Año: 2018 Volúmen: 1
Número: 4 ISSN: 2448-623X

Cluster de Alta Disponibilidad con Servidores Linux

J. I. Vega-Luna[1], J. F. Cosme-Aceves[1], G. Salgado-Guzmán[1], F. J. Sánchez-Rangel[1], D. Contreras-Tovar[1].

***Resumen*—Se presenta el diseño e implantación de un cluster de alta disponibilidad integrado por dos servidores y un arreglo de discos compartido. El objetivo fue desarrollar una solución que proporcione alta disponibilidad en el acceso a una base de datos Oracle en caso de fallar el servidor Linux donde se esté ejecutando. La solución consistió en tener redundancia del servidor configurando un cluster de dos nodos y redundancia en los puertos de red y tarjetas controladoras de disco de cada servidor. La base de datos reside en un arreglo de discos conectado a los servidores del cluster. Se implantaron cinco módulos software que realizan la administración del cluster, el monitoreo de puertos de red, puertos de tarjetas controladoras y aplicación, así la interfaz gráfica. Se logró un tiempo de conmutación de la base de datos de 5 minutos. Esto es, en caso de contingencia del nodo donde se ejecuta la base de datos, ésta tarda en arrancar en el nodo alterno del cluster en 5 minutos.**

***Palabras claves*—Cluster, contingencia, alta disponibilidad, base de datos, Linux, Oracle.**

***Abstract*—The design and implementation of a high availability cluster composed of two servers and a shared disk array is presented. The objective was to develop a solution that provides high availability in access to an Oracle database in case of failure of the Linux server where it is running. The solution consisted in having server redundancy by configuring a cluster of two nodes and redundancy in the network ports and disk controller cards of each server. The database resides in a disk array connected to the cluster servers. Five software modules were implemented to manage the cluster, monitor network ports, controller card ports and application, as well as the graphical interface. A database switching time of 5 minutes was achieved. That is, in case of contingency of the node where the database is executed, it takes time to start in the alternate node of the cluster in 5 minutes.**

***Keywords*—Cluster, contingency, data base, high availability, Linux, Oracle.**

I. Introducción

Con la rápida evolución de la Internet y la necesidad de tener disponible el acceso a sistemas cómputo en todo momento, las organizaciones se ven obligadas a contar con un alto nivel de confiabilidad y seguridad en su operación, ubicando los equipos de cómputo, telecomunicaciones y almacenamiento en un centro de datos para garantizar la continuidad del servicio [1].

A pesar de encontrarse los equipos en un centro de datos, la operación del mismo y aplicaciones puede verse afectado por diferentes factores, como por ejemplo: fenómenos naturales como inundaciones, huracanes y sismos, fuego, falla en el suministro de energía eléctrica, terrorismo, fallas de hardware y software, errores humanos o asuntos legales.

Ante esto, lo importante es contar con una estrategia para la protección de información, del hardware y software crítico para el restablecimiento y continuidad de las operaciones del negocio. De esta manera, se puede responder lo más rápidamente posible a la interrupción de los servicios usando diferentes técnicas y metodologías para continuar trabajando con las aplicaciones importantes o críticas del negocio. Los esquemas de protección y recuperación van desde los más básicos, que consisten en eliminar los llamados puntos de falla simples (Single Point of Failure-SPOF), hasta la implantación de planes de recuperación de desastres (Disaster Recovery Plan-DRP), con los cuales se dispone de centros de datos alternos y redundantes [2].

Un SPOF es cualquier elemento hardware o software de un sistema de cómputo cuya falla cause que las aplicaciones de usuarios no se encuentren disponibles durante un cierto tiempo de inactividad, denominado downtime en inglés. La solución lógica para eliminar los SPOF es contar con redundancia en los componentes que representan el SPOF. La redundancia implica costos adicionales, por lo que se debe evaluar la inversión contra los riegos que implica el downtime.

