



GNU

LINUX

MAGAZINE / FRANCE

France Métro : 6,50€ - DOM : 7,00€ - TOM : 9,50XPF - BEL : 7,50€ - LUX : 7,50€ - PORT. CONT. : 7,50€ - CH : 13,8FS - CAN : 13\$CAD - MAR : 75DH

Juillet/Août 2008

HORS-SÉRIE N°37

Complétez l'installation de votre

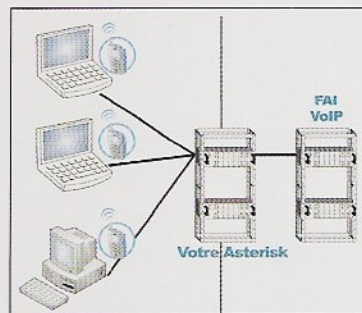
Serveur DÉDIÉ

Supervision, Streaming, VoIP, VPN, Syslog...

Bonus

VoIP illimitée avec Asterisk

ou comment installer Asterisk sur votre serveur dédié pour passer et recevoir vos appels comme si vous étiez chez vous.



P. 49

Supervision

Découvrez les nouveautés de Nagios 3. Une solution mature en passe de devenir la pierre angulaire des architectures de supervision dans le monde de l'Open Source.

P. 20

Diffusion audio

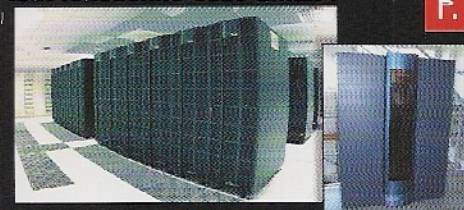
Utilisez votre serveur dédié pour streamer vos flux audio MP3, OGG ou AAC en toute simplicité avec IceCast2, IceS2 et DarkIce.

P. 34

Reportage

Visitez le centre de calcul scientifique du Commissariat à l'Energie Atomique (CEA), une véritable usine à téraflops où presque tout fonctionne sous Linux !

P. 04



Un numéro spécial pour gérer votre serveur dédié !

V.D.S.

- *Un système dédié complet sans aucune limitation.
(Distribution Debian accès root intégral - panel de gestion ultra simplifié).*
- *Des ressources réservées et confortables
(Ram, CPU, I/O, Bande Passante).*
- *Un prix bas pour un rapport qualité/prix unique.*

Bande passante incluse de 2 Mbps à 6 Mbps en trafic illimité - Espace disque de 4 à 40 Go - mémoire vive de 64 Mo à 512 Mo



php



www.sivit.fr

V.D.S.

Serveur Dédié Virtuel



Reportage

p. 04 ■ Le CEA plébiscite les superclusters et l'Open Source

Supervision

p. 20 ■ Actualité : découvrez les avancées de Nagios 3

p. 26 ■ Introduction à la supervision avec Zabbix

Services

p. 34 ■ Monter un service de diffusion audio avec Icecast2

p. 40 ■ Chrootez vos connexions SSH

p. 44 ■ Architecture Haute Disponibilité avec Lighttpd et JBoss

p. 49 ■ VoIP illimitée avec Asterisk

Entretien avec

p. 58 ■ Odile Boutleux

p. 61 ■ Émile Heitor

Outils

p. 64 ■ MultiTail, un super-tail pour garder un œil ouvert sur vos journaux

p. 66 ■ Maîtriser le système de gestion des logs/journaux



Encore ! Encore ! Encore !

Maintenir un ou plusieurs serveurs dédiés est une grande aventure, une grande et belle aventure.

Résumé des épisodes précédents : vous êtes un villageois heureux. Si vous avez lu les hors-série 35 et 36, vous disposez d'une belle auberge bien équipée et d'un magnifique bureau de poste avec un service de tri performant. Pourquoi vous arrêter là ?

Pourquoi ne pas installer le téléphone, une radio locale, un commissariat de police, des chambres d'hôtes, et un journal local (oui, cette dernière métaphore est un peu maladroite, je fais comme je peux avec les maigres compétences poétiques qui sont les miennes) ? En d'autres termes, pourquoi ne pas vous essayer à la VoIP over VPN, au *streaming*, à la supervision, à la gestion de comptes utilisateurs SSH *chrootés* et à un peu plus de compréhension du système Syslog/Syslog-ng ? Ne cherchez pas, vous n'avez pas d'excuse valable (le temps est une question de perception).

Les articles présents dans les pages qui vont suivre vous apprendront à exploiter au mieux votre serveur dédié et à ne pas simplement vous limiter au Web et à la messagerie. Ce troisième volet est donc complémentaire aux deux précédents sans en être dépendant. Nous ne parcourons qu'une petite partie des services qu'il est possible d'installer sur un serveur dédié. Non, il n'y aura pas de quatrième épisode. Vous devrez, pour la suite, laisser aller votre imagination pour occuper au mieux l'espace disque et les ressources de votre serveur. Comme le précise l'un des sysadmins interviewés pour ce numéro, il s'agit de bien plus que d'administration système. Il s'agit de construire une œuvre. À vous de construire la vôtre...

Nous ferons également un petit aparté en vous parlant des « jouets » qu'utilisent les grandes personnes du CEA. Cela, en plus de constituer de la lecture de loisir (comprendre « consommable ailleurs qu'à côté d'un clavier »), vous fera peut-être comprendre qu'une certaine humilité reste toujours de rigueur. Au petit concours de qui a la plus grosse (infrastructure/configuration), il y a toujours quelqu'un devant vous :)

Je vous laisse, sans plus attendre, découvrir cela par vous-même et vous donne rendez-vous le 16 août prochain pour un nouveau numéro hors-série. D'ici là, peut-être aurons-nous l'occasion de nous croiser à Mont-de-Marsan entre le 1er et le 5 juillet pour les RMLL (*Rencontres Mondiales du Logiciel Libre*).

Denis Bodier

GNU/Linux Magazine France
Hors-série

est édité par Diamond Editions
B.P. 20142 - 67603 Sélestat Cedex

Tél. : 03 88 58 02 08

Fax : 03 88 58 02 09

E-mail :
lecteurs@gnulinuxmag.com

Service commercial :
abo@gnulinuxmag.com

Sites : www.gnulinuxmag.com
www.ed-diamond.com

Directeur de publication :
Arnaud Merzler

Rédacteur en chef :
Denis Bodier

Secrétaire de rédaction :
Véronique Wilhelm

Relecture :
Dominique Grosse

Conception graphique :
Fabrice Krachentfels

Responsable publicité :
Tél. : 03 88 58 02 08

Service abonnement :
Tél. : 03 88 58 02 08

Impression :
VPM Druck Allemagne

Distribution France :
(uniquement pour les dépositaires de presse)

MIP Réassort :
Plate-forme de Saint-Barthélemy
d'Anjou.
Tél. : 02 41 27 53 12

Plate-forme de
Saint-Quentin-Fallavier.
Tél. : 04 74 82 63 04

Service des ventes :
Distri-médias :
Tél. : 05 61 72 76 24

IMPRIMÉ en Allemagne - PRINTED in Germany

Dépôt légal : À parution / N° ISSN : 1291-78 34

Commission paritaire : 09 08 K78 976

Périodicité : Bimestrielle

Prix de vente : 6,50 €

WWW.GNULINUXMAG.COM

La rédaction n'est pas responsable des textes, illustrations et photos qui lui sont communiqués par leurs auteurs. La reproduction totale ou partielle des articles publiés dans Linux Magazine France est interdite sans accord écrit de la société Diamond Editions. Sauf accord particulier, les manuscrits, photos et dessins adressés à Linux Magazine France, publiés ou non, ne sont ni rendus, ni renvoyés. Les indications de prix et d'adresses figurant dans les pages rédactionnelles sont données à titre d'information, sans aucun but publicitaire.

