, J. A., Scannell, A. M., Patchin, P., Rumble, S. M., de Lara, E., Brudno, M., y Satyanarayanan, M. Snowflock: rapid virtual machine cloning for cloud computing. *En Proceedings of the 4th ACM European conference on Computer systems*, EuroSys '09, págs. 1–12. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-482-9. doi:10.1145/1519065.1519067.
URL `http://doi.acm.org/10.1145/1519065.1519067`

[Libnet] Libnet. Libnet home page.

[Libpcap] Libpcap. Tcpdump/libpcap public repository.

[LINBIT11] LINBIT. Distributed replicated block device, Sep 2011.
URL `http://www.drbd.org/`

[Machine12] Machine, K. B. V. Main page - kvm, Sep 2012.
URL `http://www.linux-kvm.org/page/Main_Page`

[Marston09] Marston, S. R., Li, Z., Bandyopadhyay, S., Ghalsasi, A., y Zhang, J. Cloud Computing: The Business Perspective. *SSRN eLibrary*, 2009.

[Mel09] The nist definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.
URL `http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc`

[Milojíčić00] Milojíčić, D. S., Douglis, F., Paindaveine, Y., Wheeler, R., y Zhou, S. Process migration. *ACM Comput. Surv.*, 32(3):241–299, 2000. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/367701.367728.

[Mishra98] Mishra, S. y Wu, L. An evaluation of flow control in group communication. *IEEE/ACM Trans. Netw.*, 6(5):571–587, oct. 1998. ISSN 1063-6692. doi: 10.1109/90.731193.
URL `http://dx.doi.org/10.1109/90.731193`

[Morris03] Morris, R. J. T. y Truskowski, B. J. The evolution of storage systems. *IBM Systems Journal*, 42(2):205 –217, 2003. ISSN 0018-8670. doi:10.1147/sj.422.0205.

[Mullender90] Mullender, S. J., van Rossum, G., Tanenbaum, A. S., van Renesse, R., y van Staveren, H. Amoeba: A distributed operating system for the 1990s. *Computer*, 23:44–53, May 1990. ISSN 0018-9162. doi:http://dx.doi.org/10.1109/2.53354.
URL `http://dx.doi.org/10.1109/2.53354`

[MySQL] MySQL. Mysql 5.0 reference manual.

[Nathuji07] Nathuji, R. y Schwan, K. Virtualpower: coordinated power management in virtualized enterprise systems. *SIGOPS Oper. Syst. Rev.*, 41(6):265–278, oct.

2007. ISSN 0163-5980. doi:10.1145/1323293.1294287.
URL http://doi.acm.org/10.1145/1323293.1294287

[Nelson05] Nelson, M., Lim, B.-H., y Hutchins, G. Fast transparent migration for virtual machines. *En ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, págs. 25–25. USENIX Association, Berkeley, CA, USA, 2005.

[Nieh05] Nieh, J. Autopod: Unscheduled system updates with zero data loss. *En Proceedings of the Second International Conference on Automatic Computing*, págs. 367–368. IEEE Computer Society, Washington, DC, USA, 2005. ISBN 0-7965-2276-9. doi:10.1109/ICAC.2005.16.
URL http://dl.acm.org/citation.cfm?id=1078027.1078504

[Oracle09] Oracle. An oracle white paper. virtualization with oracle solaris 10. http://www.oracle.com/technetwork/server-storage/solaris10/solaris-10-virtualization-167782.pdf, June 2009.

[Osman02] Osman, S., Subhraveti, D., Su, G., y Nieh, J. The design and implementation of zap: a system for migrating computing environments. *SIGOPS Oper. Syst. Rev.*, 36:361–376, December 2002. ISSN 0163-5980. doi: http://doi.acm.org/10.1145/844128.844162.
URL http://doi.acm.org/10.1145/844128.844162

[Podlesny12] Podlesny, M. y Williamson, C. Improving tcp performance in residential broadband networks: a simple and deployable approach. *SIGCOMM Comput. Commun. Rev.*, 42(1):61–68, ene. 2012. ISSN 0146-4833. doi: 10.1145/2096149.2096158.
URL http://doi.acm.org/10.1145/2096149.2096158

[Popek73] Popek, G. J. y Goldberg, R. P. Formal requirements for virtualizable third generation architectures. *SIGOPS Oper. Syst. Rev.*, 7(4):121, 1973. ISSN 0163-5980. doi:http://doi.acm.org/10.1145/957195.808061.