Un esquema de protección y recuperación inicia con tener redundancia en servidores de cómputo, en equipos de almacenamiento y acceso a la red de datos. En cada servidor existen tres componentes básicos que representan un SPOF, estos son: los puertos de red, los puertos de tarjetas controladoras de disco y el servidor mismo. Para eliminar el SPOF de los primeros dos componentes se configura redundancia en los mismos [3], mientras que para eliminar el SPOF que representa el servidor, se instala y configura un cluster de computadoras con acceso compartido a la información y aplicaciones de la empresa. Bajo este esquema, las aplicaciones se pueden ejecutar en cualquier computadora del cluster. Si alguna computadora del cluster falla, las aplicaciones que se encontraban ejecutándose en ella arrancarán en las computadoras restantes del cluster, con el objetivo de seguir prestando el servicio al usuario. Esto constituye un sistema de cómputo altamente disponible, donde la falla de uno de sus

[1]*Universidad Autónoma Metropolitana Azcapotzalco, Av. San Pablo No.180 Col. Reynosa Tamaulipas C.P. 02200. Delegación Azcapotzalco. Ciudad de México, CDMX México. Tel:(52 55) 5318-9000*

José Ignacio Luna Vega (vlji@correo.azc.uam.mx)

Año: 2018 Volúmen: 1
Número: 4 ISSN: 2448-623X

componentes interrumpe el funcionamiento del sistema solo por un periodo de tiempo breve [4].

Existen en el mercado soluciones y productos de diferentes proveedores para configurar e implantar un cluster de computadoras y sistema operativo Linux, algunas se utilizan en una distribución específica de Linux como Red Hat [5] y otras son para las distribuciones más usadas como Fedora o SuSe. Todas las soluciones tienen costo, sirven para proporcionar alta disponibilidad en los servidores del cluster y no monitorean los puertos de HBA. El costo de la solución y servicios de consultoría para configurarla es muy elevado, siendo a veces más alto que el mismo sistema operativo. Adicionalmente, para contar con una aplicación altamente disponible, el usuario debe adquirir un producto llamado toolkit, que sirve para una aplicación específica y pagar por la consultoría para instalarlo y configurarlo [6]. Aún más, estos productos ofrecen una solución básica y limitada de alta disponibilidad que carece de funcionalidades necesarias en ambientes de misión crítica.

Se encuentran también disponibles soluciones de código abierto para configurar un cluster que proporcionan alta disponibilidad en nodos del cluster, no monitorean puertos de HBA, puertos de red ni los procesos de la aplicación [7].

El objetivo de este trabajo fue implantar un cluster de computadoras altamente disponible para pequeñas y medianas organizaciones cuya plataforma de cómputo esté integrada por servidores con sistema operativo Linux y requieran prestar un servicio de manera continua. Se eligió el sistema operativo Linux porque en la actualidad es el más usado en servidores empresariales, cuenta con soporte de todos los fabricantes de servidores, tiene gran aceptación en la industria por su robustez y es mucho más barato que otros, como UNIX, siendo inclusive algunas distribuciones gratuitas. El cluster está compuesto por dos servidores y un arreglo de discos compartido. Cuenta con las siguientes características que lo hacen altamente disponible: redundancia y monitoreo de puertos de red y puertos de tarjetas controladoras de disco o HBA (Host Bus Adapter), soporte de LVM V2 (Logical Volume Manager) para formatear los discos compartidos del cluster, monitoreo de estado de la aplicación e interfaz gráfica para administración.

Los trabajos realizados durante los últimos años con clusters de computadoras no han tenido como objetivo desarrollar soluciones de cómputo altamente disponibles, eficientes y de bajo costo, sino que han usado soluciones existentes para crear clusters y utilizarlos para diferentes propósitos. Algunos desarrollos han tenido como objetivo balancear la carga entre servidores del cluster para obtener mejor desempeño de la aplicación [8], acelerar el acceso simultáneo desde varios nodos de un cluster a una base de datos [9] o a sistemas de archivos para obtener alto rendimiento a bajo costo [10]. Otras investigaciones se han centrado en crear algoritmos para migrar en línea máquinas virtuales entre los servidores de un cluster, balancear cargas de trabajo y reducir el consumo de energía eléctrica, lo cual se está usando bastante en ambientes de nubes de cómputo [11]-[12]. En el campo de big data, donde es imperativo el uso de alta disponibilidad de sistemas de cómputo, se han realizado trabajos para minimizar el impacto de queries no completados a una base de datos al fallar el servidor desde el cual se estaba accediendo la base de datos. Estos trabajos consisten en implantar puntos de control o checkpoints parciales selectivos [13].

