Études volumétriques et de capacité

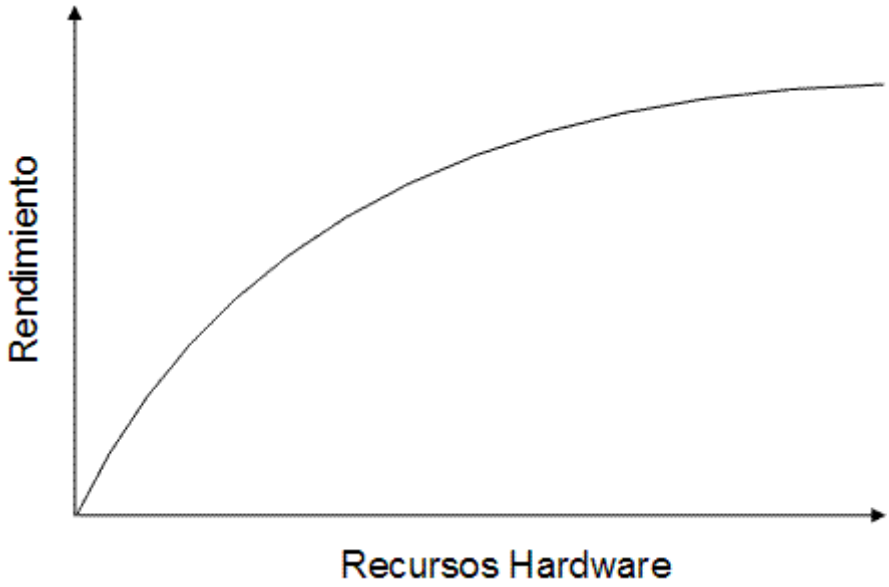
Études volumétriques et de capacité

Études volumétriques dans Pandora FMS

Introduction

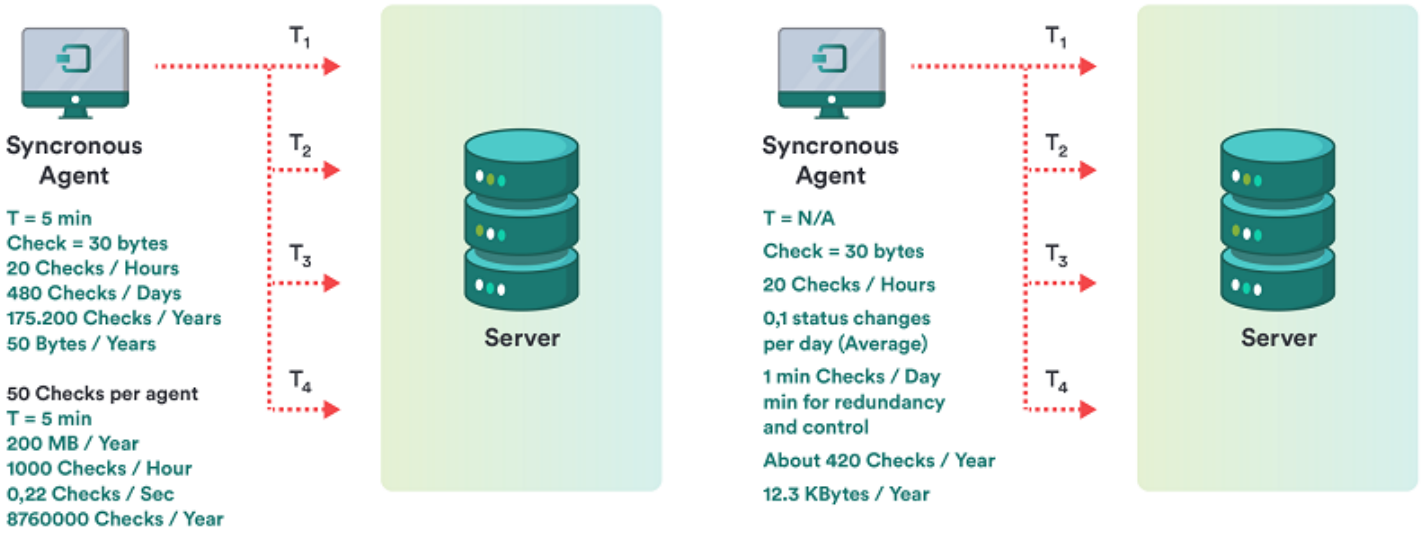
Pandora FMS est une application distribuée assez complexe qui comporte différents éléments clés, susceptibles de représenter un goulot d'étranglement si elle n'est pas dimensionnée et configurée correctement. Le but de ce chapitre est de vous aider à réaliser votre propre étude de capacité, afin d'analyser *l'évolutivité* de Pandora FMS en fonction d'une série spécifique de paramètres. Cette étude aidera à connaître les exigences que l'installation devrait avoir pour pouvoir supporter une certaine capacité.