[RFC81] Rfc 791 internet protocol - darpa inernet programm, protocol specification, September 1981.

[Rodríguez09] Rodríguez, A., Caro, A., y Fernández-Medina, E. Towards framework definition to obtain secure business process from legacy information systems. *En Proceedings of the first international workshop on Model driven service engineering and data quality and security*, MoSE+DQS '09, págs. 17–24. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-816-2. doi: 10.1145/1651415.1651419.
URL `http://doi.acm.org/10.1145/1651415.1651419`

[Rosenblum04] Rosenblum, M. The reincarnation of virtual machines. *Queue*, 2(5):34–40, 2004. ISSN 1542-7730. doi:http://doi.acm.org/10.1145/1016998.1017000.

[Rosenblum05] Rosenblum, M. y Garfinkel, T. Virtual machine monitors: Current technology and future trends. *Computer*, 38(5):39–47, mayo 2005. ISSN 0018-9162. doi:10.1109/MC.2005.176.
URL `http://dx.doi.org/10.1109/MC.2005.176`

[Rozier91] Rozier, M., Abrossimov, V., Armand, F., Boule, I., Gien, M., Guillemont, M., Herrmann, F., Kaiser, C., Langlois, S., Léonard, P., y Neuhauser, W. Overview of the chorus distributed operating systems. *Computing Systems*, 1:39–69, 1991.

[Sapuntzakis02] Sapuntzakis, C. P., Chandra, R., Pfaff, B., Chow, J., Lam, M. S., y Rosenblum, M. Optimizing the migration of virtual computers. *En In Proceedings of the 5th Symposium on Operating Systems Design and Implementation*, págs. 377–390. December 2002.

[Satyanarayanan05] Satyanarayanan, M., Kozuch, M. A., Helfrich, C. J., y O'Hallaron, D. R. Towards seamless mobility on pervasive hardware. *Pervasive Mob. Comput.*, 1(2):157–189, jul. 2005. doi:10.1016/j.pmcj.2005.03.005.
URL `http://dx.doi.org/10.1016/j.pmcj.2005.03.005`

[Satyanarayanan07] Satyanarayanan, M., Gilbert, B., Toups, M., Tolia, N., O'Hallaron, D., Surie, A., Wolbach, A., Harkes, J., Perrig, A., Farber, D., Kozuch, M., Helfrich, C., Nath, P., y Lagar-Cavilla, H. Pervasive personal computing in an internet suspend/resume system. *Internet Computing, IEEE*, 11(2):16–25, March-April 2007. ISSN 1089-7801. doi:10.1109/MIC.2007.46.

[Satyanarayanan09] Satyanarayanan, M., Bahl, P., Caceres, R., y Davies, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, oct. 2009. ISSN 1536-1268. doi:10.1109/MPRV.2009.82.
URL `http://dx.doi.org/10.1109/MPRV.2009.82`

[Shao08] Shao, Z., Jin, H., Cheng, B., y Jiang, W. Er-tcp: an efficient tcp fault-tolerance scheme for cluster computing. *The Journal of Supercomputing*, págs. 127–145, 2008.

[Shenoy00] Shenoy, G., Satapati, S. K., y Bettati, R. Hydranet-ft: Network support for dependable services. *En Proceedings of the The 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, ICDCS '00, págs. 699–. IEEE Computer Society, Washington, DC, USA, 2000. ISBN 0-7695-0601-1.
URL `http://dl.acm.org/citation.cfm?id=850927.851754`

[Smith05] Smith, J. y Nair, R. The architecture of virtual machines. *Computer*, 38(5):32–38, May 2005. ISSN 0018-9162. doi:10.1109/MC.2005.173.

[SourceForge] SourceForge. Open source tripwire.

[Sripanidkulchai10] Sripanidkulchai, K., Sahu, S., Ruan, Y., Shaikh, A., y Dorai, C. Are clouds ready for large distributed applications? *SIGOPS Oper. Syst. Rev.*, 44(2):18–23, abr. 2010. ISSN 0163-5980. doi:10.1145/1773912.1773918.
URL `http://doi.acm.org/10.1145/1773912.1773918`

[Sultan02] Sultan, F., Srinivasan, K., Iyer, D., y Iftode, L. Migratory tcp: Connection migration for service continuity in the internet. *En ICDCS'02*, págs. 469–470. 2002.

