La supervision à l'INPL

Notes concernant Nagios, Centreon, et les projets périphériques

---

Stage ASRALL 2010

Julien VAUBOURG <julien@vaubourg.com>

14 avril 2010

Table des matières

1 Préambule

Ce document a pour but de synthétiser les différentes observations faîtes sur le système de supervision de l'INPL Les idées évoquées ne sont que des idées de pistes pour améliorer le mécanisme global et sont soumises à débat.

2 Nagios

2.1 Préambule

Nagios est **la référence libre de la supervision**. A ce titre, de **nombreuses sondes** sont mises à disposition par la communauté. **Sa qualité ne fait plus douter**, et avec Centreon elle s'est dotée d'un puissant allié. On peut noter que sa popularité entraîne **la profusion de logiciels compatibles** qui apportent encore de nouvelles possibilités.

Enfin, afin de **gérer au mieux l'existant**, il faudrait une raison sans parade pour décider de renier tout le travail déjà effectué. Cette raison n'a pas été mise sous la lumière, ainsi **Nagios n'a aucune raison d'être remplacé**.

Avant de complexifier la supervision déjà en place et avant de mettre en place de nouveaux services générateurs de notifications, il m'a semblé important de **faire un point sur les problèmes qui peuvent déjà être notés de ce qui est en place**. Parmi ces problèmes, celui de la gestion des notifications et leur pertinence vis-à-vis des administrateurs, est un problème qui sort de l'aspect technique mais qui représente pourtant la finalité concrète de la supervision.

2.2 Problèmes constatés

2.2.1 Nom des machines

Hétérogénéité au niveau de la désignation des machines des hôtes : adresse IP ou **nom de domaine**. Les hôtes utilisant des noms de domaine (ex : chenas) pour pointer les machines sont dépendants du service DNS. Si celui-ci ne répond plus, Nagios considérera que l'hôte ne répond plus, et le mettra en cause injustement.

2.2.2 Service `ping`

La présence de tous les hôtes est vérifiée par un test de `ping`. Sur la plupart des serveurs, un service `ping` est également monitoré. **La vérification est donc double.** En considérant le nombre de machines concernées, ce nombre de requêtes inutiles est non-négligeable.

2.2.3 Hôtes non-surveillés

Certains serveurs (ex : athenas) proposent des services qui sont monitorés en temps que tel, mais **ne sont pas elles-mêmes monitorées en temps que *hosts***. On peut remarquer que si aucun service ne répond, c'est probablement que c'est la machine qui ne répond plus, et s'en contenter. Mais puisque Nagios ne connait pas l'hôte auquel relier ces services (et ne sait donc pas qu'il est en carafe), il enverra une alerte pour chacun de ces services plutôt que de comprendre qu'il faut juste envoyer une alerte pour l'hôte (les services sous-jacents sont donc considérés `UNREACHABLE`, ce qui ne doit pas donner lieu à une levée de notifications).

2.2.4 Limites `PROCS` et `LOAD`

Des alertes sur ces services sont levées plusieurs fois par jour pour les mêmes machines. Ils reviennent rapidement en `OK`, sans aucune intervention. Soit la limite est trop basse, soit il y a un emballement régulier des serveurs. Dans tous les cas, **les `WARNING` entraînés polluent les alertes**.

2.2.5 Notifications `UNKNOWN`

Un statut `UNKNOWN` est souvent lié à une autre ressource qui n'est pas correcte. Cette ressource est probablement monitorée, et lèvera une alerte d'elle-même si elle ne se rétablit pas très vite. Ainsi, reporter **ce type de notification qui ne donne aucune information** est rarement utile et pollue les alertes.

2.2.6 Intitulés des messages

Améliorer la clarté des emails permettrait de **mieux les interpréter** (ex : `PROCS` désigne les processus mais pourrait désigner les processeurs), et de les rendre plus agréables.

