# Estudios de volumetría y capacidad

# Estudios de volumetría y capacidad

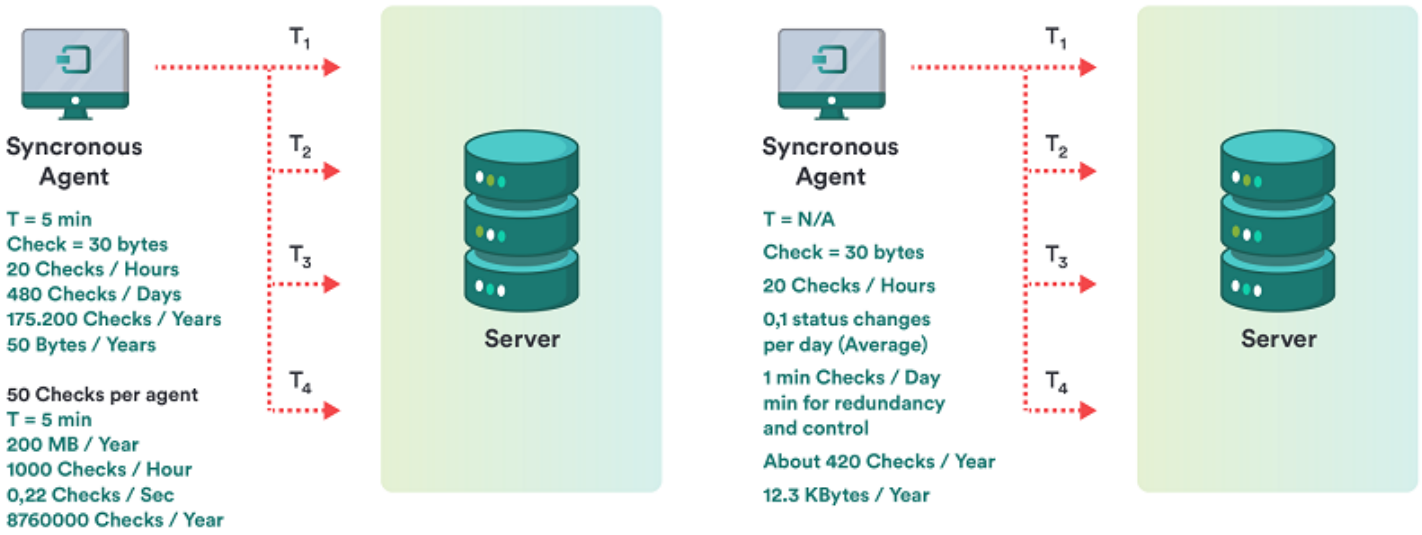
## Introducción

Pandora FMS es una aplicación distribuida compleja que tiene diferentes elementos clave, susceptibles de representar un cuello de botella si no se dimensiona y configura correctamente. El propósito de este capítulo es ayudar a realizar un estudio propio de capacidad, para analizar la *escalabilidad* de Pandora FMS en función de una serie específica de parámetros. Este estudio ayudará a conocer los requisitos que debería tener la instalación para poder soportar determinada capacidad.

Las pruebas de carga sirven también para observar la capacidad máxima por servidor. En el modelo de arquitectura actual (v3.0 o posterior), con "N" servidores independientes y una Metaconsola , esta *escalabilidad* tiende a ser de orden lineal, mientras que la *escalabilidad* basada en modelos centralizados es exponencial.

### Almacenamiento de datos y compactación

El hecho de que Pandora FMS compacte datos en tiempo real, es muy relevante de cara a calcular el tamaño que van a ocupar. Se realizó un estudio inicial que comparaba la forma de almacenar los datos de un sistema clásico con la forma "asíncrona" de almacenar los datos de Pandora FMS. Esto se puede ver en el esquema que hay incluido en este capítulo.




En un sistema convencional

Para un chequeo, con una media de 20 iteraciones al día, se tiene un total de 5 MB al año en espacio ocupado. Para 50 chequeos por agente, son 250 MB por año.

En un sistema no convencional, asíncrono o con compactación, como Pandora FMS

Para un chequeo, con una media de 0,1 variaciones al día, se tiene un total de 12,3 KB al año en espacio ocupado. Para 50 chequeos por agente, son 615 KB por año.

### Terminología específica

Se describe a continuación un glosario de términos *específicos* para este estudio, para mejor comprensión del lector.