Le CEA plébiscite les superclusters

Cet article vous invite à découvrir les supercalculateurs civils du CCRT à Bruyères-le-Châtel (Île-de-France). Ce centre de calcul scientifique du Commissariat à l'Énergie Atomique (CEA) est une véritable usine à TéraFLOPS, au service des chercheurs et des industriels. Et presque tout fonctionne avec le noyau Linux !

1 La course aux TéraFLOPS

Revenons deux ans en arrière, dans le numéro 84 de ce magazine. À la suite d'une annonce étonnante dans le catalogue des ventes domaniales (un NEC SX-5 mis à prix à 10000 euros), je suis parti assister au démontage d'un supercalculateur vectoriel et, de fil en aiguille, à la découverte de l'IDRIS et de son parc de serveurs, destinés aux chercheurs du CNRS. Dans la conclusion, j'y lançais comme une boutade le défi de faire un reportage sur des machines emblématiques telles Earth Simulator, Tera ou même Tera10. Évidemment, il faudrait pour cela lever *quelques limitations...*

Earth Simulator est basé sur la technologie du NEC SX-6 [1], une génération de calculateurs vectoriels en cours de remplacement (les SX-5 étant remplacés depuis deux ans). À sa mise en service en 2002, ses 35 TFLOPS ont fait grand bruit, les Américains se trouvant pris au dépourvu par ce qu'ils ont appelé un *computenik* (en référence au lancement du Sputnik en 1957). Aussi, il est remarquable que ce vainqueur mondial toutes catégories (premier au Top500 durant presque trois ans) soit une machine vectorielle, alors que tout le monde se convertissait aux *clusters* de processeurs scalaires (dont le rapport performance/prix est supérieur malgré une efficacité moindre). Mais les machines vectorielles ne sont pas mortes, car la génération SX-9 est en cours d'installation chez les premiers clients. NEC vient aussi d'annoncer pour 2009 l'augmentation à 130 TFLOPS de ce supercluster [2], dont le développement a commencé en 1997 (ce qui est aussi une incroyable longévité). Mais

une visite à Yokohama serait trop chère et je ne parle pas le japonais.

Quant au cluster français Tera, avec ses 2560 processeurs Alpha à 1 GHz totalisant presque 4 TFLOPS soutenus, il a permis à la France de pousser un grand « cocorico ! » en se classant numéro 4 mondial, juste derrière le Japon et les USA, lors de son entrée au Top500 en 2002. Il est aussi resté numéro un européen pendant trois ans, durant lesquels il a fait tourner des codes destinés à « garantir le fonctionnement des bombes et réacteurs atomiques français ». C'est en effet un calculateur destiné aux applications de défense. Ses caractéristiques sont assez bien connues, mais les programmes ou les données qu'il utilise sont classés « secret ». Malheureusement (pour moi), ce cluster a été démantelé en 2007 et revendu pour pièces à son constructeur (aujourd'hui HP), afin d'assurer la maintenance des serveurs commerciaux encore en service et basés sur la technologie de l'Alpha. Et de toute façon, en dehors des rares visites de presse, il est très difficile d'obtenir l'autorisation de s'approcher du cluster.

Depuis 2006, le successeur de Tera est Tera10, un autre cluster dix fois plus puissant. Étant exploité par les mêmes équipes, il est donc aussi difficile de lui rendre visite. Pourtant, Tera10 est encore plus intéressant, et en fait tout est notable dans cette machine. On peut facilement trouver un certain nombre d'informations [3, 4, 5], dont voici une petite synthèse personnelle.

2 Tera10

Ce cluster a réussi à atteindre la cinquième place mondiale en juin 2006, à une époque où les USA essayaient par tous les moyens de restaurer leur suprématie dans le domaine et de faire oublier l'affront d'Earth Simulator. Les autres pays d'Europe se sont aussi remis dans la course et Tera10 n'était plus que 19e en novembre dernier. L'expérience de Tera a été prise en compte, en particulier avec un système de stockage extrêmement puissant, et les logiciels *open source* (sous forme d'une distribution Red Hat Linux adaptée par Bull) ont remplacé les systèmes d'exploitation propriétaires et spécifiques.

Le processus d'acquisition d'une telle machine montre que le secteur a beaucoup évolué. Il y a 15 ans, un client regardait les offres des constructeurs et choisissait le modèle de serveur qui correspondait à son budget et ses besoins. La situation actuelle est inverse : c'est le client qui définit les caractéristiques de la machine et les constructeurs dimensionnent leurs produits existants pour répondre à l'appel d'offre. Ceci est possible aujourd'hui, car les ordinateurs ne sont plus des serveurs monolithiques, mais des grappes de serveurs, extensibles à la demande. Même les calculateurs vectoriels suivent ce principe, dont la validité dépend de la

et l'Open Source

capacité du réseau d'interconnexion à s'agrandir en restant aussi efficace (la *scalabilité*).

Les 60 TFLOPS théoriques de Tera10 sont capables de soutenir 52 TFLOPS en *benchmark*, et ce nombre est divisé par 5 environ dans les programmes réels, d'où le nom « Tera10 » et non « Tera50 » ou « Tera60 ». Cette puissance est fournie par 544 serveurs Bull de type Novascale 6160, contenant 8 puces Itanium2 Montecito double cœur à 1,6 Ghz, soit 16 cœurs par rack. Les serveurs sont reliés par un réseau redondé Quadrics à très faible latence et très haut débit. 56 autres serveurs sont dédiés aux entrées-sorties et 2 à l'administration, ce qui porte le nombre total de cœurs à 9632 !

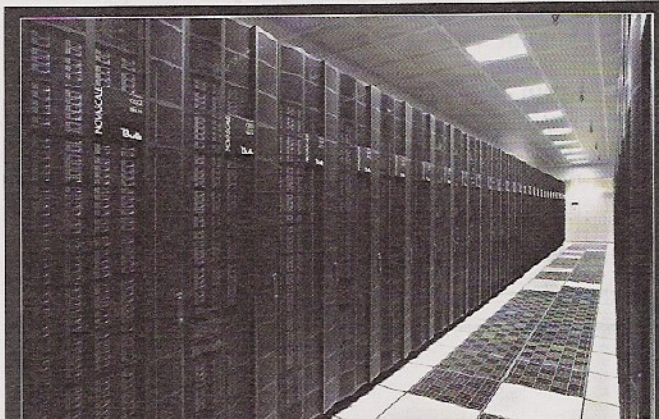


Figure 1 : Aperçu d'une rangée de baies composant Tera10. Derrière les premières portes, on devine les tiroirs de disques durs des racks de stockage. (crédit : CEA)

La mémoire joue aussi un rôle critique, pour que les calculs soient confortablement réalisés et que les résultats soient sauvegardés sans perdre de temps. Tera10 est *généreusement* doté de 30 téraoctets de RAM répartis dans les serveurs, et 1 pétaoctet de disques de travail dans des racks Data Direct Network.

Le tout est appuyé par un système d'archivage temporaire de 4 pétaoctets sur 10000 disques SATA, contrôlés par 13 serveurs Bull à base de processeurs Intel Xeon. Un serveur IBM se charge enfin de 5 silos SUN/STK pour le stockage sur bandes de longue durée.

Tout cela nécessite des infrastructures énormes. Les calculateurs consomment environ 1,8 MW et il faut encore plus de puissance pour les refroidir (l'énergie qui entre a besoin d'énergie pour ressortir). Avec les armoires de stockage, cela fait plus de 3 MW de consommation, et le centre a été prévu pour offrir une capacité totale de 5MW. Un raccordement supplémentaire au réseau électrique à haute tension a donc été spécialement installé.

