

Luis Caballero Cruz

Ingeniero Técnico Informático de Sistemas y Máster en Gestión de las Tecnologías de la Información y las Comunicaciones, Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla

<luiscc1989@gmail.com>

1. Introducción

En el sector de las Tecnologías de la Información y de las Comunicaciones (TIC de aquí en adelante) es necesario contar con redes y servicios de **alta disponibilidad**, ya que cada día están más ligados a los resultados comerciales que se obtienen y a la calidad con la que se ofrecen.

Las aspiraciones de las organizaciones hoy en día es intentar hacer realidad los cinco nuevos famosos de disponibilidad: 99,999%, que conlleva una inactividad de unos 5 minutos de *downtime* al año (ver **tabla 1**).

Cada minuto de inactividad, además de poder causar un impacto económico en las pérdidas, en la reputación de la empresa y en sus usuarios, supone una amenaza para la existencia del propio negocio, ya que estos tiempos de inactividad pueden significar una violación de una empresa en su nivel de acuerdo de servicio (SLA, *Service Level Agreement*) y conllevar una pérdida de clientes.

¿Cuáles son las causas que provocan estos tiempos de inactividad?

Podemos distinguir entre las causas **externas** y las causas **internas**. Las causas externas son imprevisibles como los graves desastres naturales producidos por incendios, terremotos o inundaciones. Sin embargo, las causas internas en las propias empresas suponen probablemente el mayor índice de causas de tiempos de inactividad en las organizaciones, como por ejemplo producidos por las propias personas, por caídas de la red, por errores de software o por errores de hardware. Un estudio realizado en 2010 por científicos investigadores de Microsoft Research [1] en un periodo de 14 meses en un centro de datos de gran escala con cientos de miles de servidores que se comunican entre sí para coordinar tareas con el fin de ofrecer alta disponibilidad, afirma que: "los errores por componentes hardware se magnifican y estos fallos pasan de ser una excepción a la norma convirtiéndose en un suceso común".

La mayor causa de reemplazos hardware son provocados por fallos en disco, en concreto el 78% de los tiempos de inactividad. Este dato es significativo y refleja que es fundamental la monitorización de los componentes, y en concreto de las bases de datos, debido a que son sistemas que realizan un trabajo intensi-

Monitorización de PostgreSQL: Plugin para Pandora FMS

Resumen: Los sistemas de monitorización son herramientas que nos permiten conocer en todo momento la situación en la que se encuentran las redes y servicios de alta disponibilidad. En concreto, con el potente y flexible sistema de monitorización Pandora FMS de Software Libre podemos conseguir una gestión segura sobre el control de diferentes componentes. En este artículo, destacamos los sistemas gestores de bases de datos que son esenciales para el almacenamiento y transacción de información que sucede cada día en Internet. Con Pandora FMS es posible realizar la supervisión de diferentes fabricantes como Oracle o MySQL, y ahora también PostgreSQL gracias a la contribución de un plugin realizado por el autor de este artículo dentro del periodo del proyecto final de carrera. Este plugin recopila datos estadísticos generales del servidor PostgreSQL y específicos de cada base de datos que almacena, así como del subproceso Background Writer. De esta forma conseguimos evitar interrupciones indeseadas y aumentar la disponibilidad.

Palabras clave: Alta disponibilidad, Pandora FMS, Perl, PostgreSQL, sistemas gestores de bases de datos, sistema de monitorización, Software Libre.

vo en disco por el número de transacciones que pueden realizarse en un entorno de producción.

Otro estudio realizado por la consultora Gartner en 2001 [2] afirma que es de vital importancia poseer algún mecanismo o herramienta que nos sirva de ayuda para el buen comportamiento del sistema y poder reaccionar ante estas adversidades. Teniendo en cuenta que la disponibilidad absoluta de los servicios es imposible debido a que conlleva una gran cantidad de recursos, es conveniente tener prevista una estrategia de respuesta mediante una gestión de la disponibilidad.

En concreto, aconseja el uso de medidas preventivas y la utilización de un **sistema de monitorización** como solución para la disponibilidad de los componentes, ya que la identificación de un fallo es el primer paso hacia su resolución.

2. Estado del arte

Ante la búsqueda de una solución en forma de sistema de monitorización para este problema real y actual en el sector de las TIC, realizamos un estudio y una comparativa entre nuestra solución seleccionada **Pandora FMS** [3] y algunos de los diferentes sistemas de monitorización más importantes y más utilizados, tanto de **software libre** como de software privativo.

2.1. Comparativa entre soluciones existentes

Para realizar una comparativa exhaustiva (ver **figura 1**) nos basamos en factores globales y competencias que las herramientas debían cumplir como las siguientes:

■ **Funcionalidad:** Debe ser capaz de monitorizar diferentes servicios, hardware y sistemas operativos. Realizar una recolección de datos y almacenarlo en bases de datos relacionales SQL a ser posible y posterior-

Porcentaje de disponibilidad	Tiempo de inactividad aproximado por año
95%	18 días
99%	4 días
99,9%	9 horas
99,99%	1 hora
99,999%	5 minutos

Tabla 1. Equivalencia de disponibilidad a tiempo de inactividad.

