Tuning de una Red de Clientes Ultralivianos para ambientes de desarrollo JAVA

F. Javier Díaz, Viviana M. Ambrosi[1], Paula Venosa,
Juan P. Giecco, Adrián Pousa, Nicolás Alonso
Laboratorio de investigación en nuevas tecnologías informáticas (LINTI)
Facultad de Informática – UNLP
TEL +54+221+4223528
{jdiaz, vambrosi, pvenosa@info.unlp.edu.ar}, jpgiecco@gmail.com, apousa@lidi.info.unlp.edu.ar, nfalonso@gmail.com

Abstract

Este artículo presenta el tuning de una red de clientes ultralivianos utilizados para ambientes de desarrollo JAVA, con tecnología SUN[2], en el marco del proyecto COE[3].

Se investigó la performance en dicho entorno, analizando el backend de los clientes ultralivianos (SunRays), de las aplicaciones, la red, así como los parámetros que afectan su rendimiento. Se estudiaron alternativas para mejorar la performance en general y se realizaron ajustes en consecuencia que implicaron cambios en la topología de la red, en los servicios y en su configuración.

Nuestra investigación muestra como es posible lograr un desempeño aceptable a partir de este tipo de tecnología, partiendo de un uso convencional de la misma y luego extendiéndolo de manera tal de permitir el desarrollo de software por parte de los alumnos de la Facultad de Informática de la Universidad Nacional de La Plata, cumpliendo con los requerimientos que ello impone.

Keywords:

Tuning, clientes ultralivianos, SunRay, SRSS[4], JAVA, Jdeveloper, Eclipse, LDAP, COE, grupo de fallas, balanceo de carga.

Introducción

En la Facultad de Informática de la UNLP se ha instalado en el año 2003, en el marco del Proyecto COE [1], un centro de excelencia en Web Services que consta de una red de clientes ultralivianos con tecnología SUN [2].

El mismo fue originalmente utilizado para e-learning, y luego extendido su uso a actividades de desarrollo de software, especialmente en entornos JAVA.

Utilizar una sala de clientes ultralivianos [3] presenta varias ventajas y desventajas. Las ventajas están relacionadas a la seguridad, administración y escalabilidad [4] [5].

Es difícil la determinación de la performance de las aplicaciones que usan arquitecturas de clientes livianos durante su diseño [6].

Poder utilizar este tipo de arquitecturas en ambientes de desarrollo JAVA presenta la problemática de la exigencia del consumo de recursos por parte de dichos entornos. El servidor debe ser capaz de atender a todas las estaciones durante la codificación, compilación, debugging y running de aplicaciones para cada uno de los proyectos, sin que su rendimiento sea afectado, y evitando caídas de la red que podrían provocar pérdida de información.

Plantear una adecuada distribución de los recursos tanto físicos como lógicos es una tarea que debe llevarse a cabo para alcanzar la performance deseada, así como realizar una estimación de

[1] Profesional Principal CIC-BA
[2] SUN Microsystems es una empresa informática de Silicon Valley, fabricante de semiconductores y software
[3] COE: Centro de Excelencia de Web Services de SUN
[4] SRSS: SunRay Server Software

la cantidad de estaciones de trabajo que pueden ser utilizadas simultáneamente sin afectar el rendimiento total.

Contar con un software de soporte que administre las sesiones desde las estaciones de trabajo es de vital importancia, siendo imprescindible implementar un grupo de fallas y balanceo de carga entre los servidores, características ambas provistas por el servidor propio de las SunRays [7].

Por otro lado debe ser posible manejar la autenticación de usuarios (alumnos/docentes), dotarlos de movilidad, teniendo en cuenta el almacenamiento de información y su posterior accesibilidad.

Para evaluar cuál es la configuración óptima de nuestra red de clientes ultralivianos, determinando la cantidad de estaciones que pueden ser atendidas simultáneamente con una performance razonable, resulta imprescindible analizar el rendimiento del entorno y efectuar mediciones en el tráfico de la red para detectar posibles cuellos de botella [8].

Dado que las plataformas de clientes livianos están diseñadas de forma muy diferente a los sistemas de escritorio tradicionales, medir su performance es una tarea complicada. Algunos factores pueden influenciar en la comparación de performance, a saber: mecanismos opcionales de los protocolos de clientes livianos, aplicaciones que se ejecutan en el servidor, ancho de banda de la red y tipo de tráfico que comparte el ancho de banda [9]

Arquitectura básica de la sala a ser tuneada

El Laboratorio cuenta en la actualidad con 19 clientes ultralivianos SunRay conectados a tres (3) servidores SUN FIRE V280 desde los cuales inician sesión. El tipo de conexión entre los servidores y los clientes es uno de los primeros aspectos a ser ajustados. Las pruebas iniciales fueron realizadas utilizando un enlace Ethernet de 100 Mb. La capacidad del enlace resultó insuficiente, debiéndose utilizar un enlace de fibra óptica de 1 Gb.