- Fragmentación de la información: La información que maneja Pandora FMS puede tener diferente comportamiento, ya sea cambiar constantemente (por ejemplo un medidor de porcentaje de CPU) o ser estática (por ejemplo el estado de un servicio). Dado que Pandora FMS aprovecha esto para "*compactar*" la información en la base de datos (BD), es un factor crítico para el rendimiento y el estudio de capacidad, ya que a más fragmentación, más tamaño en la BD y más capacidad de proceso será necesario emplear para procesar la misma información.
- Módulo: Es la pieza básica de información recolectada para su monitorización. En algunos entornos se lo conoce como Evento, en otros como monitor y en otros como métrica.
- Intervalo: Es la cantidad de tiempo que transcurre entre recogidas de información de un módulo. Generalmente se expresa en minutos o segundos. El valor "medio" suele ser de 5 minutos.
- Alerta: Se conoce así a la notificación que ejecuta Pandora FMS cuando un dato se sale de unos márgenes establecidos o cambia de estado a `CRITICAL` (crítico) o `WARNING` (advertencia).

## Ejemplo de estudio de capacidad

### Definición del alcance

El estudio se ha hecho pensando en una implantación dividida en tres fases principales:

- Fase 1: Despliegue de 500 agentes.
- Fase 2: Despliegue de 3000 agentes.
- Fase 3: Despliegue de 6000 agentes.

Para determinar con exactitud los requisitos de Pandora FMS en implantaciones de este volumen de datos hay que conocer muy bien que tipo de monitorización se va a realizar, con la mayor exactitud posible. Para el siguiente estudio se ha tenido en cuenta de forma específica las características del entorno de un cliente ficticio llamado "QUASAR TECNOLOGIES" que se pueden resumir en los siguientes puntos:

- Monitorización basada al 90% en agentes software.
- Sistemas homogéneos con una serie de caracterizaciones agrupadas en tecnologías / políticas.
- Intervalos muy variables entre los diferentes módulos / eventos a monitorizar.
- Gran cantidad de información asíncrona (eventos, elementos en *logs*).
- Mucha información de estado de procesos con muy poca probabilidad de cambio.
- Poca información de rendimiento respecto del total.

Después de hacer un estudio exhaustivo de todas las tecnologías y determinar el alcance de la implementación (identificando los sistemas y sus perfiles de monitorización), se ha llegado a las siguientes conclusiones:

- Existe una media de 40 módulos / eventos por máquina.
- El intervalo medio de monitorización es de 1200 segundos (20 minutos).
- Teniendo módulos que reportan información cada 5 minutos y módulos que reportan información una vez a la semana.
- De todo el conjunto de módulos totales (240 000) se ha determinado que la probabilidad de cambio de cada evento para cada muestra es del 25 %.
- Se ha determinado que la tasa de alertas por módulo es del 1,3 (es decir 1,3 alertas por módulo/evento).
- Se estima (en este caso es una estimación basada en nuestra experiencia) que la probabilidad de disparo de una alerta es del 1%.

## Medida de capacidad

Una vez conocidos los requisitos básicos para la implantación en cada fase (tasa de módulos / segundo), número de alertas totales, módulos por dia y megabytes por mes, se va a realizar una prueba de esfuerzo (*stress*) real sobre un servidor relativamente similar a los sistemas de producción (no se ha podido hacer la prueba en una máquina similar a las de producción).

Estas pruebas de *stress*, dirán qué capacidad de proceso tiene Pandora FMS en un servidor y cuál es su nivel de degradación con el tiempo. Esto sirve para los siguientes objetivos:

1. Por medio de una extrapolación, saber si el volumen final del proyecto será asumible con el hardware provisto a tal efecto.
2. Saber cuales son los límites de almacenamiento en línea (*online*) y cuáles deben ser los puntos de corte a partir de los cuales se mueva la información a las bases de datos de histórico.
3. Conocer los márgenes de respuesta ante picos de proceso, derivados de problemas que puedan surgir (parada de servicio, paradas planificadas) donde se acumule la información pendiente de procesar.
4. Conocer el impacto en el rendimiento derivado de la diferente calidad (porcentaje de cambio) de la información de monitorización.
5. Conocer el impacto del proceso de alertas en grandes volúmenes.