“ Teniendo en cuenta que la disponibilidad absoluta de los servicios es imposible debido a que conlleva una gran cantidad de recursos, es conveniente tener prevista una estrategia de respuesta mediante una gestión de la disponibilidad ”

mente, a partir de estos datos, generar gráficos e informes para su representación, así como enviar alarmas y notificaciones a los usuarios tras dispararse una alerta en algún nivel preestablecido.

■ **Fácil uso:** De cara al usuario, debe tener una consola web con control total sobre la aplicación tras realizar su instalación, evitando así tener que depender y recurrir constantemente de una interacción directa con la interfaz de línea de comandos (CLI). Además de poder realizar una personalización de dicha interfaz web y accesos mediante diferentes privilegios y roles, especialmente útil para las organizaciones. Tuvimos también en cuenta el tiempo necesario para instalar la herramienta, su configuración y su puesta a punto, y en relación a esto, la existencia de opciones que permitan realizar tareas en minutos que podrían conllevar días en realizarlas dentro de una gran red o infraestructura, como son las operaciones masivas en Pandora FMS.

■ **Arquitectura:** Debe permitir una monitorización híbrida [4], es decir, **con**

agentes en cada nodo a monitorizar y **sin agentes** de forma remota mediante protocolos de red, como por ejemplo SNMP (*Simple Network Management Protocol*). La herramienta debe ser multiplataforma en agente, permitiendo supervisar cualquier sistema operativo. Ha de ser una herramienta multiproceso [5] que divida el trabajo y optimice el rendimiento, permitiendo así a las distintas partes completar sus tareas sin obstaculizar al resto. La posibilidad de adaptarse a un entorno distribuido y soportar miles de agentes para un mismo servidor. Y una de las competencias más importantes a satisfacer, es que la herramienta debe permitir extensiones del sistema mediante **plugins** personalizables por los usuarios escritos en cualquier lenguaje de programación para la monitorización específica de diferentes herramientas.

■ **Calidad de soporte de la comunidad:** Intentamos medir cuánto de activo se encuentra el proyecto actualmente. Mediante las nuevas versiones y mejoras que se realizan en la herramienta en los últimos años corri-

giendo posibles *bugs*, la actividad de la comunidad en general atendiendo la resolución de problemas y peticiones de usuarios. También la posibilidad de disponer de un soporte profesional mediante una versión Enterprise de la herramienta en caso de requerir aspectos más específicos en entornos de producción donde inicialmente se confió en una herramienta de Software Libre o un soporte profesional en su defecto. Y por último, los idiomas disponibles de la documentación.

2.2. Solución seleccionada: Pandora FMS

Pandora FMS es una herramienta de Software Libre licencia GPL versión 2 (*GNU Public License*) y *GNU Lesser License v2* (LGPLv2). Permite supervisar e identificar cambios inesperados en nuestros sistemas y restablecer el nivel de servicio, atacando a los problemas desde su raíz.

Tal y como muestra la **figura 2** con datos obtenidos del sitio web ohloh.net [6], Pandora

Sistema de Monitorización	Software Libre	Funcionalidad	Fácil Uso	Arquitectura	Soporte
Nagios	✓	✓	✗	✗	✓
Hyperic HQ	✓	✓	✓	✓	✗
Zabbix	✓	✓	✓	✗	✓
Zennos	✓	✓	✗	✗	✓
Ganglia	✓	✗	✗	✗	✗
OpenNMS	✓	✓	✗	✗	✓
Cacti	✓	✗	✗	✗	✗
Munin	✓	✓	✗	✗	✗
BMC Patrol	✗	✓	✓	✓	✗
HP OpenView	✗	✓	✓	✓	✗
IBM Tivoli	✗	✓	✓	✓	✗
Pandora FMS	✓	✓	✓	✓	✓

Figura 1. Comparativa gráfica entre las diferentes soluciones de sistema de monitorización.

“ Como resultado de la comparativa obtenemos que Pandora FMS posee un mayor potencial frente a estas herramientas y una mayor estabilidad, siendo un proyecto de Software Libre que cumple con todas las competencias y factores globales mencionados anteriormente ”

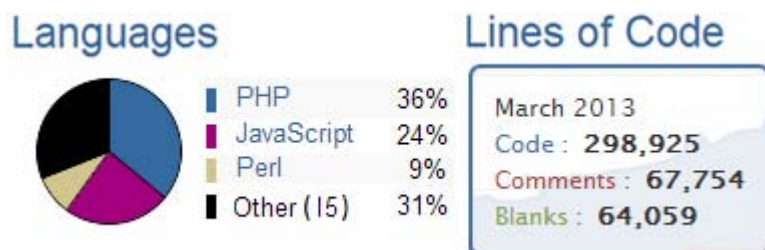


Figura 2. Resumen de tecnologías empleadas en Pandora FMS. Fuente: ohloh.net.

FMS cuenta actualmente con casi 300.000 líneas de código en diferentes lenguajes de programación, desde Perl y C++ para el servidor y agentes hasta PHP entre otros para la consola web.

Tiene más de 500.000 descargas, según estadísticas de SourceForge.net [7] (ver figura 3), y cuenta con el apoyo de una gran comunidad de más de 5.000 usuarios, de la cual formo parte gracias al proyecto final de carrera realizado [8].