Un bâtiment de 60 mètres de côté a aussi été construit pour l'ensemble du système, sur l'emplacement d'un ancien terrain de football. La climatisation, le stockage des données et les réserves de secours (en cas d'interruption du courant) sont aussi très imposantes. Mais, pour l'instant, seulement 800 m² des 2000 m² de faux plancher disponibles sont utilisés, car le reste accueillera le successeur (Tera100) déjà prévu pour 2010.

3

Le rôle de la simulation

Tous ces investissements et ces efforts ne cherchent pas à satisfaire une lubie de scientifiques ou à gagner au *concours de la plus grosse machine*. En fait, la simulation des phénomènes physiques est un vrai *trou noir à calculs*, surtout dans le domaine de la physique nucléaire qui est critique pour la défense et la dissuasion militaire.

Le développement d'une bombe ou d'un réacteur nucléaire est donc plus qu'une activité scientifique ou d'ingénierie : elle demande une quantité phénoménale de calculs. Déjà, à l'époque du projet Manhattan, les calculs nécessaires étaient tellement lourds qu'une armée de personnes (femmes appelées *calculatrices* et équipées de machines mécaniques) ne pouvaient donner qu'une approximation. Par la suite, l'informatique de pointe a été en partie « subventionnée » par les commandes des agences militaires.

Aujourd'hui, avec les données recueillies après des centaines d'expériences nucléaires, les modèles mathématiques sont beaucoup plus précis, ce qui augmente d'autant plus la complexité des calculs. Les modèles font appel à de multiples branches de la physique (mécanique des fluides, photonique...) qui sont *couplées* ou associées pour donner

une image globale du phénomène observé. Dans le meilleur des cas, les temps de simulation sont quadratiques, ce qui fait que même en multipliant par 1000 la capacité de calcul d'un ordinateur, on obtient à peine un facteur 10 dans la résolution des résultats.

En décidant d'arrêter ses essais nucléaires en 1995, la France a compté sur les outils de simulation, appuyés par son expérience passée, pour « maintenir la crédibilité de ses armes ». Elle s'est donc lancée dans un projet ambitieux, baptisé « programme Simulation », qui associe les ordinateurs Tera, le Laser MégaJoule (LMJ) près de Bordeaux, la machine de radiographie AIRIX à Moronvilliers, et d'autres accélérateurs de particules. Le but est de valider toutes les étapes du fonctionnement d'une arme nucléaire, uniquement par calcul et la mise au point de nouveaux outils d'investigation.

Les quantités de données sont gigantesques et j'imagine que même Tera10 peine à la tâche, ses « expériences numériques » générant de 10 à 30 To (utiles) par jour ! Ces défis sont bien connus du CEA qui a décidé dès le début du programme Simulation d'aller au-devant de l'évolution technologique normale : il a été prévu de commander successivement

3 ordinateurs, chacun ayant des performances dix fois plus élevées que son prédécesseur, à un rythme plus rapide que ce que l'industrie fournit normalement.

Indirectement, cela change beaucoup les préoccupations des scientifiques, des politiques et du public : pour les pays ayant ratifié les traités internationaux bannissant les essais, la question est maintenant de trouver un succédané au tir réel. On se concentre donc plus sur la théorie, la physique fondamentale et les architectures d'ordinateurs, que sur la production en série et la détonation de têtes dont on aurait modifié un paramètre ou un autre pour voir si ça fonctionne mieux.

Mais les avantages vont bien au-delà de l'absence de retombées ou de fuites radioactives. Les investissements économiques ou technologiques sont plus facilement recyclés dans le

tissu social : la maîtrise du façonnage de l'uranium enrichi présente moins d'intérêt dans une société de l'après guerre

froide que la maîtrise de l'informatique. La puissance de calcul est aussi considérée comme un indicateur de la puissance du pays : le Top500 reflète cette préoccupation, et l'analyse de ses classements évoque toujours des remarques politiques ou économiques, au-delà de l'intérêt scientifique initial.

Tera est donc une machine hyper-puissante, absolument imposante et soigneusement conçue par des spécialistes français. Mais son usage est protégé par le secret défense, et je ne savais pas que les seules occasions de lui rendre visite étaient lors de visites organisées, comme celle de janvier 2006 [3] ou de septembre 2006 [6].

Tera100 ?

Tera100 a été imaginé dès le début du programme Simulation et devrait atteindre une puissance théorique d'environ 500 ou 600 TFLOPS. Pourtant, son architecture, son constructeur ou ses caractéristiques sont encore inconnus, car l'appel d'offre vient seulement d'être lancé. L'installation a été prévue pour avoir lieu en 2009 et la machine serait opérationnelle en 2010. Cela ne laisse pas beaucoup de temps aux constructeurs pour la concevoir. Ils doivent donc utiliser les technologies existantes et cela les incite à renforcer leur R&D en amont.

Bull semble bien placé grâce à l'expérience positive de Tera10, mais je pense que les Itanium ont du mal à augmenter leur puissance suffisamment vite et la concurrence est rude : IBM propose maintenant les calculateurs massivement parallèles (MPP) de la famille « BlueGene », au rapport performance/consommation très élevé et fonctionnant sous Linux. Or, le CEA désire spécifiquement des clusters de SMP, et je me demande comment ils pourront atteindre rapidement 500 TFLOPS en gardant les contraintes et l'infrastructure actuelles... Rendez-vous en 2010 !

4

« There is another system. »

Ceux qui ont vu le film *Colossus : the Forbin project* [7] se souviennent de cette phrase. J'ai d'ailleurs été aussi étonné que les personnages de ce film de 1970, lorsque j'ai appris qu'un système « civil » similaire à Tera10 existait.

Lors du salon *Solutions Linux* de février 2007, j'en ai profité pour trol^Wdiscuter de Tera10 sur le stand de Bull. Évidemment, les responsables des relations publiques ne pouvaient m'aider pour visiter le super-cluster que leur compagnie avait construit. Mais qui ne tente rien n'a rien et je fus surpris d'apprendre l'existence d'un autre ordinateur assez similaire, mais à usage académique et industriel : une visite était donc plus facilement envisageable.

Le responsable du stand Bull m'a cordialement donné les coordonnées d'une collègue du service de presse. À partir de là, de coups de téléphone en recherches sur Internet, de redirections en voies sans issues, et au bout de pas mal de temps, j'ai finalement pu joindre les bonnes personnes au CCRT. J'apprends alors que de temps en temps, des conférences ou des présentations ont lieu sur le site, ce qui est une occasion de visiter la salle des ordinateurs.

Pour pouvoir accéder au CCRT, il ne faut pas seulement se munir d'une pièce d'identité, mais aussi donner son pedigree complet bien à l'avance. Je donne alors toutes les informations personnelles nécessaires et j'attends, en continuant à fouiller sur le web.

5

Le CCRT

Le Centre de Calcul Recherche et Technologie fournit des moyens de calcul intensif aux chercheurs du CEA (voir le Projet Horizon plus loin) ainsi qu'à des partenaires académiques et industriels [8]. Depuis 2007, sous l'égide de GENCI, 20% des ressources de calcul du CCRT sont ouvertes à l'ensemble de la communauté de la recherche française (CNRS et enseignement supérieur) [9]. Une cinquantaine d'autres établissements publics ou privés, regroupés dans une structure appelée Teratec (association créée en août 2005), participent au développement de la simulation numérique française.