Las pruebas realizadas, han sido sobre un servidor DELL PowerEdge T100® con procesador Intel Core Duo® a 2,4Ghz y con 2 GB de RAM. Este servidor, funcionando sobre un Ubuntu Server 8.04 y ha proporcionado la base de estudio para las pruebas en entornos de Alta Capacidad. Las pruebas se han realizado sobre configuraciones de agente relativamente similares a las del proyecto de QUASAR TECNOLOGIES. La intención de las pruebas no es replicar exactamente el mismo volumen de información que va a tener QUASAR TECNOLOGIES, ya que no se dispone del mismo hardware, sino replicar un entorno de alta capacidad, similar al de QUASAR TECNOLOGIES para evaluar el impacto en el rendimiento a lo largo del tiempo y determinar otros problemas (principalmente de usabilidad) derivados de manejar grandes volúmenes de datos.

| | | | | | | |
|---|---|---|---|---|---|---|
| | Agentes | 6005 | Intervalo | 3000 | Modulos por Agente | 30 |
| | Modulos Tot. | 180150 | Prob. Cambio % | 100 | Alertas por Agente | 20 |

| | Probabilidad cambio (100%) | | | Probabilidad cambio (25%) | | |
|---|---|---|---|---|---|---|
| | Tasa Paq/Sec | Tasa Mod/Sec | Tasa Procesasdos | Tasa Paq/Sec | Tasa Mod/Sec | Tasa Procesasdos |
| Dia 1 | 3 | 90 | 149,88% | 8 | 240 | 399,67% |
| Dia 2 | 2,85 | 85,5 | 142,38% | 7,5 | 225 | 374,69% |
| Dia 3 | 2,71 | 81,23 | 135,26% | 6,75 | 202,5 | 337,22% |
| Dia 4 | 2,57 | 77,16 | 128,50% | 6,08 | 182,25 | 303,50% |
| Dia 5 | 2,44 | 73,31 | 122,07% | 5,47 | 164,03 | 273,15% |
| Dia 6 | 2,32 | 69,64 | 115,97% | 4,92 | 147,62 | 245,83% |
| Dia 7 | 2,21 | 66,16 | 110,17% | 4,43 | 132,86 | 221,25% |
| Dia 8 | 2,1 | 62,85 | 104,66% | 3,99 | 119,57 | 199,12% |
| Dia 9 | 1,99 | 59,71 | 99,43% | 3,59 | 107,62 | 179,21% |
| Dia 10 | 1,89 | 56,72 | 94,46% | 3,23 | 96,86 | 161,29% |
| Dia 11 | 1,8 | 53,89 | 89,74% | 2,91 | 87,17 | 145,16% |
| Dia 12 | 1,71 | 51,19 | 85,25% | 2,62 | 78,45 | 130,65% |
| Dia 13 | 1,62 | 48,63 | 80,99% | 2,35 | 70,61 | 117,58% |
| Dia 14 | 1,54 | 46,2 | 76,94% | 2,12 | 63,55 | 105,82% |
| Dia 15 | 1,46 | 43,89 | 73,09% | 1,91 | 57,19 | 95,24% |
| Dia 16 | 1,39 | 41,7 | 69,44% | 1,72 | 51,47 | 85,72% |
| Dia 17 | 1,32 | 39,61 | 65,96% | 1,54 | 46,33 | 77,14% |
| Dia 18 | 1,25 | 37,63 | 62,67% | 1,39 | 41,69 | 69,43% |
| Dia 19 | 1,19 | 35,75 | 59,53% | 1,25 | 37,52 | 62,49% |
| Dia 20 | 1,13 | 33,96 | 56,56% | 1,13 | 33,77 | 56,24% |
| Dia 21 | 1,08 | 32,26 | 53,73% | 1,01 | 30,39 | 50,61% |

Los resultados obtenidos son muy positivos ya que el sistema, aunque muy sobrecargado, era capaz de procesar un volumen de información muy considerable (180 000 módulos, 6000 agentes y 120 000 alertas). Las conclusiones derivadas de este estudio son las siguientes:

1. Se debe mover la información de “tiempo real” a la base de datos de histórico en un plazo máximo de 15 días, siendo lo óptimo hacerlo para datos de más de una semana. Esto garantiza una operación más rápida.
2. El margen de maniobra en el caso óptimo es de casi de un 50 % de capacidad de proceso, más alto de lo esperado teniendo en cuenta este volumen de información.
3. La tasa de fragmentación de la información es clave para determinar el rendimiento y la capacidad necesaria para el entorno donde se necesite implantar el sistema.