Les tests de charge sont également utilisés pour observer la capacité maximale par serveur. Dans le modèle d'architecture actuel (v3.0 ou ultérieure), avec « N » serveurs indépendants et une Metaconsole , cette *évolutivité*tend à être linéaire dans l'ordre, tandis que *l'évolutivité* basée sur des modèles centralisés serait du type illustré dans le graphique suivant:



Stockage et compactage des données

Le fait que Pandora FMS compacte les données en temps réel est très pertinent pour calculer la taille qu'elles occuperont. Une première étude a été réalisée qui a comparé la façon de stocker les données d'un système classique avec la manière « asynchrone » de stocker les données Pandora FMS. Cela peut être vu dans le plan inclus dans ce chapitre.



Dans un système conventionnel

Pour un check-up, avec une moyenne de 20 itérations par jour, vous disposez d'un total de 5 Mo par an en espace occupé. Pour 50 contrôles par agent, cela représente 250 Mo par an.

Dans un système non conventionnel, asynchrone ou de compactage, tel que Pandora FMS

Pour un check-up, avec une moyenne de 0,1 itérations par jour, vous disposez d'un total de 12,3 Ko par an en espace occupé. Pour 50 contrôles par agent, cela représente 615 Ko par an.

Terminologie spécifique

Un glossaire de termes *spécifiques* pour cette étude est décrit ci-dessous, pour une meilleure compréhension du lecteur.

- Fragmentation des informations: Les informations traitées par Pandora FMS peuvent avoir un comportement différent, qu'il s'agisse de changer constamment (par exemple, un compteur de pourcentage de CPU) ou d'être statiques (par exemple, l'état d'un service). Étant donné que Pandora FMS en profite pour « *compacter* » les informations dans la base de données (BD), il s'agit d'un facteur critique pour la performance et l'étude de capacité, car plus la fragmentation est importante, plus la BD est grande et plus la capacité de traitement est importante.
- Module: C'est la pièce de base de l'information recueillie pour la supervision. Dans certains environnements, il est connu comme un événement, dans d'autres comme un moniteur et dans d'autres comme une métrique.
- Intervalle: C'est la quantité de temps qui s'écoule entre les collectes d'informations d'un module. Il est généralement exprimé en minutes ou en secondes. La valeur « moyenne » est généralement de 5 minutes.
- Alerte: C'est ce qu'on appelle la notification que Pandora FMS exécute lorsqu'une donnée sort des marges établies ou change d'état vers `CRITICAL` (critique) ou `WARNING` (avertissement).

Exemple d'étude de capacité

Définition du champ d'application

L'étude a été réalisée en pensant à une mise en œuvre divisée en trois phases principales:

- Phase 1: Déploiement de 500 agents.
- Phase 2: Déploiement de 3 000 agents.
- Phase 3: Déploiement de 6 000 agents.

Pour déterminer exactement les exigences de Pandora FMS dans la mise en œuvre de ce volume de données, il est nécessaire de savoir très bien quel type de supervision va être effectué, aussi précisément que possible. Pour l'étude suivante, les caractéristiques de l'environnement d'un client fictif appelé « QUASAR TECNOLOGIES » ont été spécifiquement prises en compte, ce qui peut être résumé dans les points suivants:

- Supervision basée à 90 % sur les agents logiciels.
- Systèmes homogènes avec une série de caractéristiques regroupées en technologies / politiques.
- Intervalles très variables entre les différents modules/événements à superviser.
- Grande quantité d'informations asynchrones (événements, éléments dans les journaux).
- Beaucoup d'informations sur l'état du processus avec très peu de probabilité de changement.
- Peu d'information sur le rendement par rapport au total.