Contrairement aux clusters de la DAM (défense) ou à l'IDRIS (CNRS), c'est donc un centre de calcul civil et mixte [10]. Aussi, à l'inverse de Tera10 qui n'est pas connecté à l'extérieur pour des raisons évidentes, les serveurs du CCRT sont accédés de toute la France par un réseau sécurisé, pour soumettre des tâches ou transférer les paramètres de calcul.

Le CEA a réuni ses calculateurs « défense » et « recherche » sur le site de Bruyères-le-Châtel pour mettre en commun de nombreuses ressources, en particulier la ligne haute tension et les compétences des développeurs et des experts en calcul intensif. Cependant, il y a un compartimentage

physique et psychologique, et, à mesure qu'on s'approche des « applications militaires », le secret s'épaissit.

D'ailleurs, pour préparer la future visite, j'ai cherché où se trouve le site et comment s'y rendre. Je n'ai pas de voiture et les indications routières du site du CEA [11] me sont inutiles. Il y a aussi une gare RER, mais celle-ci est très loin du CCRT et je ne trouve pas de ligne de bus... Cela s'annonce plus difficile que la visite à l'IDRIS.

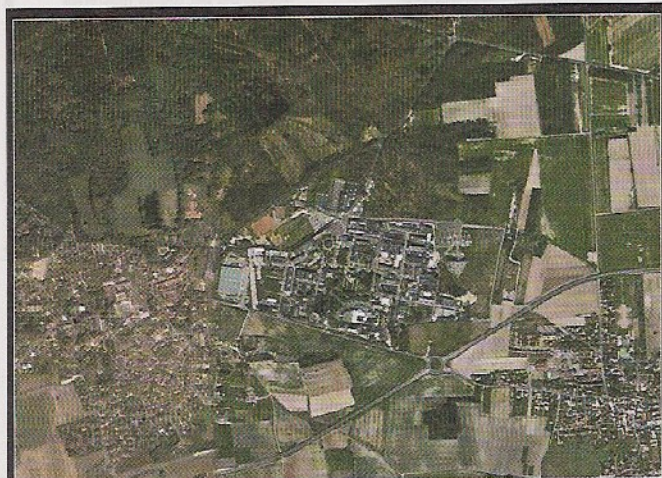


Figure 2 : Sur <http://maps.google.com>, le village de Bruyères-le-Châtel est visible à très haute résolution alors que l'image du site du CEA est floue et vieille de plusieurs années. On ne plaisante pas avec la sécurité nationale...

Mais ce qui m'intéresse le plus est la visite du parc de calculateurs. Le site web du CCRT [12] donne les caractéristiques principales de ses machines, car il y en a en fait plusieurs, et je commence à faire connaissance avec elles.

6 Tourisme scientifique

Je reçois enfin un email de Mme Ménaché, responsable opérationnel au CCRT, m'annonçant qu'un voyage de presse aura lieu le surlendemain, le mercredi 16 avril. D'autres contacts téléphoniques m'apprennent qu'un transport en car est assuré par le CEA, ce qui évite bien des soucis pour tout le monde.

Pour cette excursion, je suis accompagné par Laura, ma geekette dévouée, pour récolter un maximum d'informations. Le rendez-vous est fixé à 9h30 devant la gare de Denfert Rochereau, et je suis tellement électrique que nous arrivons à 9h. Nous sommes rejoints progressivement par une douzaine d'autres journalistes d'horizons divers et, en attendant le départ, nous patientons en découvrant le dossier de presse [13] imprimé pour l'occasion. J'ai presque deux heures, entre l'attente du départ et la route bouchonnée, pour essayer de comprendre quelques annonces surprenantes et préparer des questions à leur sujet...

Finalement, en sortant de la route départementale 116, après Arpajon, au milieu de la forêt, des murs recouverts de barbelés surgissent, entourant des bâtiments d'âges variés, aux toits

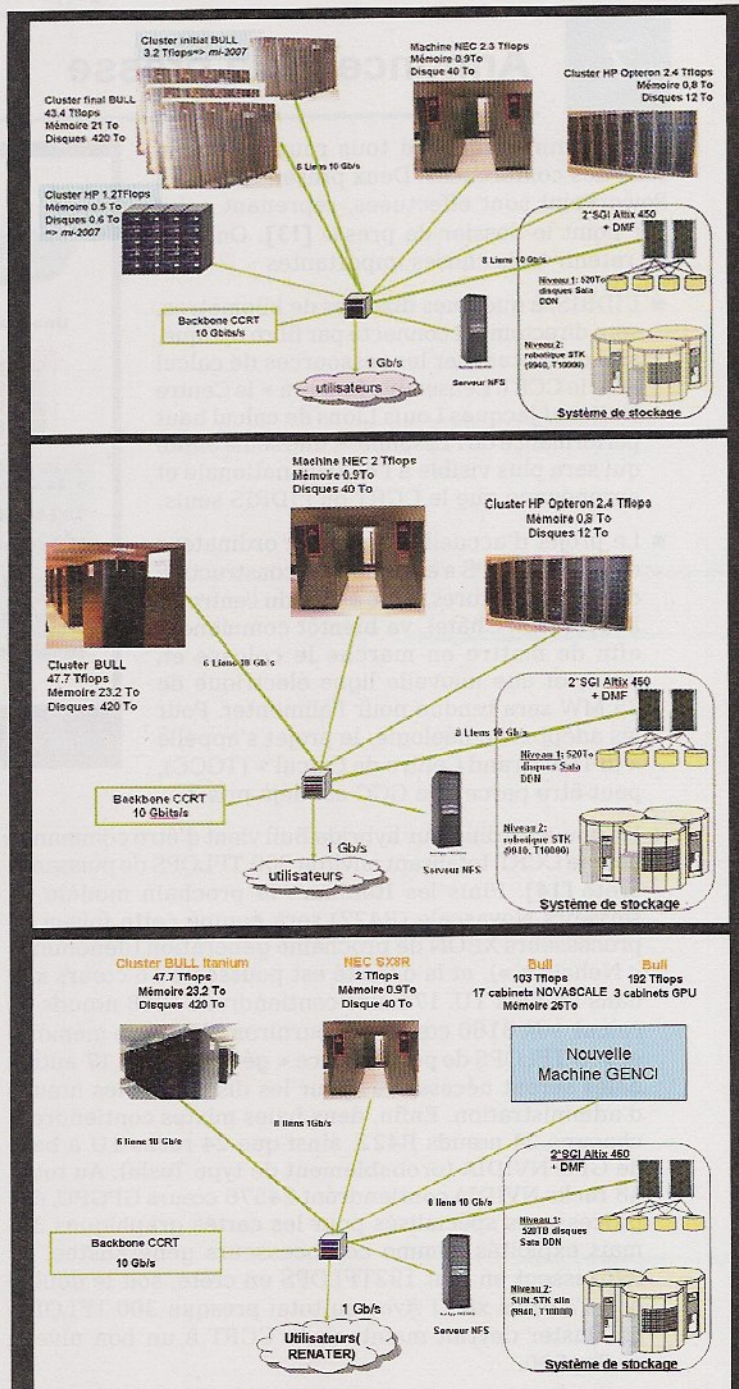


Figure 3 : Évolution du parc du CCRT : passé, présent et futur proche (crédit : CEA)

surmontés d'antennes et de paraboles. Il faut absolument éteindre son téléphone portable, et aucune photographie n'est permise, sauf sur autorisation expresse du personnel qui nous encadre. C'est une atmosphère étrange où nous sommes surveillés (on ne sait jamais), mais bienvenus car nous sommes ici pour parler du CEA et en donner la meilleure image possible. La préparation et le sérieux de chacun fait que tout se passe très bien (malgré le retard d'un journaliste qui décalera tout le programme de la visite). Les contacts sont francs et cordiaux, et je me retrouve rapidement en possession d'une demi-douzaine de cartes de visite de personnes qui se proposent de m'aider pour l'article.