Los servidores están compuestos por dos procesadores de alta performance UltraSPARC III de 1.2GHz, 4GB de RAM y sistema operativo Solaris 10.

Las SunRay (clientes ultralivianos) no tienen capacidad de procesamiento ni de almacenamiento, el sistema es provisto por los servidores a través de un software propietario de SUN llamado SunRay Server Software (SRSS) [10].

Los tres (3) servidores SRSS se encuentran formando un grupo de fallas que permiten restaurar el sistema en caso de caída de alguno de ellos, además de proveer balanceo de carga.

Autenticación

Un problema propio del servidor SRSS es el hecho de no proveer un manejo adecuado de usuarios cuando existe más de un servidor atendiendo a las estaciones, quedando cada usuario ligado al servidor en el cual han iniciado sesión. Debe recurrirse a la administración de usuarios provista por el sistema operativo.

A fin de lograr una administración centralizada, es necesario dotar a la sala de un servidor LDAP[5] (Sun Java System Directory Server) [11]; configurando al sistema operativo de todos los servidores y al SRSS para que permitan autenticación mediante LDAP. De esta manera se logra que cada usuario sea creado sólo una vez, en un repositorio centralizado, y posteriormente reconocido por el resto de servidores.

Esta solución debe ser acompañada de un servidor LDAP secundario para formar un grupo de fallos, evitando de esta forma que los usuarios queden imposibilitados de utilizar las estaciones de trabajo ante la caída de un único servidor.

[5] LDAP: Lightweight Directory Access Protocol

Movilidad

Una característica propia de las SunRays es la posibilidad de la movilidad de usuarios. La misma puede ser considerada como física y/o lógica.

La movilidad física es cuando un usuario cambia de máquina, se mueve de lugar y se autentica ante el SRSS migrando la sesión de trabajo sin inconvenientes, en forma intacta y transparente.

La movilidad lógica es cuando un usuario no cambia de máquina pero si cambia el servidor que lo esta atendiendo, ya sea por la caída de un servidor o por balanceo de carga [12] [13].

Dado el uso del laboratorio COE, es deseable contar con estas características. Ambas deben ser lo más transparentes posibles para el usuario y mantener el estado de su sesión.

Esta son otras de las cuestiones que deben ser ajustadas y están íntimamente relacionadas al almacenamiento.

Almacenamiento

Ajustes relacionadas al almacenamiento deben ser considerados, tanto de ubicación de la información como de optimización del espacio, evitando replicaciones innecesarias debidas a las características propias de los clientes ultralivianos.

La información generada por los usuarios debe estar disponible independientemente del servidor que lo esté atendiendo en un determinado momento. Debe existir un espacio único de trabajo y conocido por todos los servidores. Con NFS[6] puede ser logrado [14].

Ese espacio, el "home directory" de cada usuario, reside en uno de los servidores y es compartido mediante NFS con los restantes servidores del grupo de fallas. De esta forma independientemente del servidor que le sea asignado por el software de administración de las estaciones, un usuario siempre tendrá disponible la información guardada en su home directory.

NFS también permite compartir programas, reduciendo el espacio de almacenamiento requerido para los usuarios, evitando duplicaciones innecesarias de software. Los mismos son instalados en una única ubicación y compartidos por el resto de los servidores. Con ésto es logrado otro ajuste: la centralización de la administración de aplicaciones, facilitando su mantenimiento y optimizando el consumo de recursos.

Análisis de rendimiento

Una vez ajustados los parámetros anteriormente citados debe considerarse la cantidad de clientes ultralivianos que pueden ser atendidos simultáneamente y el comportamiento de la red.

A tal efecto se realizaron mediciones para estimar el comportamiento de la red, partiendo de un (1) servidor SRSS e iniciando paulatinamente sesiones en las estaciones de trabajo (SunRays). Posteriormente se realizaron las mismas mediciones agregando un servidor secundario y un terciario con balanceo de carga durante la codificación, compilación, debugging y running de aplicaciones .

El objetivo a ser logrado es una mejor distribución de recursos en un ambiente de desarrollo Java, a partir del análisis del consumo de: CPU, memoria RAM y de las herramientas de desarrollo utilizadas.