La contribución del trabajo aquí presentado es proporcionar un sistema de cómputo altamente disponible para pequeñas y medianas organizaciones que les permita proporcionar servicio a usuarios, clientes y asociados en caso de contingencia al presentarse un evento no planeado como por ejemplo la falla de un servidor o un componente del mismo. Es útil también cuando sea necesario liberar un servidor para realizar actividades planeadas como tareas de mantenimiento preventivo o correctivo al hardware o software y mover las aplicaciones a otra computadora del cluster.

Estas actividades pueden ser actualización del sistema operativo, instalación de parches, cambio de puertos de red, cambio de tarjetas controladoras, actualizaciones de firmware o simplemente re-ínicializar la computadora. Las ventajas del cluster desarrollado sobre las soluciones actualmente disponibles son las siguientes: monitorea el estado de puertos de red, de HBA y procesos de la aplicación y cuenta con una interfaz gráfica para visualizar y administrar el estado del cluster, nodos y aplicación de misión crítica.

II. Parte Técnica del Artículo

La metodología usada para el desarrollo del sistema consistió en dividirlo en cinco módulos: el administrador del cluster, el administrador de puertos de red, el administrador de puertos de HBA, el administrador de la aplicación y la interfaz gráfica. En la Figura 1 se muestra la arquitectura del cluster implantado. Está compuesto por dos servidores conectados a un arreglo de discos compartidos donde se encuentra instalada la aplicación de misión crítica. La programación para implantar el cluster se compone de un programa principal y cinco módulos. Cada módulo es un programa, invocado desde el programa principal, que se ejecuta en segundo plano. La programación se realizó en lenguaje C, excepto la interfaz gráfica, la cual se desarrolló en lenguaje C#. La

Año: 2018 Volúmen: 1
Número: 4 ISSN: 2448-623X

comunicación entre el programa principal y los módulos se lleva a cabo a través de sockets. Los módulos de administración del cluster y el de interfaz gráfica entre nodos intercambian información usando también sockets. El programa principal arranca en el proceso de boot de cada nodo.

A. El administrador del cluster

Los nodos del cluster son dos servidores Proliant DL180 Gen 9 en los cuales se instaló el sistema operativo Red Hat 7. Este módulo del cluster es el más extenso y el más importante ya que realiza varias tareas. Su funcionamiento se basa en la información establecida por el administrador en el archivo de configuración del cluster y en el archivo de configuración de la aplicación.