7

Annonces à la presse

Nous sommes d'abord tous réunis dans la salle des conférences. Deux présentations en PowerPoint sont effectuées, reprenant point par point le dossier de presse [13]. On peut en retenir trois choses importantes :

- LIDRIS, à quelques dizaines de kilomètres, sera directement connecté par fibre optique, afin de mutualiser les ressources de calcul avec le CCRT. L'ensemble formera « le Centre National Jacques Louis Lions de calcul haut performance de l'Essonne », une seule entité qui sera plus visible à l'échelle nationale et européenne que le CCRT ou l'IDRIS seuls.
- Le projet d'accueillir un grand ordinateur de 1 PétaFLOPS a été lancé. La construction des infrastructures, juste à côté du centre de Bruyères-le-Châtel, va bientôt commencer afin de mettre en marche le colosse en 2011, et une nouvelle ligne électrique de 25 MW sera tendue pour l'alimenter. Pour les adeptes de néologie, le projet s'appelle « le Très Grand Centre de Calcul » (TGCC), peut-être parce que GCC est déjà pris ?
- Un nouvel ordinateur hybride Bull vient d'être commandé pour le CCRT, totalisant environ 300 TFLOPS de puissance crête [14]. Finis les Itanium : le prochain modèle de serveurs Novascale (R422) sera équipé cette fois-ci de processeurs XEON de prochaine génération (dénommée « Nehalem »), et la densité est poussée à 16 cœurs x86 dans un rack 1U. 17 baies contiendront 1068 nœuds de calcul, soit 8160 cœurs, et fourniront 25 TO de mémoire et 103 TFLOPS de performance « généraliste ». 17 autres baies seront nécessaires pour les disques et les nœuds d'administration. Enfin, deux baies mixtes contiendront chacune 24 nœuds R422, ainsi que 24 racks 1U à base de GPU NVIDIA (probablement de type Tesla). Au total, 48 racks NVIDIA contiendront 24576 cœurs GPGPU, des processeurs spécialisés pour les cartes graphiques 3D, mais exploités comme coprocesseurs généralistes. Ils fournissent en tout 192TFLOPS en crête, soit le double des 17 baies x86 ! Avec au total presque 300 TFLOPS, ce cluster devrait maintenir le CCRT à un bon niveau au Top500.

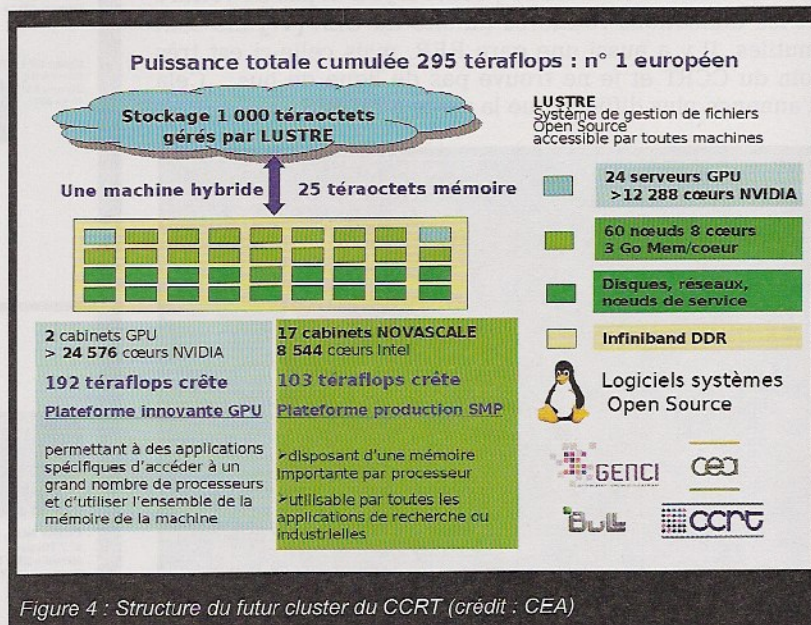


Figure 4 : Structure du futur cluster du CCRT (crédit : CEA)

Un détail, toutefois, n'échappera pas aux spécialistes : le nombre de FLOPS est normalement basé sur des opérations en virgule flottante sur 64 bits (double précision), alors que les GPU travaillent sur 32 bits (simple précision). Un mode d'émulation 64 bits est souvent possible (comme pour le Cell), mais il est plus lent. Cependant, Christophe Béhar, directeur du centre, m'assure que de nombreux types de programmes scientifiques sont de type « purement parallèles » et travaillent sur des données qui n'ont pas besoin de 64 bits de précision, comme les images ou d'autres types de signaux.

Le CCRT espère aussi stimuler la recherche dans le domaine des architectures hybrides, car cela apparaît comme une solution prometteuse pour faire tenir les prochains clusters, comme Tera100 ou celui du TGCC, dans les limites de temps, d'argent, de consommation et de place. D'ailleurs, ce futur cluster sera installé en 2009 à côté de Platine, et ses performances crête seront jusqu'à six fois plus élevées pour un encombrement presque deux fois inférieur !

8

Le petit musée du CEA

Après la conférence de presse et la première séance de questions-réponses, le groupe est convié à découvrir les machines existantes et le site du futur cluster (pour l'instant appelé « GENCI »). La première grosse surprise est que le CEA a placé d'anciennes machines Cray dans les couloirs qui mènent à la salle des machines. Mon excitation atteint déjà son comble : il est très rare de voir une telle collection de machines ! Le CEA a utilisé beaucoup d'ordinateurs et nombreux sont ceux qui ont été démontés et mis à la ferraille (phrase à lire en s'imaginant le bruit d'un cœur qui se fend). De plus, la

place dans les locaux est comptée et seuls trois ordinateurs, les plus fameux, ont été gardés (en excellent état). Ce sont les plus puissants de leur génération et proviennent de différents sites du CEA, ce qui explique certains « trous » dans la continuité des périodes d'exploitation. On remarque aussi que la période séparant l'introduction d'un modèle et sa décommission se réduit de plus en plus avec le temps : 14 ans, 11 puis 8 ans, et 4 ans seulement pour les machines actuelles : les évolutions technologiques et les conditions commerciales se sont emballées !

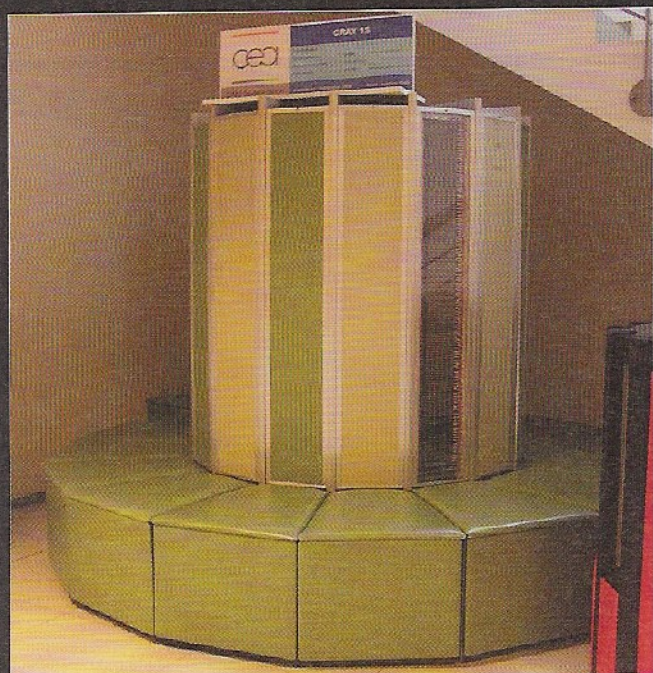


Figure 5a : Cray 1S : 1 CPU, 160 MFLOPS@80MHz, introduit en 1976, exploité entre 1982 et 1990