Après avoir procédé à une étude exhaustive de toutes les technologies et déterminé la portée de mise en œuvre (identification des systèmes et de leurs profils de supervision), les conclusions suivantes ont été tirées:

- Il y a en moyenne 40 modules/événements par machine.
- L'intervalle de supervision moyen est de 1200 secondes (20 minutes).
- Ayant des modules qui rapportent des informations toutes les 5 minutes et des modules qui rapportent des informations une fois par semaine.
- Sur l'ensemble des modules totaux (240 000), il a été déterminé que la probabilité de changement de chaque événement pour chaque échantillon est de 25 %.
- Il a été déterminé que le taux d'alerte par module est de 1,3 (c'est à dire 1,3 alertes par module/événement).
- On estime (dans ce cas c'est une estimation basée sur notre expérience) que la probabilité de déclenchement d'une alerte est de 1 %.

Ces conclusions servent de base à la préparation de l'estimation, et sont codées dans la feuille de calcul pour faire l'étude:

Probabilidad cambio (%)	25	Probabilidad Disparo Alerta	1,00%
Intervalo medio (sec)	1200		
Módulos por agente	40		
Alertas / Modulo	130,00%		
Total Alertas	52		

Avec ces données de départ, et en appliquant les calculs nécessaires, vous pouvez estimer la taille de la base de données, le nombre de modules par seconde à traiter et d'autres paramètres essentiels:

Hito	Agentes	Total Módulos	Tasa Mod/Sec	Alertas	Módulos/Dia	MB/Mes	Modulos/Mes	Paq/Dia
Fase 1	500	20.000	16,67	26.000	360.000	1.030	10.800.000	144.000
	1.000	40.000	33,33	52.000	720.000	2.060	21.600.000	288.000
	2.000	80.000	66,67	104.000	1.440.000	4.120	43.200.000	576.000
Fase 2	3.000	120.000	100	156.000	2.160.000	6.180	64.800.000	864.000
	4.000	160.000	133,33	208.000	2.880.000	8.240	86.400.000	1.152.000
	5.000	200.000	166,67	260.000	3.600.000	10.300	108.000.000	1.440.000
Fase 3	6.000	240.000	200	312.000	4.320.000	12.360	129.600.000	1.728.000
	10000	400.000	333,33	520.000	7.200.000	20.599	216.000.000	2.880.000

Mesure de la capacité

Une fois connues les exigences de base pour l'implantation à chaque phase (taux de modules/seconde), nombre total d'alertes, modules par jour et mégaoctets par mois, un test d'effort (*stress*) réel va être effectué sur un serveur relativement similaire aux systèmes de production (le test n'a pas pu être effectué sur une machine similaire à celles de production).

Ces tests de *stress* indiqueront la capacité de traitement de Pandora FMS sur un serveur et son niveau de dégradation au fil du temps. Cela sert les objectifs suivants:

1. Au moyen d'une extrapolation, savoir si le volume final du projet sera assumable avec le matériel prévu à cet effet.
2. Savoir quelles sont les limites de stockage en ligne (*online*) et quels doivent être les points de coupure à partir desquels l'information est déplacée vers les bases de données historiques.
3. Connaître les marges de réponse aux pics de processus, découlant de problèmes pouvant survenir (arrêt de service, arrêts planifiés) où les informations en attente de traitement sont accumulées.
4. Connaître l'impact sur les performances dérivées de la qualité différente (pourcentage de changement) des informations de supervision.
5. Connaître l'impact du processus d'alerte sur de grands volumes.

Les tests ont été effectués sur un serveur DELL PowerEdge T100® équipé d'un processeur Intel Core Duo® à 2,4 GHz et de 2 Go de RAM. Ce serveur, fonctionnant sur un Ubuntu Server 8.04, a fourni la base de l'étude pour les tests dans les environnements à grande capacité. Les tests ont été effectués sur des configurations d'agent relativement similaires à celles du projet QUASAR TECHNOLOGIES. L'intention des tests n'est pas de reproduire exactement le même volume d'informations que QUASAR TECHNOLOGIES, car le même matériel n'est pas disponible, mais de reproduire un environnement de haute capacité, similaire à celui de QUASAR TECHNOLOGIES pour évaluer l'impact sur les performances au fil du temps et déterminer d'autres problèmes (principalement d'utilisabilité) découlant de la gestion de grands volumes de données.