Su comunidad ayuda mediante contribuciones al mantenimiento, desarrollo y progreso del proyecto en general, desde la traducción de documentación hasta la incorporación de nuevas extensiones para la herramienta.

Su empresa fundadora es Ártica Soluciones Tecnológicas con base en Madrid especializada en ámbitos de seguridad, la cual se encarga actualmente de dar soporte mediante la versión Enterprise de Pandora FMS a multitud de clientes por todo el mundo (ver figura 4).

Entre las herramientas con licencia de Software libre seleccionadas para comparar con Pandora FMS que pueden observarse en la figura 1, nos encontramos con Nagios core versión 3.x, un software popular y considerado pionero entre los sistemas de monitorización pero que presenta un código original con más de doce años de antigüedad con varias limitaciones en su arquitectura que ha quedado obsoleta.

Esto es debido a que no permite una monitorización híbrida al no disponer de agentes multiplataforma instalables en cada equipo a monitorizar y su dependencia con una herramienta auxiliar como NSClient++ para supervisar un equipo funcionando bajo el sistema operativo Microsoft Windows.

Además, es un sistema monotarea que no permite distribuir la carga de su ejecución ni autodescubrir nuevos equipos en la red. Esto conlleva como consecuencia que el sistema requiera un reinicio por completo del servicio en caso de realizar cualquier modificación en su configuración.

Por último en cuanto a su arquitectura, es un sistema que utiliza RRDtool (*Round Robin Database Tool*) y no soporta ningún SGBD que trabaje bajo SQL.

Por otra parte respecto a su facilidad de uso, carece de una consola web con control total sobre la aplicación y su interfaz web sólo sirve para visualizar los resultados, aunque existe la posibilidad de integrar Nagios con otra herramienta externa llamada Centreon para conseguir esta capacidad.

Su proceso de instalación junto con la configuración manual mediante la interacción directa con la interfaz de línea de comandos (CLI) necesita una importante inversión de tiempo y puede resultar en ocasiones tediosa y poco amigable.

Continuamos la comparativa con **Hyperic HQ** cuya principal desventaja ha sido localizada en su soporte. Debido a que posee una versión con licencia comercial propietaria de VMware Inc (filial de EMC Corporation), este software transmite una imagen excesivamente comercial de su producto. Debido a que la financiación del producto proviene de un fondo comercial, aparenta ser una mera organización que hace parte de su código *OpenSource*. Esto conlleva que el soporte de su comunidad de cara al usuario sea de menor calidad y menos activa en cuanto a nuevas actualizaciones y resolución de problemas. **OpenNMS** es otra herramienta de monitorización analizada con un sistema de notificaciones robusto, pero que presenta desventajas en su arquitectura y facilidad de uso. Debido a su desarrollo en el lenguaje de programación Java y la necesidad de su des-

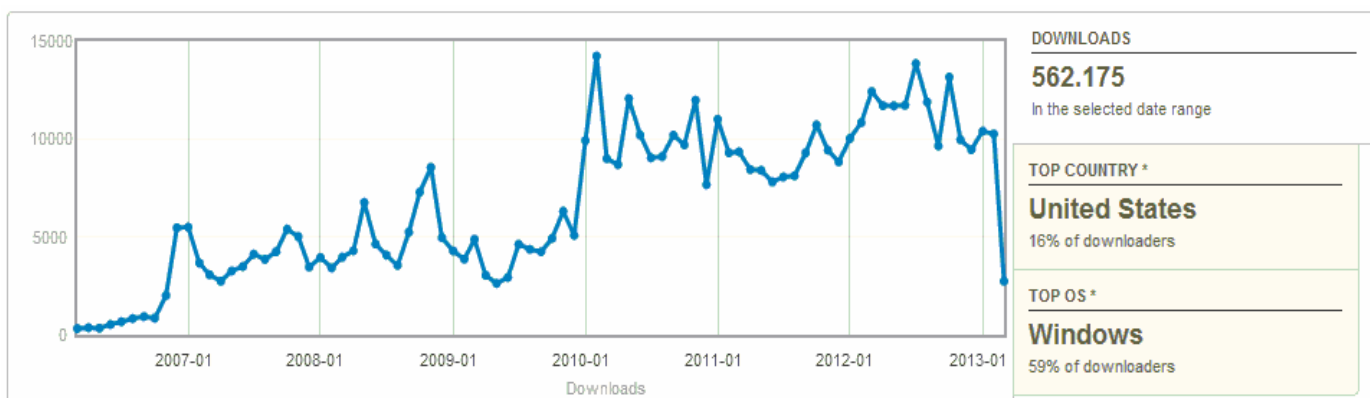


Figura 3. Estadísticas de descargas de Pandora FMS. Fuente: SourceForge.net.