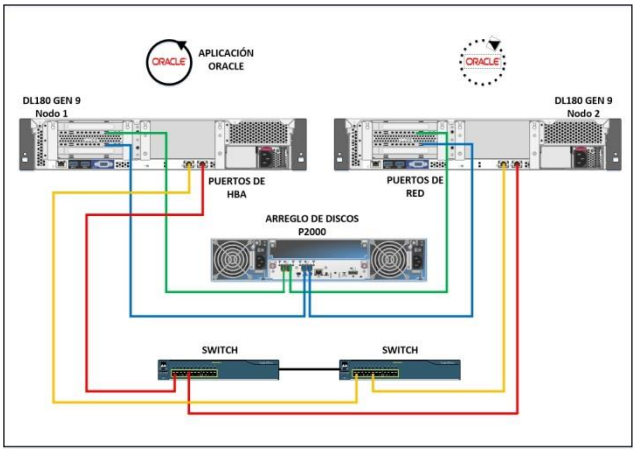

Figura 1. Arquitectura del cluster

El archivo de configuración del cluster contiene los siguientes parámetros: nombre del cluster, nombre de los nodos y valor del temporizador de la señal de latencia. El archivo de configuración de la aplicación contiene los siguientes parámetros: nombre, nodo primario y nodo alterno de la aplicación, ruta del script de arranque y paro, dirección IP y procesos de la aplicación a monitorear. El programa que implanta este módulo es el primero que se invoca desde el programa principal y realiza las siguientes tareas: trasmite periódicamente la señal de latencia entre los nodos del cluster, arranca el cluster, detiene el cluster, arranca un nodo, detiene un nodo y mantiene la comunicación con la interfaz gráfica. Este módulo almacena en una base de datos el estado el cluster, nodos y aplicación. La primera tarea que realiza el módulo es intentar arrancar el cluster con los dos nodos. Inicia transmitiendo mensajes de la señal de latencia al otro nodo y espera recibir respuesta en un tiempo no mayor al temporizador indicado en el archivo de configuración. Si la respuesta indica que el otro nodo ha formado el cluster, el nodo donde se ejecuta este proceso se integra al cluster. Si ambos nodos están intentando formar el cluster, éste se forma con los dos nodos. En cualquiera de los dos casos, el programa asigna al nodo y al cluster el estado "running", actualizando la base de datos y transmitiéndola al otro nodo para mantenerla sincronizada. En caso de no recibir respuesta de la señal de latencia trasmitida, el cluster no se forma de manera automática, el programa asigna al nodo y al cluster el estado "halted" y termina. El cluster podrá arrancarse manualmente solo con un nodo desde la interfaz de usuario, la cual invocará el programa de este módulo. Si el nodo se integró o formó el cluster, el programa de este módulo determina, a partir de la base de datos, si la aplicación se encuentra ejecutándose en el otro nodo. En caso negativo, obtiene, a partir del archivo de configuración de la aplicación, el nombre del nodo primario de la misma y solicita al módulo administrador de la aplicación el arranque de ésta en ese nodo. Después de que la aplicación arrancó, el programa de este módulo asigna a la aplicación el estado "running", actualizando la base de datos y transmitiéndola al otro nodo. Posteriormente, el programa entra a un ciclo donde transmite cada segundo mensajes de la señal de latencia al otro nodo. Si en algún momento expira el temporizador establecido en el archivo de configuración y no recibe repuesta de la señal de latencia, el programa verifica si en el otro nodo se encontraba ejecutándose la aplicación. En caso afirmativo, solicita al módulo administrador de la aplicación arrancarla en este nodo, asignando al otro nodo el estado "failed", a la aplicación el estado "running", actualizando la base de datos, deteniendo el envío de la señal de latencia y entrando a un ciclo donde espera un mensaje de uno de los módulos restantes. Desde la interfaz gráfica, se puede solicitar detener el cluster, el nodo o la aplicación. Si la opción es detener el nodo o el cluster, el programa verifica si en el nodo se encuentra ejecutándose la aplicación. En caso afirmativo, solicita al módulo administrador de la aplicación detenerla. A continuación, asigna a la aplicación y al nodo el estado "halted", actualizando la base de datos, transmitiéndola al otro nodo, deteniendo el envío de la señal de latencia y terminando el programa. En esta situación, la aplicación no arranca automáticamente en el otro nodo del cluster, solo lo hace cuando el nodo donde se encontraba ejecutándose falle, lo cual se conoce como "fail over" y es lo que proporciona alta disponibilidad en la aplicación. Cuando un nodo ha sido detenido y salido del cluster, para realizar en él tareas de mantenimiento, el administrador puede integrarlo, desde la interfaz gráfica, en el momento que esté listo para retornar al cluster. En este caso, la interfaz gráfica solicita al programa principal el arranque del programa que implanta este módulo del cluster para intentar integrar el nodo al cluster.