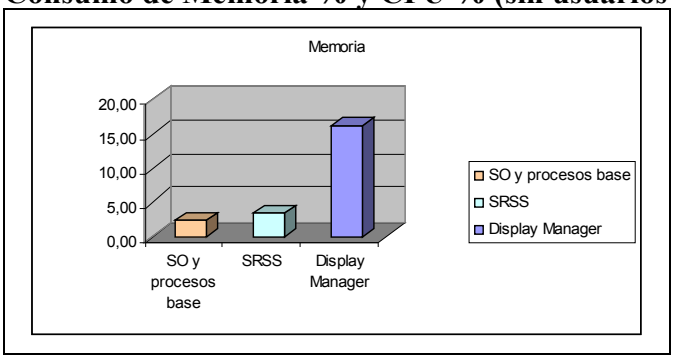
A continuación se muestran los resultados de las mediciones

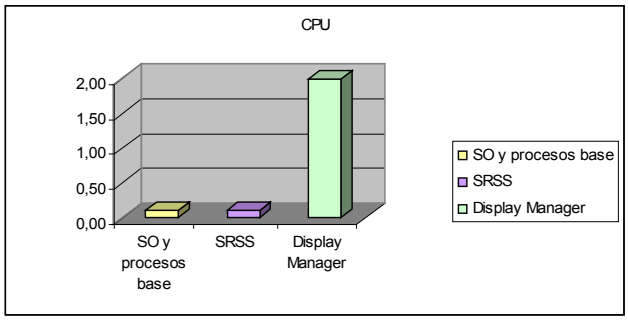
Análisis del consumo de CPU y RAM

Las mediciones iniciales se realizaron conectando los clientes ultralivianos a un único servidor. Se utilizó como herramienta PRSTAT [15] provista por el sistema operativo SOLARIS 10 [16] y el mismo arrojó los siguientes resultados:

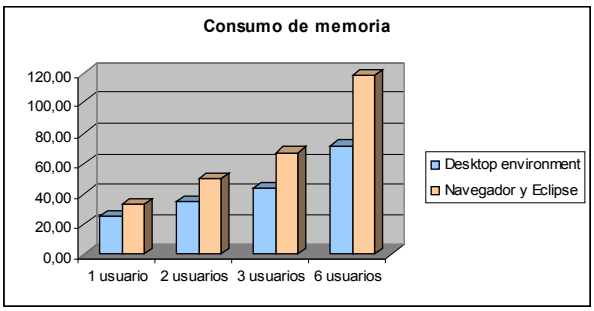
[6] NFS: Network File System

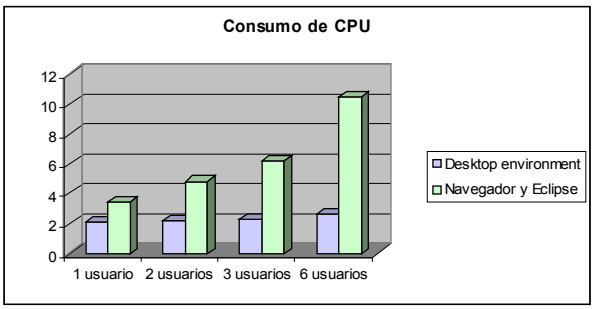
Consumo de Memoria % y CPU % (sin usuarios conectados):





Consumo de Memoria % y CPU % (con usuarios conectados):





De los datos obtenidos en las mediciones se observa que:

1. El Display Manager consume gran cantidad de recursos.
2. Teniendo un solo servidor, hasta cinco (5) sesiones pueden ser atendidas.
3. Seis (6) sesiones en desarrollo de software superan la capacidad de memoria física. Las mismas pueden ser atendidas debido al uso de memoria "swap" por parte del sistema operativo.

En nuestro caso particular, para lograr un rendimiento aceptable de la red de 19 clientes ultralivianos, debió crearse un grupo de fallas y balanceo de carga de tres (3) servidores.

Análisis de herramientas de desarrollo JAVA

Fueron realizadas mediciones durante los cursos de desarrollo de software en JAVA para calcular el consumo de recursos usando las siguientes herramientas:

- Eclipse (IDE para desarrollo JAVA).
- Jdeveloper (IDE para desarrollo JAVA).
- NetBeans (IDE para desarrollo JAVA).

La estimación del consumo se ve afectada por las acciones de cada usuario, ya sea la utilización de otro software, un navegador, procesadores de texto o la ejecución del programa en desarrollo, incrementando la cantidad de recursos consumidos.

Pudo observarse durante el análisis que, la cantidad de memoria utilizada por cada usuario variaba entre 400MB y 1GB, con valor promedio de 500MB. El pico más alto de consumo se producía cuando el usuario, además de tener abierto el ambiente de desarrollo, realizaba la ejecución de su propia aplicación contando con otros programas abiertos.