	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%

Les résultats obtenus sont très positifs car le système, bien que surchargé, était capable de traiter un volume d'informations très important (180 000 modules, 6 000 agents et 120 000 alertes). Les conclusions de cette étude sont les suivantes:

1. Les informations « temps réel » doivent être transférées vers la base de données historique dans un délai maximum de 15 jours, ce qui est optimal pour les données de plus d'une semaine. Cela garantit une opération plus rapide.
2. La marge de manœuvre dans le cas optimal est de près de 50 % de capacité de traitement, soit plus que prévu compte tenu de ce volume d'informations.
3. Le taux de fragmentation de l'information est essentiel pour déterminer les performances et la capacité requises pour l'environnement où le système doit être implémenté.

Méthodologie détaillée

Alors que le point précédent représentait une étude « rapide » basée uniquement sur des modules de type « serveur de données », ce chapitre présente un moyen plus complet d'effectuer une analyse de la capacité de Pandora FMS.

Comme point de départ, dans tous les cas, on utilisera toujours la philosophie du « pire des cas » pour autant que l'on puisse choisir. On suppose que, si on ne peut pas choisir, ce sera la philosophie « le cas habituel ». Rien ne sera jamais estimé dans le « meilleur des cas » car il n'est pas valide.

Ensuite, vous verrez comment calculer la capacité du système, par type de supervision ou en

fonction de la source de l'information.

Serveur de données

Sur la base d'objectifs, calculés sur la base du point précédent, on supposera que l'objectif estimé est de voir comment il se comporte avec une charge de 100 000 modules, répartis entre un total de 3 000 agents, soit une moyenne de 33 modules par agent.

Une tâche `pandora_xmlstress` exécutée à l'aide d'un cron ou d'*script* manuel contenant 33 modules, avec une configuration similaire à celle-ci, sera créée:

- 1 module de type chaîne.
- 17 modules de type `generic_proc`.
- 15 modules de type `generic_data`.

Les seuils des 17 modules de type `generic_proc` seront configurés de la manière suivante:

```
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
```

Dans les 15 modules de type `generic_data`, des seuils doivent être définis. La procédure à suivre est la suivante:

Les seuils des 15 modules de type `generic_data` seront configurés de manière à générer des données de type:

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

Les seuils pour ces 15 modules seront configurés de manière à avoir ce modèle:

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

De nouveaux jetons seront ajoutés au fichier de configuration `pandora_xml_stress` afin de définir les seuils depuis la génération du XML. Attention: cela est dû au fait que Pandora FMS « adopte » uniquement la définition de seuils lors de la création du module, mais pas lors de la mise à jour avec de nouvelles données.

```
module_min_critical 75
module_min_warning 50
```

Exécutez le `pandora_xml_stress`.

Vous devez laisser passer au moins 48 heures sans interruption et superviser (avec un agent Pandora FMS) les paramètres suivants:

- Nombre de paquets en file d'attente:

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

- UCT de `pandora_server`:

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

- Mémoire totale du serveur PFMS:

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

- UCT de mysqld (examen de la syntaxe d'exécution, en fonction de la distribution MySQL utilisée)

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

- Temps de réponse moyen de la base de données de Pandora FMS:

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

- Nombre de moniteurs dans un état `inconnu`:

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

(où est le mot de passe de l'utilisateur `pandora`)

Les premières exécutions doivent nous servir à affiner le serveur et la configuration de MySQL.

Le *script* `/usr/share/pandora_server/util/pandora_count.sh` sera utilisé pour compter (lorsqu'il y a des fichiers XML en attente de traitement) le taux de traitement des paquets. L'objectif est de faire en sorte que tous les paquets générés (3 000) puissent être « traités » dans un intervalle inférieur à 80 % du temps limite (5 minutes). Cela implique que 3 000 paquets doivent être traités en 4 minutes, puis:

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

Un taux de traitement d'au moins 12,5 paquets doit être atteint pour être raisonnablement certain que Pandora FMS peut traiter ces informations.

Éléments à ajuster:

- Nombre de fils.
- Nombre maximal d'éléments en file d'attente intermédiaire (`max_queue_files`).
- Bien sûr, tous les paramètres pertinents de MySQL (très important).