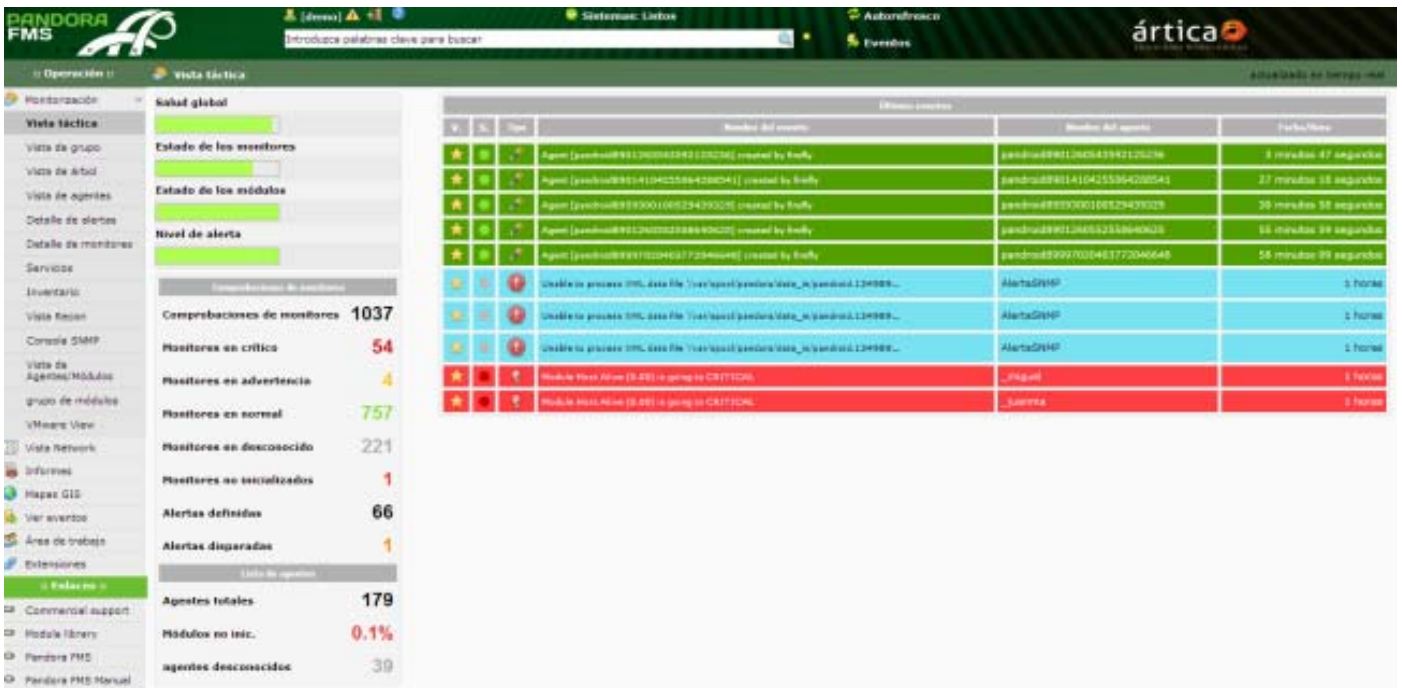


Figura 4. Consola web de Pandora FMS v4.0.2.

pliegue bajo un servidor de aplicaciones, existe la posibilidad de que sea una herramienta más pesada que Pandora FMS. Además, no dispone de agentes multiplataforma instalables en los equipos a monitorizar, ya que la recolección de datos puede realizarla de forma remota con protocolos estándar como SNMP y tecnologías Java como JMX (*Java Management Extensions*) y JDBC (*Java DataBase Collection*). Por último destacar que la evaluación de los datos mostrados y la predicción de estadísticas se hace a veces complicada.

Otra de las herramientas analizadas en la comparativa es **Zenoss** que posee una arquitectura que no soporta la utilización de agentes multiplataforma y por tanto sólo ofrece una monitorización remota con protocolos de red. También cabe destacar que a pesar de incorporar una serie extensiones del sistema llamadas ZenPacks, éstas son necesarias para monitorizar servicios básicos como HTTP, información específica del sistema operativo o recursos hardware. Esto afecta a su facilidad de uso y puesta a punto tras su instalación, la cual no es todo lo sencilla que debería, para cubrir funcionalidades básicas disponibles en Pandora FMS.

Otro de los sistemas que aparecen en la comparativa es **Zabbix** versión 1.8.8 cuya principal ventaja reside en la compatibilidad con cualquier SGBD SQL. Sin embargo, existen detalles más básicos respecto a Pandora FMS como su consola web, la visualización automática de mapas de red y conexiones, mapas GIS así como la creación de grupos lógicos. Además, hasta la versión 2.0 publicada en 2012, Zabbix no poseía una

arquitectura capaz de supervisar grandes entornos y presentaba limitaciones en su escalabilidad.

Esta desventaja también se refleja en sistemas de monitorización como **Munin** versión 1.4.6, **Cacti** versión 0.8.7 y **Ganglia** versión 2.1.8 junto a debilidades en sus arquitecturas: ausencia agentes multiplataforma, imposibilidad de autodescubrir la red e incompatibilidad con bases de datos SQL. Además se echa en falta una consola web con control total sobre la aplicación y un soporte profesional o una versión con licencia comercial. Los dos últimos sistemas mencionados, Cacti y Ganglia, además carecen de algunas funcionalidades básicas como un sistema para la generación de alarmas y eventos.