Figure 5b : Cray 2 : 4 CPU, 1,9 GFLOPS@240Mhz, introduit en 1985, exploité entre 1990 et 1994

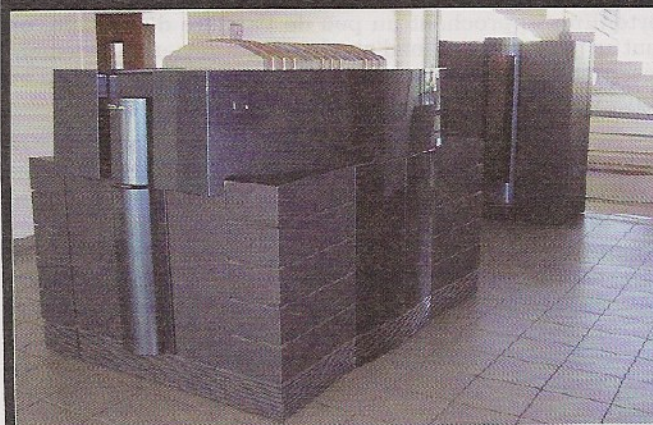


Figure 5c : Cray T90 : 24 CPU, 43 GFLOPS@455MHz, introduit en 1994, exploité entre 1996 et 2002

9

La salle des machines

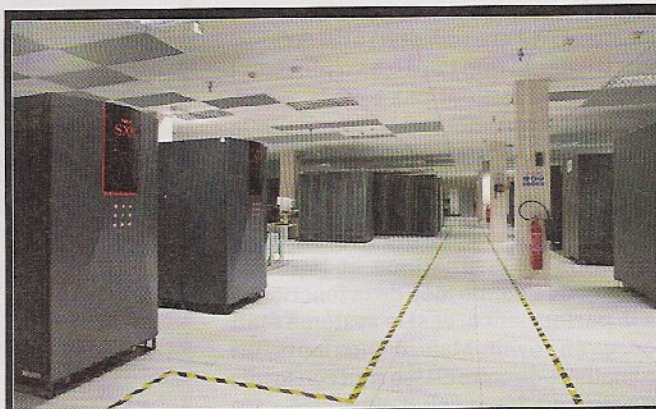


Figure 6 : Aperçu de la salle des machines avec les calculateurs vectoriels NEC à gauche, le cluster HP au centre et le cluster Bull à droite. Le stockage est au fond à gauche, caché par le cluster HP, et les frontaux NEC sont hors champ à gauche. Au fond à droite, un espace libre accueillera la future machine qui vient d'être annoncée. (crédit : CEA)



Figure 7 : Une des armoires électriques avec son indispensable gros bouton rouge \o/ Le compteur électronique indique que la consommation s'élève à 113KW, ce qui semble correspondre à l'alimentation des ordinateurs vectoriels. (photo : Laura)

Toutes les machines en exploitation sont réunies dans une grande salle. Il n'y a rien de bien particulier, le faux plancher est classique, les installations sont standards... La température est correcte et le bruit des ventilateurs et des climatisations n'est pas particulièrement fort, surtout pour un tel parc de machines.

10

Platine

La vedette de la salle en occupe environ le quart de la surface. Il faut savoir à l'avance que c'est Platine, sinon on s'imaginerait dans un centre d'hébergement mutualisé, conçu à partir de briques standardisées et reproduites à volonté, sans trop y penser... Au niveau du design, cela n'a rien à voir avec les Cray ou les NEC : les baies 19 pouces 42U sont tout ce qu'il y a de plus standard, le logo de Bull aurait pu être remplacé sans qu'on s'en rende compte.

Seymour Cray pensait que s'il construisait la machine la plus puissante et la plus chère du monde, augmenter son prix de 1% pour la rendre visuellement agréable était tout à

fait justifié. Cette démarche a beaucoup joué sur son image de marque, comme les calculateurs du musée du CEA en témoignent. Un riche client texan en aurait même profité pour décorer son Cray à la manière d'une peau de vache tachetée. C'est une exception anecdotique, mais il est vrai que j'ai déjà vu plusieurs Cray et ils n'avaient pas tous la même couleur, ce qui montre le soin dans la différenciation de chaque exemplaire.

Marcel Dassault pensait un peu de la même manière, « un bel avion est un bon avion » : l'esthétique et la forme donnent une bonne idée de la fonction et de la performance, et si le système était bon, ça se voyait. Mais, de nos jours, les ordinateurs de cette classe sont construits en réponse à des appels d'offre, et le « 1% design » n'entre pas dans le cahier des charges. Les budgets sont trop serrés pour ces considérations.

Platine a été installé en deux étapes. La version initiale s'appelait Argent et atteignait 3,2 TFLOPS. Le reste du cluster a pu être installé en parallèle de la phase de prise en main, jusqu'à la mise en service de l'ensemble à la rentrée 2007.

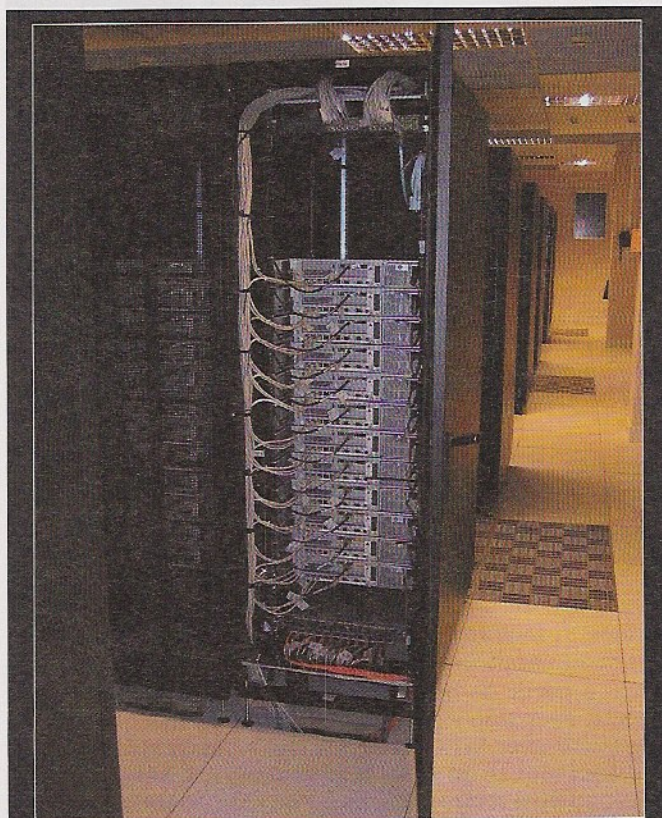


Figure 9 : Vue de l'intérieur d'une baie de platine, par derrière. Douze nœuds 2U et un switch Infiniband tiennent confortablement dans 42U.