[Tanenbaum95] Tanenbaum, A. S. *Distributed operating systems*. Prentice Hall, 1995. ISBN 978-0-13-143934-4.

[Tanenbaum01] Tanenbaum, A. S. y Steen, M. V. *Distributed Systems: Principles and Paradigms*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1ª ed$^{\underline{\text{ón}}}$., 2001. ISBN 0130888931.

[Torres08] Torres, J., Carrera, D., Hogan, K., Gavalda, R., Beltran, V., y Poggi, N. Reducing wasted resources to help achieve green data centers. *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, págs. 1–8, April 2008. ISSN 1530-2075. doi:10.1109/IPDPS.2008.4536219.

[Travostino06] Travostino, F. Seamless live migration of virtual machines over the man/wan. *En Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, SC '06. ACM, New York, NY, USA, 2006. ISBN 0-7695-2700-0. doi: 10.1145/1188455.1188758.
URL `http://doi.acm.org/10.1145/1188455.1188758`

[Uhlig05] Uhlig, R., Neiger, G., Rodgers, D., Santoni, A., Martins, F., Anderson, A., Bennett, S., Kagi, A., Leung, F., y Smith, L. Intel virtualization technology. *Computer*, 38(5):48–56, May 2005. ISSN 0018-9162. doi:10.1109/MC.2005.163.

[Vaughan-Nichols06] Vaughan-Nichols, S. New approach to virtualization is a lightweight. *Computer*, 39(11):12–14, Nov. 2006. ISSN 0018-9162. doi: 10.1109/MC.2006.393.

[VMWare] VMWare. Historia de la virtualización.

[VMware12] VMware. Vmware gsx server documentation, 2012.
URL `http://www.vmware.com/support/pubs/gsx_pubs.html`

[Waldspurger02] Waldspurger, C. A. Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36:181–194, December 2002. ISSN 0163-5980. doi: http://doi.acm.org/10.1145/844128.844146.
URL `http://doi.acm.org/10.1145/844128.844146`

[William I. Bullers06] William I. Bullers, J., Burd, S., y Seazzu, A. F. Virtual machines - an idea whose time has returned: application to network, security, and database courses. *SIGCSE Bull.*, 38(1):102–106, 2006. ISSN 0097-8418. doi: http://doi.acm.org/10.1145/1124706.1121375.

[Williams12] Williams, D., Jamjoom, H., y Weatherspoon, H. The xen-blanket: virtualize once, run everywhere. *En Proceedings of the 7th ACM european conference on Computer Systems*, EuroSys '12, págs. 113–126. ACM, New York, NY, USA, 2012. ISBN 978-1-4503-1223-3. doi:10.1145/2168836.2168849.
URL `http://doi.acm.org/10.1145/2168836.2168849`

[Wood07] Wood, T., Shenoy, P., Venkataramani, A., y Yousif, M. Black-box and gray-box strategies for virtual machine migration. *En Proceedings of the 4th USENIX conference on Networked systems design & implementation*, NSDI'07, págs. 17–17. USENIX Association, Berkeley, CA, USA, April 2007.
URL `http://dl.acm.org/citation.cfm?id=1973430.1973447`

[Wood11] Wood, T., Ramakrishnan, K. K., Shenoy, P., y van der Merwe, J. Cloudnet: dynamic pooling of cloud resources by live wan migration of virtual machines. *SIGPLAN Not.*, 46(7):121–132, mar. 2011. ISSN 0362-1340. doi: 10.1145/2007477.1952699.
URL `http://doi.acm.org/10.1145/2007477.1952699`

[Xen.org11] Xen.org. Xen.org. http://www.xen.org/, 2005-2011.

[Zagorodnov09] Zagorodnov, D., Marzullo, K., Alvisi, L., y Bressoud, T. C. Practical and low-overhead masking of failures of tcp-based servers. *ACM Trans. Comput. Syst.*, 27:4:1–4:39, May 2009. ISSN 0734-2071. doi: http://doi.acm.org/10.1145/1534909.1534911.
URL `http://doi.acm.org/10.1145/1534909.1534911`