## Metodología detallada

Si bien el punto anterior representaba un estudio “rápido” basado únicamente en módulos de tipo “dataserver”, en este capítulo se presenta una forma más completa de hacer un análisis de la capacidad de Pandora FMS.

Como punto de partida, en todos los casos se utilizará siempre la filosofía de “el peor de los casos” siempre que se pueda escoger. Se asume que, si no se puede escoger, será la filosofía “el caso habitual”. Nunca se estimará nada en el “mejor de los casos” ya que no es válida.

A continuación, se verá cómo calcular la capacidad del sistema, por tipo de monitorización o en base al origen de la información.

### Servidor de datos

En base a unos objetivos, calculados según el punto anterior, se supondrá que el objetivo estimado es ver cómo se comporta con una carga de 100 000 módulos, repartido entre un total de 3000 agentes, esto es, una media de 33 módulos por agente.

Se creará una tarea de `pandora_xmlstress` ejecutada mediante cron o *script*) manual, que contenga 33 módulos, repartidos con una configuración similar a esta:

- 1 módulos de tipo cadena.
- 17 módulos de tipo `generic_proc`.
- 15 módulos de tipo `generic_data`.

Se configurarán los umbrales de los 17 módulos de tipo `generic_proc` de esta manera:

```
module_begin
module_name Process Status X
module_type generic_proc
module_description Status of my super-important daemon / service / process
module_exec type=RANDOM;variation=1;min=0;max=100
module_end
```

En los 15 módulos de tipo `generic_data` se deben establecer umbrales. El procedimiento a seguir será el siguiente:

Se configurarán los umbrales de los 15 módulos de tipo `generic_data` de forma que genere datos de tipo:

```
module_exec type=SCATTER;prob=20;avg=10;min=0;''m''ax=100
```

Se configurarán los umbrales para esos 15 módulos, de forma que tengan este patrón:

```
0-50 normal
50-74 warning
75- critical
```

Se añadirán al fichero de configuración del `pandora_xml_stress` unos *tokens* nuevos para poder definir los umbrales desde la generación del XML. Atención: esto debido a que Pandora FMS solo “adopta” la definición de umbrales en la creación del módulo, pero no en la actualización con datos nuevos.

```
module_min_critical 75
module_min_warning 50
```

Ejecute el `pandora_xml_stress`.

Se debe dejar corriendo al menos 48 horas sin ningún tipo de interrupción y debe monitorizar (con un agente de Pandora FMS) los siguientes parámetros:

- Número de paquetes encolados:

```
find /var/spool/pandora/data_in | wc -l
```

- CPU de `pandora_server`:

```
ps aux | grep "/usr/bin/pandora_server" | grep -v grep | awk '{print $3}'
```

- Memoria total de `PFMS server`:

```
ps aux | grep "/usr/bin/pandora_server" | grep -v grep | awk '{print $4}'
```

- CPU de mysqld (revisar sintaxis de la ejecución, depende de la distribución usada de MySQL)

```
ps aux | grep "sbin/mysqld" | grep -v grep | awk '{print $3}'
```

- Tiempo medio de respuesta de la BBDD de Pandora FMS:

```
/usr/share/pandora_server/util/pandora_database_check.pl
/etc/pandora/pandora_server.conf
```

- Número de monitores en estado desconocido (`unknown`):

```
 echo "select SUM(unknown_count) FROM tagente;" | mysql -u pandora -p<password>
-D pandora | tail -1
```

(donde `<password>` es la contraseña del usuario `pandora`)

Las primeras ejecuciones tienen que servirnos para afinar el servidor y la configuración de MySQL.

Se utilizará el *script* `/usr/share/pandora_server/util/pandora_count.sh` para contar (cuando haya ficheros XML pendientes de procesar) la tasa de procesamiento de paquetes. El objetivo es lograr que se puedan "procesar" todos los paquetes generados (3000) en un intervalo inferior al 80% del tiempo límite (5 minutos). Esto implica que se tienen que procesar 3000 paquetes en 4 minutos, luego:

```
3000 / (4x60) = 12,5
```

Se tiene que lograr una tasa de procesamiento de 12,5 paquetes como mínimo para estar razonablemente seguros de que Pandora FMS puede procesar esa información.