La suma de las capacidades de los tres (3) servidores del grupo de fallas es de 12GB, lo cual permite atender a las 19 SunRays con un consumo promedio total de 9.5 GB de RAM.

Cabe mencionar que si ocurre un fallo en un servidor y la carga es distribuida entre los restantes servidores, todos los usuarios del sistema se verán afectados por una notable disminución en la performance.

Se pudo verificar que el rendimiento del sistema caía considerablemente cuando un usuario tenía abiertas más de una (1) instancia del ambiente de desarrollo.

Conclusiones generales

Utilizar una red de clientes ultralivianos para desarrollo de software puede ser logrado mediante un tuning adecuado.

Debe tenerse en cuenta que los entornos basados en JAVA consumen una cantidad importante de recursos. Debe poseerse una clara idea de la cantidad recursos que son requeridos por cada cliente, la cantidad total de servidores y los recursos necesarios para cada uno de ellos, lo cual dependerá de los distintos requerimientos de desarrollo.

Los conceptos de transparencia y disponibilidad son muy importantes. Una visión única del sistema, con autenticación y disponibilidad, independientemente del servidor que lo esté atendiendo. Todo esto es posible integrando el software SSRS, LDAP, NFS y SOLARIS 10 como sistema operativo, sin dejar de lado una configuración adecuada de los servicios, el grupo de fallas, el balanceo de carga y sumado a esto el buen manejo de "swap" por parte de SOLARIS.

Pero existe un factor impredecible: el usuario. De él dependerá el consumo de recursos en un momento determinado.

Referencias:

[1]Información sobre el convenio UNLP – SUN Microsystems: http://coe.info.unlp.edu.ar/COE/Documentos/Convenio/

[2] Página oficial de SUN Microsystems: http://www.sun.com

[3] Definición de cliente liviano: http://en.wikipedia.org/wiki/Thin_client

[4] http://www.eservercomputing.com/iseries/articles/index.asp?id=702

[5] Informe clientes ultralivianos: http://coe.info.unlp.edu.ar/COE/Documentos/clienteslivianos.pdf

[6] "An environment for automated performance evaluation of J2EE and ASP.NET thin-client architectures" -Grundy, J.; Wei, Z.; Nicolescu, R.; Cai, Y.; Software Engineering Conference, 2004. Proceedings. 2004 Australian 2004 Page(s):300 – 308

[7] Información sobre Sunray: http://www.sun.com/software/sunray/index.jsp.

[8] "The Performance of Remote Display Mechanisms for Thin-Client Computing" - S. Jae Yang, Jason Nieh, Matt Selsky, and Nikhil Tiwari; Proceedings of the General Track of the annual conference on USENIX Annual Technical Conference table of contents - Year of Publication: 2002 - ISBN:1-880446-00-6

[9] "Remote Didactic Laboratory ?G. Savastano,? The Italian Experience for E-Learning at the Technical Universities in the Field of Electrical and Electronic Measurement: Architecture and Optimization of the Communication Performance Based on Thin Client Technology"- Andria, G.; Baccigalupi, A.; Borsic, M.; Carbone, A.P.; Daponte, P.; De Capua, C.; Ferrero, A.; Grimaldi, D.; Liccardo, A.; Locci, N.; Lanzolla, A.M.L.; Macii, D.; Muscas, C.; Peretto, L.; Petri, D.; Rapuano, S.; Riccio, M.; Salicone, S.; Stefani, F.;Instrumentation and Measurement, IEEE Transactions on Volume 56, Issue 4, Aug. 2007 - Digital Object Identifier 10.1109/TIM.2007.899983

[10] Información sobre SunRay Server Software: http://www.sun.com/software/sunray/index.jsp

[11] Información sobre servidor de directorios de SUN Microsystems: http://www.sun.com/software/products/directory_srvr_ee/dir_srvr/index.xml.

[12] Informe de autenticación sobre las Sunrays mediante LDAP: http://coe.info.unlp.edu.ar/COE/Documentos/informe_SRSS_LDAP.pdf

[13] Información sobre sesiones móviles en Sunrays: http://docs.sun.com/source/817-6806/non-scms.html.

[14] Conectividad LDAP – NFS: http://coe.info.unlp.edu.ar/COE/Documentos/acceso_LDAP_NFS.pdf

[15] Información sobre el comando PRSTAT: http://docs.sun.com/source/819-1892-12/FMASolarisCmd.html .

[16] Información sobre Solaris 10: http://www.sun.com/software/solaris/index.jsp .