Importance de tout cela: une installation Pandora FMS avec un serveur GNU/Linux installé « par défaut » sur une machine puissante peut ne pas dépasser 5 à 6 paquets par seconde, sur une machine puissante bien « optimisée » et « conditionnée » peut parfaitement atteindre 30 à 40 paquets par seconde. Cela dépend aussi beaucoup du nombre de modules dans chaque agent.

Le système est configuré pour que le *script* de maintenance de la base de donnée dans`/usr/share/pandora_server/util/pandora_db.pl` s'exécute toutes les heures au lieu de tous les jours:

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

Le système est laissé en marche, avec le générateur de paquets un minimum de 48 heures. Passé ce délai, les points suivants sont évalués:

1. Le système est-il stable ? Il est en panne ? S'il y a des problèmes, regardez les *journaux* et les graphiques des métriques obtenues (principalement la mémoire).
2. Évaluer la tendance temporelle de la métrique « nombre de moniteurs dans un état inconnu ». Il ne doit pas y avoir de tendances ni de pics importants. Ça devrait être l'exception. S'ils se succèdent avec une régularité d'une heure, c'est qu'il y a des problèmes avec la concurrence du processus de gestion de base de données.
3. Évaluer la métrique « Temps de réponse moyen du BBDD de Pandora FMS ». Il ne devrait pas grandir avec le temps, mais rester constant.
4. Évaluer la mesure « UCT `pandora_server` »: il devrait avoir des pics fréquents, mais avec une tendance constante, pas croissante.
5. Évaluer la métrique « UCT du serveur MYSQL » devrait rester constant avec des pics fréquents, mais avec une tendance constante et pas croissante.

Évaluation de l'impact des alertes

Si tout s'est bien passé, l'impact des performances d'exécution des alertes doit maintenant être évalué. Appliquez une alerte à cinq modules spécifiques à chaque agent (de type `generic_data`), pour la condition `CRITICAL`. Quelque chose de relativement léger, comme créer un événement ou écrire à syslog (pour éviter l'impact que pourrait avoir quelque chose avec une latence élevée comme l'envoi d'un message électronique).

Vous pouvez éventuellement créer une alerte de corrélation d'événement pour générer une alerte pour n'importe quel état critique de n'importe quel agent avec l'un de ces cinq modules.

Laissez le système fonctionner pendant 12 heures selon ces critères et évaluez l'impact en suivant le critère ci-dessus.

Évaluation du nettoyage et transfert de données

En supposant que la politique de stockage des données était:

- Suppression d'événements de plus de 72 heures.
- Déplacer les données vers l'historique de plus de 7 jours.

Vous devriez laisser le système fonctionner « seul » pendant au moins 10 jours pour évaluer les performances à long terme. Vous pourriez voir vu un « pic » substantiel au bout de 7 jours en raison du mouvement des données vers la base de données historique. Cette dégradation est important à prendre en compte. Si vous ne pouvez pas avoir autant de temps, il peut être reproduit (avec moins de « réalisme ») en changeant l'intervalle de purge à 2 jours dans les événements et 2 jours pour déplacer les données à historique, pour évaluer cet impact.

Serveur ICMP (Enterprise)

Il s'agit notamment du serveur réseau ICMP. Si vous effectuez des tests pour le serveur réseau version Open, reportez-vous au point correspondant au serveur réseau (générique).

Supposons que vous avez déjà le serveur en cours d'exécution et configuré. Quelques paramètres clés pour son fonctionnement:

```
block_size X
```

Il définit le nombre de *pings* que le système effectuera pour chaque exécution. Si la plupart des *pings* prennent le même temps, vous pouvez augmenter le nombre à un nombre considérablement élevé, par exemple de 50 à 70.

Si, au contraire, le parc de modules de *ping* est hétérogène et qu'ils sont sur des réseaux très différents, avec des temps de latence très différents, il n'est pas intéressant de mettre un nombre élevé, car le test prendra autant de temps que le plus lent, vous pouvez donc utiliser un nombre relativement faible, comme 15 à 20.

```
icmp_threads X
```

Évidemment, plus vous avez de fils, plus vous pouvez effectuer de contrôles. Si vous additionnez tous les fils exécutés par Pandora FMS, ils ne devraient pas atteindre la plage de 30-40. Vous ne devriez pas utiliser plus de 10 threads ici, même si cela dépend beaucoup du type de matériel et de la version de GNU/Linux que vous utilisez.