Año: 2018 Volúmen: 1
Número: 4 ISSN: 2448-623X

B. El administrador de puertos de red

Este módulo se implantó a través del segundo programa invocado desde el programa principal. Cada nodo del cluster cuenta con dos puertos de red Ethernet de 1 Gbps. Cada puerto se conectó a un switch de LAN diferente y los switchs se conectaron entre sí a través de otro puerto. De esta forma, se cuenta con alta disponibilidad en puertos de red y switchs. Los puertos de red en cada nodo se configuraron para formar un grupo o bonding en modo round robin, de manera que cuando los dos puertos están trabajando balancean el acceso a la red en cada servidor y cuando uno falla el acceso a la red se lleva a cabo por el que continúa trabajando, respaldándose mutuamente. El grupo de puertos de red tiene asignada la dirección IP del servidor al cual pertenece y la dirección IP de la aplicación en caso de que ésta se encuentre ejecutándose en el servidor. Los datos de usuarios, de la aplicación y paquetes de la señal de latencia se transmiten por este grupo de puertos de red. El programa que implanta este módulo se encuentra en un ciclo continuo monitoreando el estado del grupo y puertos de red, ejecutando el comando *if link show* cada segundo. Cuando un puerto de red falle, el programa transmite un mensaje al módulo administrador del cluster, solicitando asignar el estado "failed" al puerto que falló para que actualice la base de datos. Cuando fallen los dos puertos, transmite un mensaje al módulo administrador del cluster solicitando asignar el estado "failed" a los puertos que fallaron y solicita al módulo administrador de la aplicación que la detenga, en caso de encontrarse ejecutándose en ese nodo, y la arranque en el otro nodo, ya que los usuarios no podrán acceder la aplicación en el nodo donde han fallado los dos puertos de red. En el momento de recuperarse un puerto de red o el grupo de puertos después de una falla, El programa de este módulo transmite un mensaje al módulo administrador del cluster solicitando asignar el estado "running" al puerto que se recuperó.

C. El administrador de puertos de HBA

Este módulo es similar al anterior y se implantó por medio del tercer programa que se invoca desde el programa principal. Cada nodo del cluster cuenta con dos puertos SAS de 6 Gbps. A través de estos puertos los nodos acceden al arreglo de discos compartido donde se encuentra instalada la aplicación del cluster. Se usó un arreglo de discos HP Modular Smart Array P2000 G3 SAS, el cual cuenta con 12 bahías para discos de 2 TB. La capacidad máxima del arreglo es 26 TB. El arreglo de discos integra una tarjeta controladora con 4 puertos SAS. Cada puerto SAS de los nodos se conectó a un puerto del mismo tipo de la tarjeta controladora del arreglo de discos para implantar de esta forma alta disponibilidad en el acceso a datos y poder arrancar la aplicación del cluster en cualquiera de los nodos del cluster. Los puertos de HBA se configuraron para trabajar en modo round robin para balancear el acceso al arreglo de discos. Cuando un puerto falle, el acceso a los datos se lleva a cabo por el otro puerto, respaldándose ambos mutuamente. El programa que implanta este módulo se encuentra en un ciclo continuo monitoreando el estado de cada puerto de HBA a través de la ejecución del comando *multipath* cada segundo. Cuando un puerto falle, el programa transmite un mensaje al módulo administrador del cluster, solicitando asignar el estado "failed" al puerto HBA que falló para que actualice la base de datos. Cuando fallen los dos puertos, transmite un mensaje al módulo administrador del cluster solicitando asignar el estado "failed" a los puertos que fallaron y solicita al módulo administrador de la aplicación que la detenga y la arranque en el otro nodo, ya que el nodo donde fallaron los puertos no podrá acceder la aplicación. Al recuperarse un puerto de HBA o ambos después de una falla, el programa de este módulo transmite un mensaje al módulo administrador del cluster solicitando asignar el estado "running" al puerto HBA que se recuperó.

D. El administrador de la aplicación

Este módulo se implantó a través del cuarto programa invocado desde el programa principal. La aplicación que se ejecuta en el cluster es una base de datos creada con el manejador Oracle Linux 7. La base de datos y el software de Oracle se encuentran instalados en 4 discos del arreglo configurados en RAID 10 para proteger físicamente la información. El programa realiza tres funciones: arranque de la aplicación, paro de la aplicación y monitoreo de procesos de la aplicación. El arranque o paro de la aplicación puede solicitarse desde cada uno de los módulos restantes del cluster. Cuando la aplicación arranca, el programa de este módulo asigna la dirección IP de la misma al grupo de puertos de red donde ésta arranca y cuando la aplicación se detiene remueve la dirección IP de la misma del grupo de puertos de red. Esta es una dirección IP relocalizable que sirve para que los usuarios se conecten a la aplicación independientemente del nodo donde ésta se encuentre ejecutándose. El monitoreo de procesos tiene como objetivo determinar si la aplicación se encuentra ejecutándose. Para realizar esta tarea, el programa de este módulo accede periódicamente al despachador de procesos del sistema operativo para determinar el estado de los procesos de la aplicación. Los procesos a monitorear se establecen en el archivo de configuración de la aplicación. En caso de que un proceso no se encuentre ejecutándose, se envía solicitud al módulo administrador de la aplicación para que la detenga y la arranque en el otro nodo y un mensaje al módulo