Estudiamos también soluciones de software privativo como son: **BMC Patrol**, **IBM Tivoli** y **HP OpenView**, las cuales cumplen mayoritariamente con las competencias técnicas requeridas. Sin embargo, la principal desventaja que encontramos es el alto coste de sus licencias, prohibitivas para la mayoría de las empresas.

Cabe añadir además la poca documentación técnica disponible de forma gratuita y la necesidad de conseguir más de una solución software que reúna todas las funcionalidades. Esto conlleva a la compra de más de una licencia, debido a que suelen tratarse de familias de productos y no proporcionan una única solución software para la monitorización de cualquier componente, al contrario de Pandora FMS que permite homogeneizar el sistema y utilizar una única herramienta para diferentes entornos.

Como resultado de la comparativa obtenemos que Pandora FMS posee un mayor potencial frente a estas herramientas y una mayor estabilidad, siendo un proyecto de Software Libre que cumple con todas las competencias y factores globales mencionados anteriormente.

Entre sus principales ventajas destacamos la **política** de la empresa Ártica ST, al desarrollar nuevos productos de software libre y realizarlo de forma global, con un código abierto y legible a todas las personas. De esta forma permite trabajar codo a codo con futuros clientes, ya que no sólo se les vende el producto, sino que permite que dichos clientes entren en el desarrollo del producto para sus propios intereses y avanzar hacia los objetivos juntos.

Este es el caso en España de Telefónica, que ha sustituido diferentes soluciones privativas como HP OpenView o IBM Tivoli, homogeneizando el sistema y consiguiendo un importante ahorro en gastos. Esta política también permite reducir los tiempos de evaluación de nuevas versiones y posibles errores, al contar con una gran comunidad de usuarios.

También cabe destacar la **flexibilidad** del sistema y su posible extensión mediante *plugins* específicos para diferentes sistemas en cualquier lenguaje de programación. A través de esta ventaja, iniciamos nuestra colaboración personal al proyecto Pandora FMS.

3. Mi contribución: Plugin para PostgreSQL en Pandora FMS

Entre la multitud de aplicaciones y sistemas que Pandora FMS es capaz de monitorizar de

“ Los *plugin* de agente permiten obtener varios módulos o piezas de información de una vez y por ello son mucho más flexibles que los *plugins* de servidor ”

forma específica encontramos sistemas operativos desde Windows a diferentes distribuciones Linux, comunicaciones de cualquier distribuidor como Cisco, y aplicaciones de virtualización o servidores web. En mi aportación a este proyecto me centré en los **sistemas gestores de bases de datos** (SGBD de aquí en adelante) y la posibilidad de una extensión específica hacia ellos. Los datos son uno de los bienes de negocio más críticos de las organizaciones y si no están disponibles y protegidos puede desembocar en diferentes desastres de negocio.

Cuando comencé mi proyecto final de carrera, existían *plugins* para diferentes fabricantes de SGBD como Oracle o MySQL. Así que la decisión fue desarrollar un *plugin* para monitorizar de forma específica el potente y robusto SGBD de Software Libre **PostgreSQL** [9].

Con este *plugin* cubrimos una monitorización para este tipo de bases de datos mediante la recolección de datos estadísticos, añadiendo una pieza necesaria para la administración de estos sistemas en alta disponibilidad. Este *plugin* se encuentra disponible para descargar en la librería de la web oficial de Pandora FMS junto a un manual de utilización [10].

3.1. Plugins en Pandora FMS

Existen dos tipos de *plugins* en Pandora FMS que cualquier usuario avanzado puede desarrollar con nuevas ideas y en cualquier lenguaje de programación. Son los llamados *plugins* de **Servidor** y de **Agente**, cuya clara diferencia es el lugar donde se realizará la carga que representa su ejecución y el número de diferentes piezas de información que recopila.

La ejecución de *plugin* de servidor es muy costosa, por lo que sólo es viable para *plugins* que no sean pesados y que no requieran de varias consultas para obtener una única pieza de información. Se recomienda que no requieran mucho procesamiento de datos y que sus intervalos de ejecución sean grandes, por ejemplo una vez a la semana.

Por otra parte, los *plugin* de agente permiten obtener varios módulos o piezas de información de una vez y por ello son mucho más flexibles que los *plugins* de servidor.

La idea es que el *plugin* recolecte la información, monte y envíe los datos en ficheros temporales en formato XML entendible al servidor de Pandora FMS, y tras su almacenamiento en la base de datos, los módulos puedan ser visualizados en la consola web. De esta forma se consigue distribuir la carga de

monitorización en distintas máquinas y no centralizarla en el servidor de Pandora FMS. Nuestro *plugin* desarrollado para PostgreSQL es del tipo Agente ya que recolecta varias piezas de información. El número de módulos mostrados es directamente proporcional al número de bases de datos en el servidor PostgreSQL a monitorizar.

3.2. Detalles y funcionamiento del plugin para PostgreSQL

El *plugin* fue escrito en el lenguaje de programación **Perl**, debido a que es un lenguaje de programación ligero, versátil, flexible y multiplataforma y permite crear complementos de forma rápida sin tener que aportar demasiadas bibliotecas y dependencias extras para su correcta ejecución. Además de que es un lenguaje muy utilizado en el proyecto de Pandora FMS para el desarrollo de nuevos *plugins*.