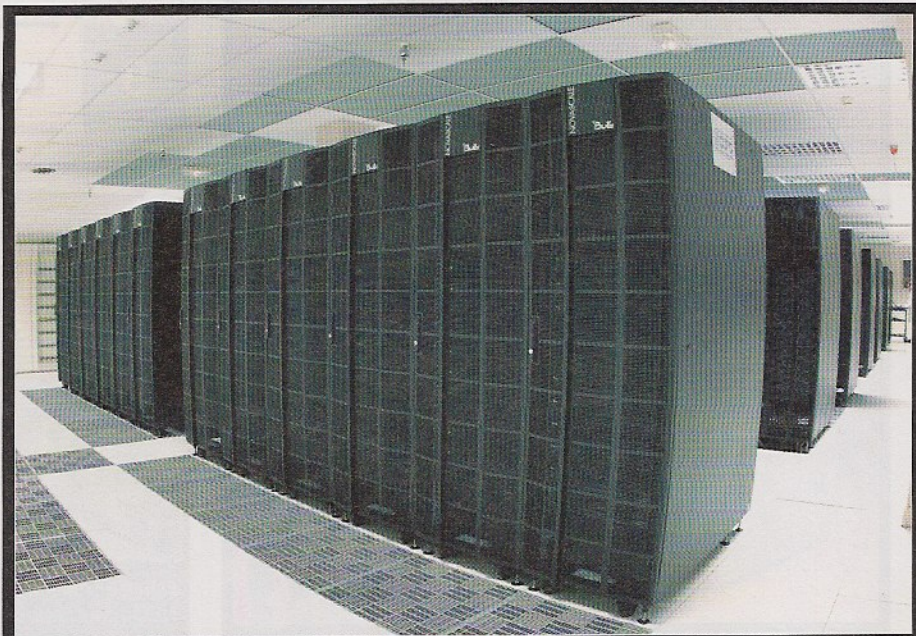


Figure 8 : Le cluster Platine est gigantesque... Difficile de photographier 84 baies d'un coup sans un objectif adapté ! (crédit : CEA)

Encore une fois, mon appareil photo n'a pas une ouverture suffisante et l'absence de recul m'a empêché de photographier le cluster dans son ensemble. Heureusement, la photothèque du CEA m'a fourni une image satisfaisante, bien que distordue : un objectif « fisheye » est nécessaire dans ce cas-là.

On voit qu'il reste de la place dans les armoires. En fait, en remplissant les armoires à bloc, il aurait été possible d'en utiliser moins. Par contre, le refroidissement par air atteindrait ses limites. Ou alors, l'espace libre aurait pu être prévu pour ajouter des nœuds ensuite. Mais il est à la fois plus rentable et pratique de construire un nouveau cluster, car ajouter des nœuds modifierait l'équilibre du système, particulièrement si les nouveaux nœuds, plus récents, ont des caractéristiques différentes. On voit qu'il reste de la place dans les armoires. En fait, en remplissant les armoires à bloc, il aurait été possible d'en utiliser moins. Par contre, le refroidissement par air atteindrait ses limites, il aurait peut-être fallu employer du refroidissement à eau car la densité serait trop forte. En tout cas, on voit qu'il y a encore de la marge pour augmenter la puissance au m², ce que le nouveau cluster s'apprête à faire. Pour commencer, les armoires seront plus hautes et auront une capacité de 48U.

En attendant, la densité de calcul de Platine est déjà très forte. En s'approchant un peu de l'arrière d'une baie, on sent un puissant courant d'air chaud, qui devient rapidement désagréable. Je me demande si le personnel du centre a eu l'idée d'y faire sécher son linge.

Platine n'est pas une « copie » de Tera10 et la **table 1** montre que le CCRT a essayé de s'équiper avec une machine « très similaire » tout en économisant là où c'était raisonnablement possible. La mémoire totale et les disques sont moins gigantesques, et les serveurs SMP à 4 processeurs (*Thin Node*) coûtent moins cher que les nœuds à 8 processeurs, mais ils n'étaient pas encore disponibles lors de l'appel d'offre pour Tera10.

^a De toute façon, c'est le nombre de TeraFLOPS qui compte pour entrer dans le TOP500.

	Tera10	Platine
CPU	Intel Itanium Montecito dual core @ 1,6 Ghz [15]	
Nombre de cœurs	8704	7456
Puissance crête	60TFLOPS	47TFLOPS
Serveurs	(544+58) × Novascale 6160	(932+26) × Novascale 3045
configuration	48GO et 8 CPU/nœud	24GO et 4CPU/nœud
RAM totale	30TO	23TO
Disques locaux	1PO (7800 disques)	420TO
Stockage externe	4PO	520TO (partagé)
interconnexion	QsNet II (Quadrics)	Voltaire (InfiniBand DDR)

Table 1 : Comparaison entre Tera10 et Platine

Il ne faut pas perdre de vue que deux ans séparent ces machines : les prix des composants ont baissé et les performances ont augmenté. En plus, l'expérience de Tera10 a permis au CEA de dimensionner Platine au plus juste, sans sacrifier sa performance^a. Par exemple, un réseau

d'interconnexion plus récent et encore plus efficace équipe Platine, ce qui permet de compenser un peu l'écart avec Tera10. N'oublions pas non plus que Platine doit s'interfacer directement avec le reste des autres machines du parc, et partager ses infrastructures de stockage des données.

11 Mercure : les calculateurs vectoriels NEC

En avril, deux types de calculateurs étaient présents : NEC SX-6 et SX-8R. Les SX-8R sont arrivés en novembre 2007 et les SX-6 sont enlevés en juin. Ne parlons donc pas de ces derniers, sauf du fait que cela forme pour l'instant un cluster hétérogène. Les SX-8 sont plus rapides, plus compacts, ont plus de RAM et surtout : ils ont un ordinateur frontal (TX-7) qui tourne sous Linux !

On peut aussi noter que c'est le même type de calculateur qui est installé à l'IDRIS [16], les différences étant le nombre de cabinets (10 au lieu de 8 au CCRT [17]) et la configuration de la mémoire. Un utilisateur peut donc demander des heures de calcul indifféremment au CCRT ou à l'IDRIS, le même code tournera partout. D'ailleurs, NEC étant le dernier fabricant de calculateurs vectoriels, le parc mondial est maintenant assez homogène. Il y a 15 ans, c'était la même chose, en remplaçant « NEC » par « Cray »...

Les SX-8R sont des versions améliorées du modèle SX-8, le « R » voulant officiellement dire « Reinforced » (officieusement, c'est « un mot japonais en R qui veut dire 'rapide' », mais, dans ce cas, je me dis que ça aurait pu tout aussi bien être un « L »). D'après les données disponibles (pour une fois, les données de Wikipédia [18] sont plus fiables que celles du CCRT [17]), les caractéristiques sont impressionnantes : chaque cabinet contient 8 CPU à 2,2 GHz fournissant chacun 35 GFLOPS. Chaque CPU est presque aussi puissant que le Cray T90 à l'entrée, mais beaucoup plus petit et moins gourmand. La puissance crête totale avec 8 cabinets atteint 2,2 TFLOPS et la nouvelle génération SX-9, en cours de déploiement sur d'autres sites, triple ou quadruple encore les performances...



Figure 10 : Vue d'ensemble des NEC SX-8R, dans leur élégante livrée noire. On aperçoit à droite un des SX-6, qui sera probablement retiré au moment où ce magazine paraîtra.