2.2.7 Espace libre des disques durs

Beaucoup de `WARNING` sont levés sur ce type d'alerte, inutilement. La raison semble être que le même test avec les mêmes limites est effectué sur tous les types de machines supervisées. Or, un espace disque d'utilisateur plein n'a pas la même importance qu'un `/var` saturé.

2.3 Disponibilité de Nagios

Que se passe-t-il si Nagios lui-même, son service de mail, ou un commutateur réseau le reliant ne répond plus ? **Plus aucune alerte ne peut être émise**, ni pour lui, ni pour le reste du réseau.

Pistes :
- Haute disponibilité (cf. plus bas)
- Utilisation d'un réseau indépendant (**GSM**)

2.4 Du côté des administrateurs

Beaucoup d'alertes sont **filtrées automatiquement** par les administrateurs (et sont donc inutiles, puisque même en étant pertinente, c'est la considération de l'alerte par l'utilisateur qui lui donne son sens). Pour les autres, elles polluent les vrais alertes.

Pistes :
- Augmenter certaines limites
- Reconsidérer l'intérêt de la notification pour certains services (qui seront toujours visibles dans la console)
- Segmenter les utilisateurs pour qu'ils ne reçoivent que les alertes qui les concernent (séparation Windows - GNU/Linux ?)
- Ajouter de la finesse dans les périodes en prenant en compte les astreintes (problème de configuration dynamique en fonction des calendriers) et les opérations redondantes (ex : redémarrage hebdomadaire du LDAP)
- Moyens d'alertes alternatifs (**SMS**, écrans, afficheurs, etc.)
- Regroupement des équipements pour envoyer des alertes pour un lot de machines (*checks* de cluster - ex : bornes wifi)

3 Dimensionnement de la machine de supervision

3.1 Estimations

Part son fonctionnement même, Nagios consomme ses ressources à l'inverse de la plupart des applications : il consomme beaucoup de mémoire CPU pour peu de mémoire vive. Ceci est dû au lancement des sondes, sa principale activité, qui augmente le nombre de processus.

On considère qu'un serveur moyenne gamme peut superviser sans problème de **15 000 à 20 000 services**[1] . Actuellement, un millier de services sont supervisés (en ne prenant en compte qu'un Nagios). En cherchant sur un moteur de recherche populaire la définition d'un serveur moyenne gamme, hormis Sun[2] qui considère ce type de serveur comme un serveur à seize coeurs avec 256Go de mémoire, la moyenne se trouve autour d'un bi-coeur à deux gigas de mémoire. En relation avec le postulat évoqué ci-dessus, le serveur actuel est donc deux fois plus puissant qu'un tel serveur. On pourrait donc en conclure qu'il est capable de superviser **dans les 30 000 services**.

La performance d'un Nagios se teste par la valeur de sa latence. Celle-ci peut s'obtenir grâce à `nagiostats`. Elle est calculée à partir du temps moyen mis par un *check* pour s'exécuter, par rapport au moment où son exécution a été réclamée. Tant que la latence est inférieure à un, on considère que le serveur est bien proportionné (et que sa

[1] Réf. : Jean Gabès, *Nagios 3*, Eyrolles
[2] http://fr.sun.com/practice/systems/sparcenterprise/midrange.jsp

configuration est bonne).

Pour tester un serveur pour Nagios, il suffit de lancer un nombre d'occurrences croissant d'un *check* de `ping`. En observant l'évolution de la latence, on peut avoir une idée du nombre de services **simultanés** maximum supportés par la machine.

3.2 Améliorer les performances

3.2.1 Scripts contre programmes

Le test est fait avec un **script bash et un programme C** qui ne font que vérifier la présence ou non d'un fichier (avec `test` pour le premier et `fopen` pour le second) : pour 10 000 occurrences du test, la version bash met **35 secondes alors que la version C n'en met que huit**.