Es un *plugin* para la monitorización de estadísticas y rendimiento sobre un servidor de base de datos PostgreSQL. Con este *plugin* de agente, obtendremos datos estadísticos generales del servidor de PostgreSQL (ver **figura 5**), así como datos estadísticos específicos de cada base de datos que almacena dicho servidor (ver **figura 6**). También permite obtener información del subproceso

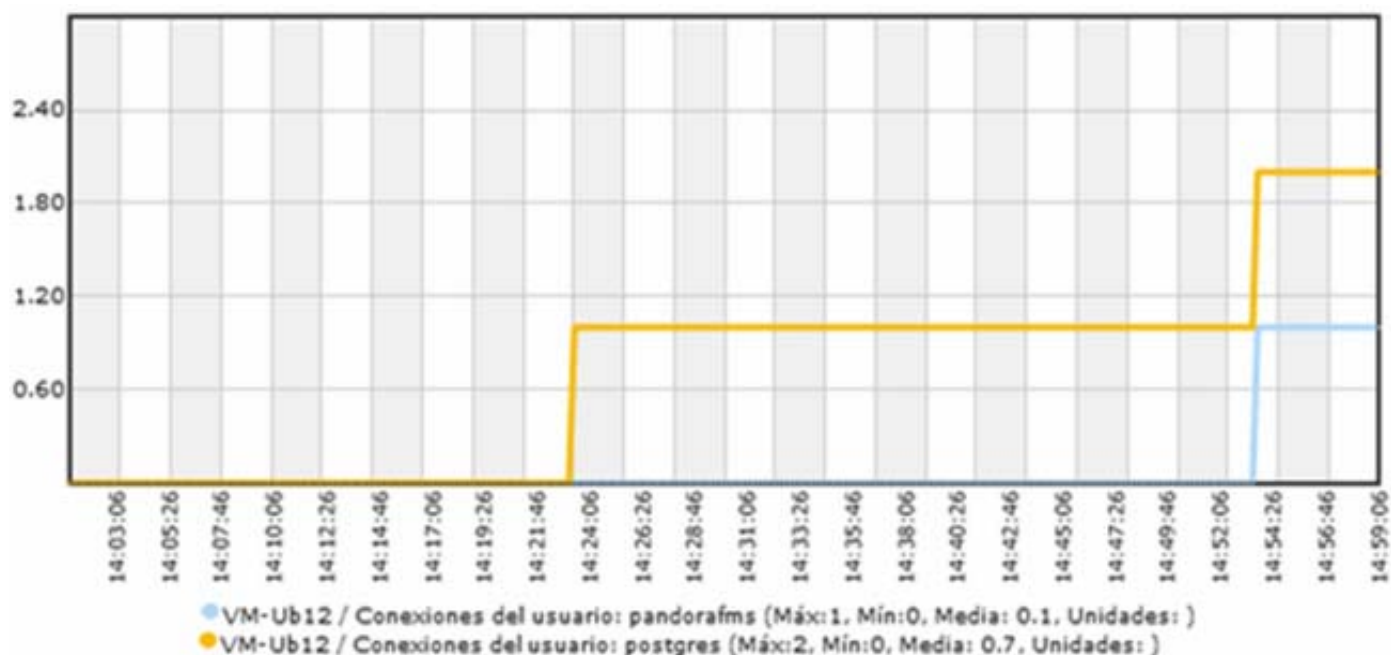


Figura 5. Número de usuarios conectados en un servidor PostgreSQL monitorizado en Pandora FMS.

“ Gracias a la flexibilidad de Pandora FMS y a su política hemos podido formar parte de su comunidad y contribuir a la extensión de la herramienta mediante nuestra pequeña aportación para este proyecto ”

Background Writer [11], existente para la optimización de servidores PostgreSQL.

Concretamente, este *plugin* recopila información acerca de:

- Estadísticas generales del servidor PostgreSQL como el número de conexiones de usuarios y el espacio en disco utilizado por cada base de datos que almacena.
- Estadísticas específicas de cada base de datos como el número de transacciones realizadas, tuplas insertadas o borradas, bloques de datos leídos tanto de caché como de disco y el número de procesos activos.
- Estadísticas acerca del subproceso de optimización para bases de datos PostgreSQL llamado BgWriter (*Background Writer*) como el número de puntos de comprobación realizados.

Todos los procesos en ejecución de un servidor PostgreSQL acceden a los datos del disco y los transfieren al buffer compartido o también llamado *Shared Buffer Pool*. Este buffer

utiliza algoritmos como LRU (*least recently used*) para seleccionar las páginas a desalojar.

En PostgreSQL encontramos el llamado proceso *Background Writer* cuya función es buscar y desalojar páginas modificadas o también llamadas páginas ‘sucias’ del buffer compartido y escribirlos en disco para conservar la coherencia en los datos. Mientras que un *checkpoint* podemos definirlo como un punto en el tiempo donde todas las páginas "sucias" se garantiza que han sido escritas en disco.