Maintenant, vous devez « créer » un nombre fictif de modules de type *ping* à tester. Il est supposé que vous allez tester un total de 3 000 modules du type *ping*. Pour ce faire, il est préférable de prendre un système sur le réseau qui est capable de supporter tous les pings (tout serveur

GNU/Linux peut faire cette tâche).

En utilisant l'importateur CSV de Pandora FMS (disponible dans la version Entreprise), créez un fichier au format suivant:

```
(Nom de l'agent, IP, os_id, Interval, Group_id)
```

Ce *shellscript* peut être utilisé pour générer ce fichier (en changeant l'adresse IP de destination et l'ID de groupe)

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

Tout d'abord, vous devez superviser Pandora FMS, en mesurant les mesures du point précédent: la consommation d'UCT (Pandora et mysql), le nombre de modules dans un état inconnu et d'autres moniteurs intéressants.

Importez le CSV pour créer 3 000 agents, ce qui prendra quelques minutes. Ensuite, allez au premier agent (AGENT_3000) et créez-y un module de type PING.

Ensuite, accédez à l'outil Opérations de masse et copiez ce module sur les 2 999 autres agents restants.

Pandora devrait commencer à traiter ces modules. Mesurez avec les mêmes mesures que le cas précédent et voyez comment il évolue. L'objectif est de laisser un système exploitable pour le nombre de modules de type ICMP requis sans qu'aucun d'entre eux n'atteigne un état inconnu.

Serveur SNMP (Enterprise)

Il s'agit ici spécifiquement du serveur réseau SNMP Enterprise. Si vous effectuez des tests pour le serveur réseau version Open, reportez-vous au point correspondant au serveur réseau (générique).

Supposons que vous avez déjà le serveur en cours d'exécution et configuré. Quelques paramètres clés pour son fonctionnement:

```
block_size X
```

Il définit le nombre de requêtes SNMP que le système effectuera pour chaque exécution. Il convient de noter que le serveur les regroupe par adresse IP de destination, de sorte que ce bloc est indicatif. Il ne doit pas être trop grand (30 à 40 au maximum). Lorsqu'un élément du bloc

tombe en panne, un compteur interne le fait réessayer par le serveur Enterprise, et si après X tentatives il ne fonctionne pas, il le transmettra au serveur Open.

```
snmp_threads X
```

Évidemment, plus vous avez de fils, plus vous pouvez effectuer de contrôles. Si vous additionnez tous les fils exécutés par Pandora FMS, ils ne devraient pas atteindre la plage de 30-40. Vous ne devriez pas utiliser plus de 10 threads ici, même si cela dépend beaucoup du type de matériel et de la version de GNU/Linux que vous utilisez.

Le moyen le plus rapide de tester est à l'aide d'un dispositif SNMP, en appliquant à toutes les interfaces, tous les modules de supervision « de base » de série. Cela se fait via l'application SNMP Explorer (Agent → Mode d'administration → SNMP Explorer). Identifiez les interfaces et appliquez toutes les mesures à chaque interface. Sur un *switch* de 24 ports, cela génère environ 650 modules.

Si vous générez un autre agent avec un autre nom, mais la même adresse IP, il aura 650 autres modules. Une autre option peut être de copier tous les modules à une série d'agents qui ont tous la même adresse IP afin que les modules copiés fonctionnent en « attaquant » le même commutateur.

Une autre option consiste à utiliser un émulateur SNMP, tel que le Jalasoft SNMP Device Simulator.

L'objectif de ce point est d'être en mesure de superviser en permanence, un *pool* de modules SNMP pendant au moins 48 heures, en surveillant l'infrastructure, pour s'assurer que le *taux* de supervision des modules par seconde est constant, et il n'y a pas de périodes de temps où le serveur produit des modules dans un état inconnu. Cette situation pourrait être due à:

- Pénurie de ressources (mémoire, UCT). On pourrait voir une tendance de ces métriques en augmentation continue, ce qui est un mauvais signe.
- Problèmes ponctuels: Redémarrage du serveur quotidien (pour la rotation des *journaux*), exécution de la maintenance planifiée de la base de données, ou autres *scripts* exécutés sur le serveur ou le serveur de base de données.
- Problèmes du réseau, résultant de processus non liés (par exemple, la sauvegarde des données d'un serveur sur le réseau) qui affectent la vitesse et la disponibilité du réseau à un moment donné.