Año: 2018 Volúmen: 1
Número: 4 ISSN: 2448-623X

administrador del cluster, solicitando actualizar la base de datos.

E. La interfaz gráfica

Desde esta interfaz el usuario administrador del cluster puede realizar las siguientes acciones: arrancar y detener el cluster, un nodo o la aplicación, así como visualizar el estado de los puertos de red y HBA. Cuando el estado de uno de los tres elementos anteriores del cluster sea "running" su ícono en la interfaz de usuario es verde y cuando el estado sea "halted" o "failed" es rojo. Esta interfaz cuenta con una opción para visualizar el archivo de texto donde se almacena la bitácora de los eventos registrados en el cluster. En la Figura 2 se muestra la ventana principal de la interfaz de usuario.

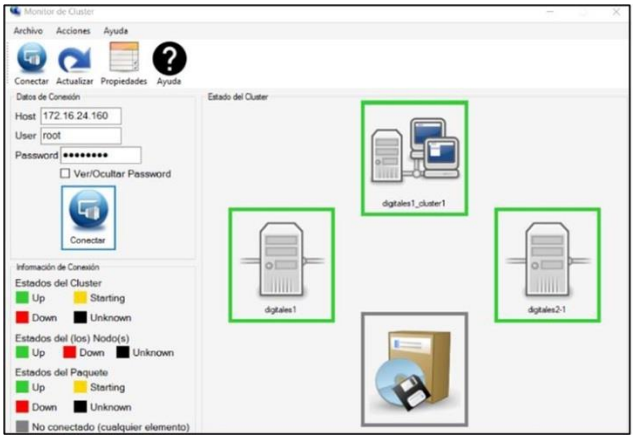

Figura 2. Interfaz de usuario del cluster

III. RESULTADOS

Se realizaron tres grupos de pruebas. El primer grupo tuvo como objetivo verificar la funcionalidad del cluster en caso de contingencia de uno de los nodos. Las pruebas consistieron en desconectar el cable de alimentación eléctrica de un nodo para simular la falla intempestiva del mismo. Al no recibir respuesta de la señal de latencia, el nodo que sobrevivió a la contingencia arrancó la base de datos Oracle correctamente, verificándose a continuación el acceso a la misma y su consistencia. Con esto, se aseguró que el cluster proporciona alta disponibilidad de la aplicación. A continuación, se conectó y arrancó el nodo en el que se simuló la falla y se verificó desde la interfaz de usuario que se integrara al cluster. Posteriormente, se repitió la misma prueba desconectando el cable de alimentación eléctrica del segundo nodo, con la aplicación ejecutándose en él. El resultado obtenido fue que la aplicación se movió al primer nodo del cluster. En estas pruebas la base de datos y el software de Oracle usaron 5 sistemas de archivos tipo VxFS, cada uno de 100 GB, y el tiempo que tardó la aplicación en arrancar en el nodo alterno al desconectar el cable de alimentación fue 5 minutos. Se realizó la misma prueba creciendo la base de datos cada vez en 5 sistemas de archivos de 100 GB hasta llegar a 50 sistemas de archivos. Se observó en cada prueba que aumentó el tiempo de arranque de la aplicación en el nodo alterno como se muestra en la gráfica de la Figura 3. Este resultado es de esperarse porque antes de arrancar la aplicación el sistema operativo verifica la consistencia de los sistemas de archivos antes de montarlos. El segundo grupo de pruebas tuvo como objetivo verificar la alta disponibilidad de puertos de red. Las pruebas de este grupo consistieron en abrir sesiones de usuario para acceder la base de datos y a continuación desconectar el cable de red de uno de los puertos. Se comprobó que los usuarios de la aplicación no perdían la conexión y podían accederla. El evento fue transparente para los usuarios, no así para el cluster quien lo registro en la bitácora. Posteriormente se desconectó el cable de red del puerto restante. El resultado fue como se esperaba, los usuarios perdieron la conexión a la aplicación y pudieron reconectarse a la misma a través de la dirección IP relocalizable de la aplicación después de que ésta arrancó en el otro nodo. El tercer grupo de pruebas tuvo como objetivo verificar la alta disponibilidad de puertos de HBA. Las pruebas y resultados obtenidos fueron similares a los del conjunto anterior de pruebas verificando la alta disponibilidad en este tipo de puertos.