Elementos a ajustar:

- Número de hilos.

- Número máximo de elementos en cola intermedia (`max_queue_files`).
- Por supuesto, todos los parámetros pertinentes de MySQL (muy importante).

Importancia de todo esto: una instalación de Pandora FMS con un servidor GNU/Linux instalado “por defecto” en una máquina potente, puede no pasar de 5 a 6 paquetes por segundo, en una máquina potente bien “optimizada” y “acondicionada” puede llegar perfectamente de 30 a 40 paquetes por segundo. Esto también depende mucho de la cantidad de módulos que haya en cada agente.

Se configura el sistema para que el *script* de mantenimiento de BBDD en `/usr/share/pandora_server/util/pandora_db.pl` se ejecute cada hora en vez de cada día:

```
mv /etc/cron.daily/pandora_db /etc/cron.hourly
```

Se deja funcionando el sistema, con el generador de paquetes un mínimo de 48 horas. Una vez pasado ese tiempo se evalúan los siguientes puntos:

1. ¿Está el sistema estable?, ¿Se ha caído?. Si hay problemas, mirar *logs* y gráficas de las métricas obtenidas (principalmente memoria).
2. Evaluar la tendencia de tiempo de la métrica “número de monitores en estado desconocido”. No tiene que haber tendencias ni picos importantes. Deberían ser la excepción. Si se suceden con una regularidad de una hora, es porque hay problemas con la concurrencia del proceso de gestión de BBDD.
3. Evaluar la métrica “Tiempo medio de respuesta de la BBDD de Pandora FMS”. No debería crecer con el tiempo sino mantenerse constante.
4. Evaluar la métrica “CPU de `pandora_server`”: debería tener frecuentes picos, pero con una tendencia constante, no creciente.
5. Evaluar la métrica “CPU del servidor MYSQL”, debería permanecer constante con frecuentes picos, pero con una tendencia constante, no creciente.

**Evaluación del impacto de alertas**

Si todo ha ido bien, ahora se debe evaluar el impacto del rendimiento de la ejecución de alertas. Aplique una alerta a cinco módulos específicos de cada agente (de tipo `generic_data`), para la condicion `CRITICAL`. Algo que sea relativamente ligero, como crear un evento o escribir a syslog (para evitar el impacto que pudiera tener algo con alta latencia como enviar un mensaje de correo electrónico).

Opcionalmente puede crear una alerta de correlación de eventos para generar una alerta para cualquier condición crítica de cualquier agente con uno de esos cinco módulos.

Deje el sistema 12 horas operando bajo esos criterios y evalúe el impacto, siguiendo el criterio

anterior.

#### Evaluación del purgado y traslado de datos

Suponiendo que la política de almacenamiento de datos fuera:

- Borrado de eventos de más de 72 horas.
- Mover datos a histórico de más de 7 días.

Debería dejar el sistema funcionando “solo” durante al menos 10 días para evaluar el rendimiento a largo plazo. Puede que viera un “pico” sustancial al cabo de 7 días debido al movimiento de datos a la BBDD de histórico. Esa degradación es importante tenerla en cuenta. Si no se puede disponer de tanto tiempo, se puede reproducir (con menos “realismo”) cambiando el intervalo de purgado a 2 días en eventos y 2 días para mover datos a histórico, para evaluar ese impacto.

### Servidor de ICMP (Enterprise)

Se trata específicamente del servidor de red ICMP. En caso de realizar las pruebas para el servidor de red versión Open, vea el punto correspondiente al servidor de red (genérico).

Suponga que ya tiene el servidor funcionando y configurado. Algunos parámetros clave para su funcionamiento:

```
block_size X
```

Define el número de *pings* que hará el sistema por cada ejecución. Si la mayoría de *pings* van a tardar lo mismo, puede subir el número a un número considerablemente alto, por ejemplo: de 50 a 70.

Si, por el contrario, el parque de módulos de *ping* es heterogéneo y están en redes muy diferentes, con tiempos de latencia muy diferentes, no interesa poner un número alto, pues la prueba tardará lo que tarde el más lento, así que puede utilizar un número relativamente bajo, como de 15 a 20.

```
icmp_threads X
```

Evidentemente, cuantos más hilos tenga, más chequeos podrá ejecutar. Si suma todos los hilos que ejecuta Pandora FMS no deberían llegar al rango de 30-40. No debería usar más de 10 hilos aquí, aunque depende mucho del tipo de hardware y la versión de GNU/Linux que esté usando.