3.2.2 Perl, une exception

Nagios propose un module qui porte le nom de `ePN`. Il s'agit d'un **interpréteur Perl intégré**. Il a pour charge de compiler les scripts et de les mettre en cache sous forme de *bytecode*. Ainsi, les prochaines exécutions se feront sans l'étape de compilation - habituellement faîte à la volée et responsable des performances moindres des langages de script. Inconvénient toutefois, si un script Perl est modifié, **il faut relancer Nagios** pour faire regénérer le cache. Les scripts doivent être testés comme compatibles, avec une version indépendante de l'interpréteur, et il est possible d'activer/désactiver cette méthode de traitement des scripts Perl par Nagios à l'aide d'un simple commentaire.

L'interpréteur Perl de Nagios permettrait de **gagner 30% de performances**.

3.2.3 La virtualisation

D'après un test réalisé par Jean Gabès[3], un Nagios sur un système virtualisé perd **jusqu'à 40% de performances** par rapport à un hébergement sur un serveur physique.

3.2.4 LIT (*Large Installation Tweaks*) 1/3 : Les variables d'environnement

Pour obtenir des informations spécifiques, une sonde a deux moyens : les variables d'environnement et les arguments qui lui sont passées. **En désactivant ces premières** (paramètre Nagios) au profit de ces secondes, **le gain de performances serait de 15%**. Il faut toutefois parfaitement vérifier que les sondes sont adaptées à ce mode de fonctionnement.

3.2.5 LIT 2/3 : Nettoyage de la mémoire

Lorsqu'une sonde vient de faire son travail, Nagios s'occupe lui-même de nettoyer sa mémoire. Sur des systèmes récents, ce processus est quasiment inutile. En faisant confiance au système, et donc en désactivant ce mode, **Nagios gagnerait 90% de performances**.

3.2.6 LIT 3/3 : *Fork* de Nagios

Afin de ne pas mettre en carafe Nagios lorsqu'une sonde s'emballe, **celui-ci se *fork* pour lancer les *checks***. En considérant que les sondes sont parfaitement fiables et en faisant confiance au système pour détecter les processus zombies, **le gain de performances serait de 190%** ! En combinant les trois LIT, on obtient un gain de performances **aux alentours de 225%**.

3.2.7 Fichiers temporaires

Certains fichiers utilisés par Nagios sont volatiles et n'ont aucune importance en cas de redémarrage. Ainsi, **ce genre de fichier peut être écrit dans la mémoire vive** (`/dev/shm`) plutôt que sur le disque dur. Le gain de performances serait de **3% et les risques absolument nuls**.

[3] *Nagios 3*, Eyrolles

4 Centreon

4.1 Les modèles

Les modèles sont une notion que Nagios ne connait pas. Ils servent à gérer des services identiques pour plusieurs hôtes, mais qui différent légèrement d'un hôte à l'autre. Une solution dans Nagios aurait été les *macro-variables*. Centreon résout le problème d'une façon plus simple avec les modèles **en générant tous les services pour tous les hôtes**, puis en laissant la main sur chacun qui devient paramétrable et qui permet donc de rajouter les différences. Si c'est indéniablement plus rapide et pratique, il est intéressant de noter que que **Centreon n'est pas capable de supprimer tous ces services** générés lorsqu'on supprime le modèle.

4.2 Lancement des *checks*

Plus besoin d'aller se connecter sur un noeud pour forcer son *check*, Centreon propose de **les lancer manuellement** si besoin est depuis l'interface.

4.3 *Acknowledge*

Parmi les commandes passives de Nagios existe la commande `ACKNOWLEDGE_SVC_PROBLEM`, qui **permet de spécifier que le problème a été pris en compte**. Ceci a pour incidence que l'état reste visible dans la console, mais que les notifications ne sont plus envoyées.

4.4 Désactivation des notifications

En un clic, un service n'enverra **plus de notifications**.

4.5 Maintenances

Si une maintenance est programmée, Centreon propose de lui indiquer : **aucune des machines concernées, et donc des services associés, n'enverra de notification dans ce labs de temps**.