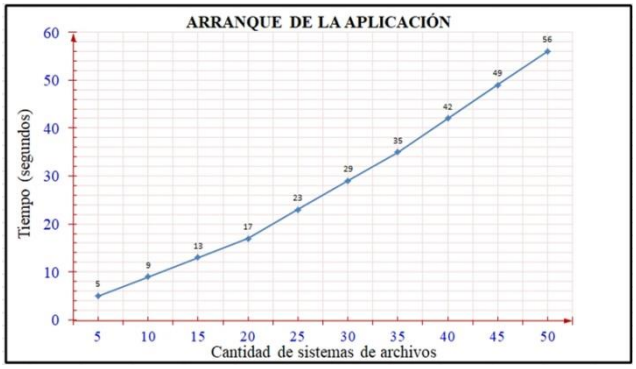

Figura 3. Tiempo de arranque de la aplicación

IV. DISCUSIÓN, CONCLUSIÓN Y RECOMENDACIONES

Se obtuvo un sistema altamente disponible compuesto por un cluster de dos servidores Linux y un arreglo de discos compartido. La falla de un puerto de red o un puerto HBA no interrumpe la ejecución de la aplicación, solo en caso de fallar los dos puertos de red o los dos puertos de HBA o todo el servidor la aplicación se moverá y arrancará en el segundo nodo del cluster. Cuando la aplicación se mueve los usuarios que estaban accediéndola pierden la conexión hasta que ésta arranque. Esto es el comportamiento normal de un cluster de alta disponibilidad, lo cual no sucede en sistemas tolerantes a fallas o non stop cuyo precio es

Año: 2018 Volúmen: 1
Número: 4 ISSN: 2448-623X

mucho más alto al de un cluster. La forma de administrar el cluster es a través de la interfaz de usuario. Está contemplado trabajar en una siguiente versión en la que el administrador pueda acceder al cluster por medio de línea de comandos. Se recomienda antes de instalar el programa del cluster, instalar y configurar el software de Oracle y crear la base de datos en el arreglo de discos compartidos.

V. AGRADECIMIENTOS

Los autores agradecen a la Universidad Autónoma Metropolitana Azcapotzalco por las facilidades otorgadas para la realización de este proyecto.

VI. REFERENCIAS

[1] Nemati, K., Zabalegui, A. y Bana, M. (2018). "Quantifying data center performance", *in Proceedings 34th Thermal Measurement, Modeling & Management Symposium (SEMI-THERM)*, pp. 141-147.

[2] Belskiy, J. y Belskaya, N. (2017). "Investigation basic characteristic operation of cloud data center", *in Proceedings 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*, pp. 245-248.

[3] Levy, M. y Raviv, D. (2017). "A framework for data center site risk metric", *in Proceedings IEEE 8th Annual Ubiquitous Computing*, Electronics and Mobile Communication Conference (UEMCON), pp. 9-15.

[4] Sasakura, K., Aoki. T. y Watanabe, T. (2017). "Study on Data Center Optimal Management by utilizing Data Center Infrastructure Management", *in Proceedings IEEE International Telecommunications Energy Conference (INTELEC)*, pp. 604-608.

[5] |AVAILABILITY ADD-ONS High Availability Data Sheet (2017). Red Hat, Inc., USA. Disponible en: https://www.redhat.com/en/files/resources/en-rhel-high-availability-add-ons-0000000.pdf.