Por lo tanto, consideramos que es un método importante para identificar y medir cuánto de ocupado está un servidor de PostgreSQL y cómo de largas son las esperas al realizar nuevas transacciones. Por ello hemos optado por incluir diferentes valores acerca de este subproceso en nuestro *plugin* para Pandora FMS, ya que puede resultar útil para la

optimización de este tipo de bases de datos. Para obtener la información que posteriormente se almacena y se monitoriza en Pandora FMS, el *plugin* realiza **consultas SQL** contra el servidor de PostgreSQL que se quiere supervisar. Estas consultas recopilan la información de vistas protegidas en el sistema. Para que estas vistas sean accesibles y disponibles, es necesario realizar la activación del subproceso **The Statistics Collector** [12] o colector de estadísticas en el servidor de PostgreSQL.

Por lo tanto, previamente a la ejecución del *plugin* y a la obtención de los datos estadísticos acerca del servidor PostgreSQL, es necesario editar algunos de sus ficheros de **configuración**. Concretamente *postgres.conf* y *pg_hba.conf*. El primero de ellos es para configurar y activar el ya mencionado subproceso colector de estadísticas, y el segundo para permitir el acceso como cliente desde el *host* donde se ejecutará el *plugin* al servidor de PostgreSQL.

Database							
	Active Process in: redmine	Numero de procesos activos de servidor c...		N/A - N/A	0		6:46 minutos
	Blocks disks in: redmine	Numero total de bloques no acertados en ...		N/A - N/A	2,025		6:46 minutos
	Blocks fetched in: redmine	Numero total de bloques leídos en la BD		N/A - N/A	813,543		6:46 minutos
	Blocks hit in: redmine	Numero total de bloques acertados en cac...		N/A - N/A	811,518		6:46 minutos
	Buffers_alloc	Numero de buffers asignados globalmente		N/A - N/A	6,471		6:46 minutos
	Buffers_backend	Numero de buffers escritos por procesos ...		N/A - N/A	14		6:46 minutos
	Buffers_Checkpoints	Numero de buffers escritos por CheckPoi...		N/A - N/A	90		6:46 minutos
	Buffers_clean	Numero de buffers limpiados en la ejecuc...		N/A - N/A	0		6:46 minutos
	Buffers_requested_Checkpoints	Numero de veces que BgWriter comenzo Che...		N/A - N/A	2		6:46 minutos
	Buffers_timed_Checkpoints	Numero de veces que BgWriter comenzo Che...		N/A - N/A	973		6:46 minutos
	Commits in: redmine	Numero total de Commits realizados en la...		N/A - N/A	11,958		6:46 minutos
	Conexiones del usuario: postgr...	Numero de Conexiones del Usuario		N/A - N/A	1		6:46 minutos
	Deleted in: redmine	Numero total de tuplas eliminadas en la ...		N/A - N/A	0		6:46 minutos
	Espacio en disco utilizado por...	Espacio en Disco Utilizado por la BD en ...		N/A - N/A	6,095,672		6:46 minutos
	Espacio en disco utilizado por...	Espacio en Disco Utilizado por la BD en ...		N/A - N/A	6,087,480		6:46 minutos
	Fetched in: redmine	Numero total de tuplas buscadas en la BD		N/A - N/A	206,444		6:46 minutos
	Inserted in: redmine	Numero total de tuplas insertadas en la ...		N/A - N/A	64		6:46 minutos
	Returned in: redmine	Numero total de tuplas devueltas en la B...		N/A - N/A	14,505,597		6:46 minutos
	Rollbacks in: redmine	Numero total de Rollbacks realizados en ...		N/A - N/A	4		6:46 minutos
	Updated in: redmine	Numero total de tuplas actualizadas en l...		N/A - N/A	2		6:46 minutos

Figura 6. Monitorización con el *plugin* específico para PostgreSQL en Pandora FMS.

En PostgreSQL es necesario realizar esta configuración para permitir la monitorización por parte de cualquier herramienta externa a la base de datos acerca de parámetros internos de la misma. Puesto que la recopilación de estadísticas agrega alguna sobrecarga sobre el servidor PostgreSQL, el sistema no viene preconfigurado y se deja a elección del usuario realizar su activación.

El *plugin* específico desarrollado para PostgreSQL recibe en su **ejecución** como argumento de entrada los datos de conexión al servidor. Estos datos los utilizará el cliente de PostgreSQL para conectarse al servidor a supervisar: superusuario (-u) y contraseña (-c), dirección IPv4 o nombre de *host* donde se encuentra el servidor (-h), nombre de la base de datos (-d) y puerto (-p). También tiene una opción que sirve de ayuda explicando su forma de utilización (-a).

Las **dependencias** directas del *plugin* para su correcta ejecución son las siguientes:

■ **Intérprete de Perl** y sus dependencias. El intérprete de Perl se encargará de compilar y ejecutar el script correctamente y es la base fundamental para que funcione este lenguaje de programación en cualquier sistema. Dicho paquete es instalado como requisito de Pandora FMS.

■ **Bibliotecas** unificadas para el acceso a bases de datos. Las bibliotecas *libdb-pg-perl* y *libdbi-perl* (requisito de Pandora FMS) son utilizadas en el *plugin* para la conexión como cliente al servidor de PostgreSQL y transforman las filas y columnas de cualquier base de datos a estructuras propias de Perl. Como propios módulos de Perl pueden descargarse desde la biblioteca *online* disponible llamada CPAN [13].