Ahora, debe “crear” un número ficticio de módulos de tipo *ping* para probar. Se asume que va a probar un total de 3000 módulos de tipo *ping*. Para ello, lo mejor es tomar un sistema en la red que sea capaz de soportar todos los pings (un servidor GNU/Linux cualquiera puede con esa

tarea).

Utilizando el importador CSV de Pandora FMS (disponible en la versión Enteprise), cree un fichero con el siguiente formato:

```
(Nombre agente, IP,os_id,Interval,Group_id)
```

Puede usarse este *shellscript* para generar ese fichero (cambiando la dirección IP de destino y el ID de grupo)

```
A=3000
while [ $A -gt 0 ]
do
    echo "AGENT_$A,192.168.50.1,1,300,10"
    A=`expr $A - 1`
done
```

Lo principal es tener a Pandora FMS monitorizada, midiendo las métricas que del punto anterior: consumo de CPU (pandora y mysql), número de módulos en estado desconocido y otros monitores interesantes.

Importe el CSV para crear 3000 agentes, lo cual tardará unos minutos. Luego vaya al primer agente (AGENT_3000) y cree en él un módulo de tipo PING.

A continuación, vaya a la herramienta de operaciones masivas y copie ese módulo a los otros 2999 agentes restantes.

Pandora debería empezar a procesar esos módulos. Mida con las mismas métricas que el caso anterior y vea como evoluciona. El objetivo es dejar un sistema operable para el número de módulos de tipo ICMP requerido sin que ninguno de ellos llegue a estado desconocido.

### Servidor de SNMP (Enterprise)

Aquí se trata específicamente del servidor de red SNMP Enterprise. En caso de realizar las pruebas para el servidor de red Open, vea el punto correspondiente al servidor de red (genérico).

Suponga que ya tiene el servidor funcionando y configurado. Algunos parámetros clave para su funcionamiento:

```
block_size X
```

Define el número de peticiones SNMP que hará el sistema por cada ejecución. Hay que tener en cuenta que el servidor los agrupa por dirección IP de destino, de forma que ese bloque es orientativo. Conviene que no sea muy grande (30 a 40 como máximo). Cuando un elemento del bloque falla, un contador interno hace que lo reintente el servidor Enterprise, y si tras X intentos

no funciona, se lo pasará al servidor Open.

```
snmp_threads X
```

Evidentemente, cuantos más hilos tenga, más chequeos podrá ejecutar. Si suma todos los hilos que ejecuta Pandora FMS no deberían llegar al rango de 30 a 40. No debería usar más de 10 hilos aquí, aunque depende mucho del tipo de hardware y la versión de GNU/Linux que esté usando.

La forma más rápida de probar es mediante un dispositivo SNMP, aplicando a todos los interfaces, todos los módulos de monitorización "básicos" de serie. Esto se hace mediante la aplicación del SNMP Explorer (Agente → Modo de administración → SNMP Explorer). Identifique las interfaces, y aplique todas las métricas a cada interfaz. En un *switch* de 24 puertos, esto genera unos 650 módulos.

Si genera otro agente con otro nombre, pero la misma dirección IP tendrá otros 650 módulos. Otra opción puede ser copiar todos los módulos a una serie de agentes que tengan todos la misma dirección IP de forma que los módulos copiados funcionen "atacando" al mismo *switch*.

Otra opción es utilizar un emulador de SNMP, como el Jalasoft SNMP Device Simulator.

El objetivo de este punto es ser capaz de monitorizar de forma constante, un *pool* de módulos SNMP durante al menos 48 horas, monitorizando la infraestructura, para asegurarse de que el *ratio* de monitorización de módulos por segundo es constante, y no existen períodos de tiempo donde el servidor produzca módulos en estado desconocido. Esta situación se podría dar por:

- Escasez de recursos (memoria, CPU). Se podría ver una tendencia de estas métricas en incremento continuo, lo cual es una mala señal.
- Problemas puntuales: reinicio del servidor diario (para rotado de *logs*), ejecución del mantenimiento programado de base de datos, u otros *scripts* que se ejecuten en el servidor o el servidor de BBBDD.
- Problemas de red, derivados de procesos no relacionados (por ejemplo el respaldo de datos de un servidor en la red) que afecten a la velocidad y disponibilidad de la red en determinado momento.