4.6 MySQL

L'interface traditionnelle de Nagios tire ses informations d'un fichier *status.dat*. Chaque consultation demande de parcourir l'intégralité du fichier. Il est courant que **celui-ci pèse près de dix méga-octets**. Centreon, grâce à NDO, utilise une base MySQL. Ainsi, les consultations sont **très fortement allégées** en terme de ressources demandées. Cerise sur le gâteau, Centreon dispose incidemment de toute **la puissance de SQL** pour sélectionner et filtrer, rapidement et efficacement.

4.7 Exports

Les données peuvent être exportées en **XML ou CVS**.

4.8 Métrologie

Point fort de Centreon : il embarque la métrologie (CentStorage). Il conserve ces données **sous forme de RRD, et dans la base MySQL**. Pour ces deux formats, il est possible de lui indiquer une **durée de rétention**.

4.9 SNMP

Second point fort de Centreon : la gestion du SNMP. L'interface propose de **compiler quasiment toutes les MIB constructeur existantes**. Il intègre tout ce qu'il faut pour faire des *checks* SNMP facilement, et **sans installation fastidieuse**.

4.10 Répartition des charges

Avec CentCore qu'il intègre, Centreon permet de facilité la répartition des charges. Ainsi, il sera possible de n'installer que Nagios sur des **serveurs satellites**, qui lui remonteront les informations qu'il synthétisera. **Centreon génère les fichiers de configuration** de tous les Nagios satellites.

4.11 NagVis

En utilisant MySQL avec NDO, **Centreon apporte quasiment tout ce qu'il faut à NagVis** pour fonctionner. Si celui-ci est amené à être utilisé, il ne reste quasiment plus rien à installer.

4.12 ACL

Un jeu d'ACL très poussé permet de créer des utilisateurs à **différents profils sur l'interface**.

4.13 Fichiers de configuration

En utilisant une base de données, des systèmes de modèles dont Nagios n'a pas connaissance, et en générant une configuration Nagios automatique, Centreon prend la main sur celle-ci de façon définitive. **Aucune factorisation** n'est effectuée sur le fichier, qui devient alors inutilisable pour une reprise « à la main » de la configuration. On peut noter enfin que cette dépendance de l'outil de gestion entraîne un léger retard sur les versions de Nagios puisque les mises à jour de l'un et l'autre ne sont pas toujours synchronisées.

5 Service de proximité

Les salles TP sont amenées à être supervisées. Pour autant, ce ne sont pas les même administrateurs qui sont concernés. Ainsi, il peut être souhaitable de monter **un second serveur dédié au service de proximité**. En s'assurant que ce serveur n'est pas dépendant des mêmes branches du réseaux que le Nagios principal, il serait envisageable que l'un surveille l'autre et avertisse les personnes appropriées. Cette solution **résout en partie la question de l'inaccessibilité du serveur Nagios**.

6 NagVis

Parmi les moyens de communication des notifications alternatifs, il y a **l'écran dans la salle des administrateurs**. En couplant cette possibilité avec NagVis, qui permet de générer des cartes de disponibilités, il deviendrait inutile de rester les yeux rivés sur son écran dans l'angoisse d'une alerte. D'un seul coup d'oeil, un problème peut être détecté et localisé.

La carte NagVis peut ensuite être associé à des **photos de baies**. Une carte générale de l'INPL proposerait des points colorés pour chaque bâtiment, à partir desquels on pourrait avoir à un plan du bâtiment avec des points colorés pour chaque salle qui héberge des machines, à partir desquels on pourrait avoir accès à ces photos qui pointeront alors directement les machines fautives.

7 Distribution FAN

Promise à un grand avenir dans le monde de la supervision, elle permet d'**installer aisément et rapidement un Nagios, un Centreon, un Nagvis ainsi qu'un Noreto** (outils de monitoring) et un serveur de mails. Elle est basée sur une CentOS et met vingt minutes pour s'installer et être prête à configurer.