[6] Managing HPE Serviceguard for Linux A.12.20.00 (2017). Hewlett Packard Enterprise, USA. Disponible en: http://h20628.www2.hp.com/km-ext/kmcsdirect/emr_na-a00039044en_us-1.pdf.

[7] Veritas™ Cluster Server Administrator's Guide Linux 6.0 (2017). Symantec Corporation, USA. Disponible en: https://sort.symantec.com/public/documents/sfha/6.0/linux/productguides/pdf/vcs_admin_60_lin.pdf.

[8] Hu, Y. y Zhu, S. (2014). "Load-balancing cluster based on Linux Virtual Server for internet-based laboratory", *in Proceedings 9th IEEE Conference on Industrial Electronics and Applications*, pp. 2181-2185.

[9] Zhang, Y., Ordonez, C. y Johnsson, L. (2017). "A Cloud System for Machine Learning Exploiting a Parallel Array DBMS", *in Proceedings 28th International Workshop on Database and Expert Systems Applications (DEXA)*, pp. 22-26.

[10] Zhao, D., Qiao, K. y Raicu, I. (2014). "HyCache+: Towards Scalable High-Performance Caching Middleware for Parallel File Systems", *in Proceedings 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 267-276.

[11] Duolikun, D., Watanabe, R. y Enokido, T. (2017). "An Eco Migration of Virtual Machines in a Server Cluster", *in Proceedings IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pp. 1098-1105.

[12] Kataoka, H., Sawada, A. y Duolikun, D. (2016). "Energy-Aware Algorithms to Select Servers in Scalable Clusters", *in Proceedings 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pp. 308-315.

[13] Playfair, D., Trehan, A. y McLarnon, B. (2016). "Big data availability: Selective partial checkpointing for in-memory database queries", *in Proceedings IEEE International Conference on Big Data (Big Data)*, pp. 2785-2794.

VII. BIOGRAFÍA

Vega-Luna José Ignacio. Estado de México, 1962. Ingeniería Electrónica, UAM-Azcapotzalco, Cd. de México, 1985. Maestría en Ciencias de la Computación, UAM-Azcapotzalco, Cd. de México, 1990.

Él labora actualmente en el área de Sistemas Digitales del Departamento de electrónica de la UAM-Azcapotzalco. Sus líneas de trabajo son: aplicaciones de microprocesadores y microcontroladores y sistemas operativos.

M. en C. Vega realiza investigación con redes inalámbricas de sensores y actuadores.

Cosme-Aceves José Francisco. Ingeniería Electrónica, UAM-Azcapotzalco, Cd. de México, 1985. Labora actualmente en el Departamento de Electrónica de la UAM-Azcapotzalco. Su línea de trabajo es lenguajes de descripción de hardware. El Ing. Cosme realiza investigación con sistemas embebidos y seguridad en redes de computadoras.

Salgado-Guzmán Gerardo. Cd. de México, 1968. Ingeniería Electrónica, UAM-Azcapotzalco, Cd. de México, 1992.

Él labora actualmente en el Departamento de Electrónica de la UAM-Azcapotzalco. Sus líneas de trabajo son: aplicaciones de microprocesadores y microcontroladores y sistemas operativos.

Ing. Salgado realiza investigación con redes inalámbricas de sensores y actuadores..

Sánchez-Rangel Francisco Rangel. Cd. de México, 1968. Ingeniería Electrónica, UAM-Azcapotzalco, Cd. de México, 1987. Maestría en Ciencias de la Computación, UAM-Azcapotzalco, Cd. de México, 1999.

Él labora actualmente en el Departamento de Electrónica de la UAM-Azcapotzalco. Sus líneas de trabajo son: aplicaciones de microprocesadores y microcontroladores y lenguajes de descripción de hardware.

M. en C. Sánchez realiza investigación con redes de computadoras y sistemas embebidos.

Contreras-Tovar David. Estado de México, 1986. Ingeniería en Computación, UAM-Azcapotzalco, Cd. de México, 2018.

Él labora actualmente en el Departamento de Electrónica de la UAM-Azcapotzalco. Sus líneas de trabajo son: sistemas operativos y redes de computadoras.

Ing. Contreras realiza investigación con sistemas de alta disponibilidad.