■ **Cliente de PostgreSQL**. Este paquete es necesario para realizar la conexión como cliente al servidor de base de datos PostgreSQL. El *psql client* es un cliente de línea de comandos distribuido con PostgreSQL y a menudo se llama el monitor interactivo o terminal interactivo. Con *psql*, se obtiene una herramienta sencilla pero poderosa con la que se puede interactuar directamente con el servidor PostgreSQL, y así comenzar a explorar en lenguaje SQL.

■ **Agente de Pandora FMS**. Como para cualquier *plugin* de Agente en Pandora FMS, es necesaria su instalación y configuración en el *host* donde se ejecutará el *plugin* para enviar los datos al servidor de Pandora FMS.

4. Conclusiones

En este artículo destacamos los sistemas de monitorización como elementos críticos para la gestión de redes y servicios de alta disponibilidad. Tras el sondeo realizado y el estudio comparativo de diferentes sistemas de monitorización apoyamos al proyecto Pandora FMS como solución de Software Libre a este problema. Pandora FMS ayuda a la toma de decisiones, a la prevención de

tendencias y a la detección de fallos recurrentes mediante el histórico de datos recopilado. Con Pandora FMS obtenemos un aumento de la disponibilidad, una mejor relación coste-beneficio y un mayor potencial frente a otras herramientas existentes en el sector. Gracias a su flexibilidad y a su política hemos podido formar parte de su comunidad y contribuir a la extensión de la herramienta mediante nuestra pequeña aportación para este proyecto.

Dicha colaboración ha sido realizada mediante un *plugin* desarrollado en Perl y validado a través de pruebas unitarias para la monitorización específica del SGBD PostgreSQL. Con este *plugin* conseguimos la supervisión de su rendimiento y estado, así como la recopilación de datos para su optimización.

Agradecimientos

Me gustaría dedicar unas palabras de agradecimiento a aquellas personas que me han ayudado, tanto en la realización de mi proyecto final de carrera como en la realización de este artículo. Comenzando desde mi tutor Pablo Neira Ayuso y toda la comunidad de Pandora FMS, gracias a los cuales he podido desarrollar mi trabajo. Terminando con mi familia y amigos que me han aportado sus consejos y ánimo incondicional para llevarlo a cabo con ilusión y esfuerzo hasta el último momento.

Referencias

- [1] Kashi Venkatesh, Nachiappan Nagappan. Microsoft Research, *Characterizing Cloud Computing Hardware Reliability*. <<http://research.microsoft.com/pubs/120439/socc088-vishwanath.pdf>>, 2010.
- [2] D.Prior, A.MacNeela, I.Brown, J.Krischer, D.Scott, J.Green-Armytage. Gartner, *Enterprise Guide to Gartner's High-Availability System Model for SAP*. <<http://www.tarrani.net/mike/docs/HiAvailModel4SAP.pdf>>, 2001.
- [3] Sancho Lerena. Ártica Soluciones Tecnológicas, *Pandora FMS*. <http://en.wikipedia.org/wiki/Pandora_FMS>, 2012.
- [4] Chris Knowles. *The Truth about Agent vs. Agentless Monitoring A Short Guide to Choosing the Right Solution*. <<http://www.uptimesoftware.com/pdfs/TruthAboutAgentVsAgentLess.pdf>>, 2007.
- [5] Gerhard Lausser. Linux Magazine nº 63, *Monitorización con Shinkem*. <http://www.linux-magazine.es/issue/63/067-071_ShinkemLM63.pdf>, 2011.
- [6] Ohloh. *The Open Source network*. Sitio web con información estadística acerca de proyectos de Software Libre, <http://www.ohloh.net/>.
- [7] SourceForge. *Web-based source code repository*. Sitio web de colaboración para proyectos software, <<http://sourceforge.net/>>.
- [8] Luis Caballero Cruz. *Memoria Proyecto Final de Carrera: Sistema de Monitorización Pandora FMS* tutorado por Pablo Neira Ayuso para la Universidad de Sevilla. <<http://1984.lsi.us.es/pfe/trac/pfe-pandora/raw-attachment/wiki/WikiStart/Memoria%20PFC%20-%20Luis%20Caballero%20Cruz.pdf>>, 2012.
- [9] PostgreSQL., <<http://www.postgresql.org>>.
- [10] Luis Caballero Cruz. *Plugin de agente de Pandora FMS específico para PostgreSQL junto a manual de utilización*. <http://pandorafms.com/index.php?sec=pandora&sec2=repository&lng=es&action=view_PUI&id_PUI=312>, 2012.
- [11] PostgreSQL. *Background Writer en PostgreSQL v9.1*, <<http://www.postgresql.org/docs/9.1/interactive/runtime-config-resource.html#RUNTIME-CONFIG-RESOURCE-BACKGROUND-WRITER>>.
- [12] PostgreSQL. *The Statistics Collector en PostgreSQL v9.1*, <<http://www.postgresql.org/docs/9.1/interactive/monitoring-stats.html>>.
- [13] CPAN. Biblioteca online de módulos para Perl, <<http://www.cpan.org/>>.