### Servidor de Plugins, Red (Open) y HTTP

Aquí aplica el mismo concepto que arriba, pero de forma más simplificada. Habrá que controlar:

- Número de hilos.
- Tiempo de expiración de procesos (*timeouts*) para calcular la incidencia en el peor de los casos.
- Tiempo medio de chequeo.

Dimensionar con esos datos un conjunto de prueba, y verificar que la capacidad del servidor es constante a lo largo del tiempo.

### Recepción de traps

Aquí el supuesto es más sencillo: se parte de que el sistema no va a recibir *traps* de forma constante, sino que más bien se trata de evaluar la respuesta ante una avalancha de *traps*, de los cuales algunos generarán alertas.

Para ello simplemente hay que hacer un *script* que genere *traps* de forma controlada a gran velocidad:

```
#!/bin/bash
TARGET=192.168.1.1
while [ 1 ]
do
   snmptrap -v 1 -c public $TARGET .1.3.6.1.4.1.2789.2005 192.168.5.2 6 666
1233433 .1.3.6.1.4.1.2789.2005.1 s "$RANDOM"
done
```

Nota: detenerlo con la tecla CTRL+C a los pocos segundos, pues generará cientos de *traps* rápidamente.

Una vez montado el entorno hay que validar los siguientes supuestos:

1. Inyección de *traps* a una tasa constante (basta con introducir un comando `sleep 1` al *script* anterior dentro del bucle while, para generar 1 trap por segundo. Se deja el sistema operando 48 horas y se evalúa el impacto en el servidor.
2. Tormenta de *traps*. Evaluar el antes, el durante, y la recuperación ante una tormenta de *traps*.
3. Efectos del sistema sobre una tabla de *traps* muy grande ( mayor a 50 mil). Esto incluye el efecto de pasar el mantenimiento de la BBDD.

### Eventos

De forma similar a los SNMP, se evaluarán los eventos del sistema PFMS en dos supuestos:

1. Tasa normal de recepción de eventos. Esto ya se ha probado en el servidor de datos, pues en cada cambia de estado, se genera un evento.
2. Tormenta de generación de eventos. Para ello, forzaremos la generación de eventos via CLI. Utilizando el comando siguiente (con un grupo existente llamado “Pruebas”):

```
/usr/share/pandora_server/util/pandora_manage.pl \
 /etc/pandora/pandora_server.conf --create_event "Prueba de evento" system
Pruebas
```

Ese comando, usado en un bucle como el usado para generar *traps*, se puede usar para generar decenas de eventos por segundo. Se puede paralelizar en un *script* con varias instancias para provocar un número más alto de inserciones. Esto serviría para simular el comportamiento del sistema ante una tormenta de eventos. De esta forma se podría probar el sistema, antes, durante y después de una tormenta de eventos.

## Concurrencia de usuarios

Para ello se usará otro servidor independiente de Pandora FMS, utilizando la funcionalidad de monitorización WEB. En una sesión de usuario donde realizará las siguientes tareas en un orden específico y medirá lo que tardan en ser procesadas:

1. Inicio de sesión en la consola web.
2. Ver eventos.
3. Ir a la vista de grupo.
4. Ir a la vista de detalle de agente
5. Visualizar un informe (en HTML). Dicho informe debería contener un par de gráficas y un par de módulos con informes de tipo SUMA o MEDIA. El intervalo de cada elemento debería ser de una semana o cinco días.
6. Visualización de una gráfica combinada (24 horas).
7. Generación de informe en PDF (otro informe diferente).

Se realiza esta prueba con al menos tres usuarios diferentes. Se puede paralelizar esa tarea para ejecutarla cada minuto, de forma que si hay 5 tareas (cada uno con su usuario), estaría simulando la navegación de cinco usuarios simultáneos. Una vez establecido el entorno, tendrá en cuenta:

1. La velocidad media de cada módulo es relevante de cara a identificar “cuellos de botella” relacionados con otras actividades paralelas, como la ejecución de *script* de mantenimiento, etcétera.
2. Se medirá el impacto de CPU y memoria en el servidor para cada sesión concurrente.
3. Se medirá el impacto de cada sesión de usuario simulada respecto al tiempo medio del resto de sesiones. Es decir, se debería estimar cuantos segundos de demora añade cada sesión extra simultánea.

Volver al Índice de Documentación Pandora FMS