Serveur de Plugins, Réseau (Open) et HTTP

Ici, il applique le même concept que ci-dessus, mais de manière plus simplifiée. Il faudra contrôler:

- Nombre de fils.
- Temps d'expiration des processus (*timeouts*) pour calculer l'incidence dans le pire des cas.
- Temps moyen de vérification.

Dimensionner avec ces données un ensemble de tests, et vérifier que la capacité du serveur est constante au fil du temps.

Réception des traps

Ici, l'hypothèse est plus simple: on part du principe que le système ne recevra pas de *traps* de manière constante, mais qu'il s'agit plutôt d'évaluer la réponse à une avalanche de déroutements, dont certains généreront des alertes.

Pour ce faire, il suffit de faire un *script* qui génère des *traps* de manière contrôlée à grande vitesse:

```
#!/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
```

Remarque: Arrêtez-le avec la touche CTRL+C en quelques secondes, car il générera rapidement des centaines de *traps*.

Une fois l'environnement installé, les hypothèses suivantes doivent être validées:

1. Injection de *traps* à un taux constant (il suffit d'entrer une commande `sleep 1` au *script* précédent dans la boucle while, pour générer 1 traps par seconde. Le système fonctionne 48 heures et l'impact sur le serveur est évalué.
2. Tempête de *traps*. Évaluer l'avant, le pendant, et la récupération avant une tempête de *traps*.
3. Effets du système sur une très grande table de *traps* (supérieure à 50 000). Cela inclut l'effet de passer la maintenance de la base de données.

Événements

De la même manière que les SNMP, les événements du système PFMS seront évalués dans deux hypothèses:

1. Taux normal de réception des événements. Cela a déjà été prouvé sur le serveur de données, car à chaque changement d'état, un événement est généré.
2. Tempête de génération d'événements. Pour ce faire, forcez la génération d'événements via CLI. En utilisant la commande suivante (avec un groupe existant appelé « Tests »):

```
/usr/share/pandora_server/util/pandora_manage.pl \
 /etc/pandora/pandora_server.conf --create_event "Event test" system Tests
```

Cette commande, utilisée dans une boucle comme celle utilisée pour générer des *traps*, peut être utilisée pour générer des dizaines d'événements par seconde. Il peut être parallélisé dans un *script* avec plusieurs instances pour provoquer un plus grand nombre d'insertions. Cela servirait à simuler le comportement du système face à une tempête d'événements. De cette façon, vous

pouvez tester le système, avant, pendant et après une tempête d'événements.

Concurrance d'utilisateurs

Pour ce faire, un autre serveur indépendant de Pandora FMS sera utilisé, en utilisant la fonctionnalité de supervision WEB. Dans une session utilisateur où vous effectuerez les tâches suivantes dans un ordre spécifique et mesurerez le temps qu'il faut pour les réaliser:

1. Connexion à la console web.
2. Voir les événements.
3. Aller à la vue de groupe.
4. Aller à la vue de détail de l'agent
5. Visualiser un rapport (en HTML). Ce rapport devrait contenir quelques graphiques et quelques modules avec des rapports de type SOMME ou MOYENNE. L'intervalle de chaque élément devrait être d'une semaine ou cinq jours.
6. Affichage d'un graphique combiné (24 heures).
7. Génération d'un rapport PDF (autre rapport).

Ce test est effectué avec au moins trois utilisateurs différents. Vous pouvez paralléliser cette tâche pour l'exécuter toutes les minutes, de sorte que s'il y a 5 tâches (chacune avec son utilisateur), vous simulerez la navigation de cinq utilisateurs simultanés. Une fois l'environnement établi, il tiendra compte:

1. La vitesse moyenne de chaque module est pertinente pour identifier les « goulots d'étranglement » liés à d'autres activités parallèles, telles que l'exécution de *scripts* de maintenance, etc.
2. L'impact de l'UCT et de la mémoire sur le serveur sera mesuré pour chaque session simultanée.
3. L'impact de chaque session utilisateur simulée sera mesuré par rapport au temps moyen des autres sessions. En d'autres termes, vous devriez estimer combien de secondes de retard chaque session supplémentaire simultanée ajoute.

Retour à l'index de documentation Pandora FMS