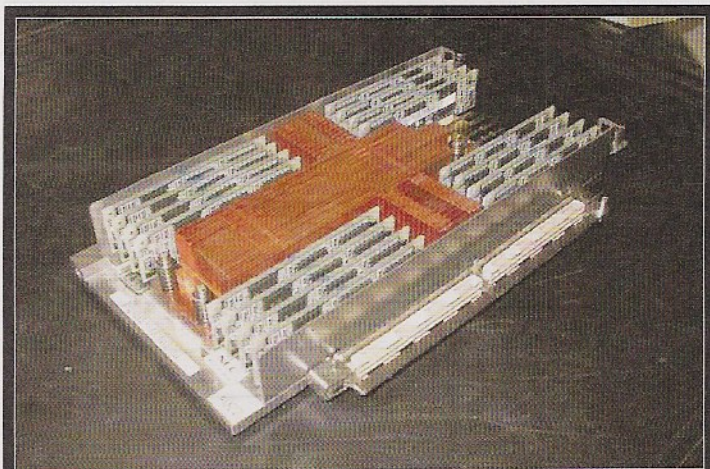


Figure 11 : Module de mémoire d'un SX-8 (crédit : NEC)

La mémoire n'a pas l'air très impressionnante : *seulement* 64 ou 128 Go par cabinet, soit 8 ou 16 Go par CPU, alors qu'un nœud de Platine, beaucoup moins cher, dispose déjà de 3 Go par cœur. Cependant, la nature de ces mémoires est totalement différente, puisque les SX sont des architectures à mémoire partagée (alors que les clusters de SMP sont à mémoire distribuée). Toute la mémoire du cabinet est accédée directement par les processeurs du même cabinet, et indirectement par tous les autres processeurs du cluster au travers de processeurs d'entrée/sortie. La bande passante est aussi beaucoup plus élevée que sur un SMP, car elle est optimisée pour le transfert de gros blocs. Chaque CPU accède à la mémoire à 70 Go par seconde, à comparer aux 17Go/s partagés par les deux cœurs des Montecito de Platine.

Pourtant, même avec 2,2 TFLOPS, la performance crête du cluster NEC reste 20 fois en dessous de Platine. En tenant compte du coefficient d'efficacité des codes en pratique, soit environ 50% sur du vectoriel contre 20% en scalaire, cela donne environ 1 TFLOPS contre 10 TFLOPS. Comme la consommation électrique ou le nombre de GFLOPS par mètre carré sont presque comparables, on peut se demander pourquoi un cluster plus grand n'a pas été commandé...

Une partie de l'explication est due aux coûts de développement, d'installation et de maintenance beaucoup plus élevés que pour des nœuds SMP, car ce sont des machines construites en très petites quantités. Le choix et la concurrence sont aussi quasiment absents, puisque NEC est presque seul sur ce marché, alors que les nœuds SMP sont disponibles partout (même chez NEC) et les prix sont beaucoup plus compétitifs. En plus, l'architecture vectorielle, en plus de nécessiter des composants hyper-spécialisés vendus au compte-goutte, est propriétaire et les logiciels, même s'ils sont souvent basés sur UNIX, ne bénéficient pas directement de la dynamique des logiciels libres (malgré [19]).

Mais, surtout, le principe vectoriel (le traitement par blocs) est à la fois son point fort et son point faible : ce n'est efficace que pour certains types de programmes traitant

des données très homogènes. Aujourd'hui, de nombreux modèles physiques tendent à considérer le comportement individuel et complexe de chaque particule d'un système, au lieu de l'intégrer dans d'immenses équations (solubles à grands coups d'opérations matricielles qui fonctionnent très bien en vectoriel). L'approche granulaire favorise les ordinateurs massivement parallèles (MPP) à base de processeurs scalaires, car les ordinateurs vectoriels ne sont pas capables d'explorer des dizaines de chemins d'exécution conditionnels simultanés. En fait, comme les instructions du SX-8 ne sont décodées qu'à 1,1 GHz, les performances scalaires sont faibles, et les opérations de compilation par exemple doivent être réalisées sur un ordinateur frontal.

Enfin, on ne peut pas mesurer la vitesse d'exécution d'un programme uniquement avec une seule métrique comme les FLOPS. Un ordinateur n'est pas juste une machine à calculer, il sert aussi à prendre des décisions : les ordinateurs vectoriels mettent l'accent sur le calcul pur au détriment d'autres caractéristiques. Le CEA continue de garder des calculateurs vectoriels pour maintenir les codes existants, mais incite les scientifiques à porter les logiciels sur les clusters de SMP. Ainsi, depuis plusieurs années, les bibliothèques logicielles sont progressivement adaptées aux ordinateurs à parallélisme fin, à la fois plus économiques et plus flexibles.

Il est intéressant de noter que depuis la génération SX-8, NEC propose un ordinateur frontal basé sur des processeurs Intel et fonctionnant sous Linux (Suse Linux Entreprise Server). Ce « frontal » est en fait une autre architecture SMP standard proposée par NEC, la famille TX-7 [20], qui est plus destinée à la gestion qu'au calcul scientifique, avec ses circuits redondants échangeables à chaud et sa capacité de partitionnement dynamique (pour travailler avec plusieurs systèmes d'exploitation simultanément). Le frontal installé au CCRT s'appelle Azama2 et dispose de 32 Go de mémoire, partagée par 8 processeurs Itanium2 double cœur à 1,4GHz.



Figure 12 : Un NEC SX-6 en sursis...

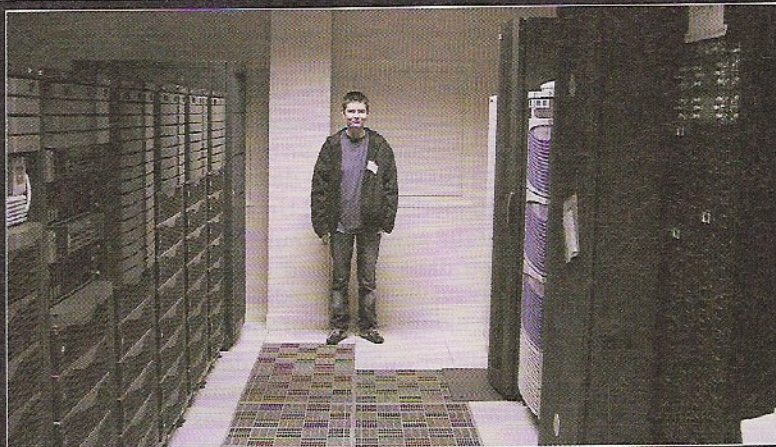


Figure 13 : Laura se sent toute petite au milieu des ordinateurs frontaux, baies de disques et autres commutateurs réseau...

Les cabinets SX-8 sont reliés entre eux par un réseau d'interconnexion interne extrêmement performant, appelé « IXS ». Ils disposent aussi chacun de 1 Tio de disques durs locaux. Le frontal et les SX-8 partagent en plus 40 To de stockage temporaire dans 8 baies de disques durs RAID. Le tout communique au moyen de liens FiberChannel à 2 et 4 Gigabits par seconde, eux aussi redondés, ce qui permet une bande passante soutenue de 2 Go par seconde. La liaison avec le reste du CCRT s'effectue au moyen d'un réseau Gigabit.

12 Tantale : le cluster d'Opteron

Ce cluster « classique », aussi puissant que les NEC SX-8, reçoit curieusement peu d'attention et les informations à son sujet sont plus rares. Je n'ai d'ailleurs eu aucun contact avec HP. L'essentiel est résumé sur la page de présentation du CCRT [21]. En fouillant un peu, j'ai trouvé juste qu'il a complété et succédé à un autre cluster HP de 1,2 TFLOPS (doté de seulement 500 Go de RAM et de 600 Go de disque), retiré du service durant l'été 2007.

Tantale est un cluster de 2,4 TFLOPS crête, basé sur des lames Proliant DL585 contenant 4 processeurs Opteron. Une mise à niveau a augmenté le nombre de nœuds de 60 à 140, les nouveaux nœuds travaillant à 2,4 GHz au lieu de 1,8 GHz. La majorité des nœuds de calcul dispose de 4 Go et de disques SCSI locaux. Le stockage est appuyé par 15 To de disques SATA accédés avec le système de fichiers Lustre.

Mme Ménaché a bien voulu m'en dire un peu plus :

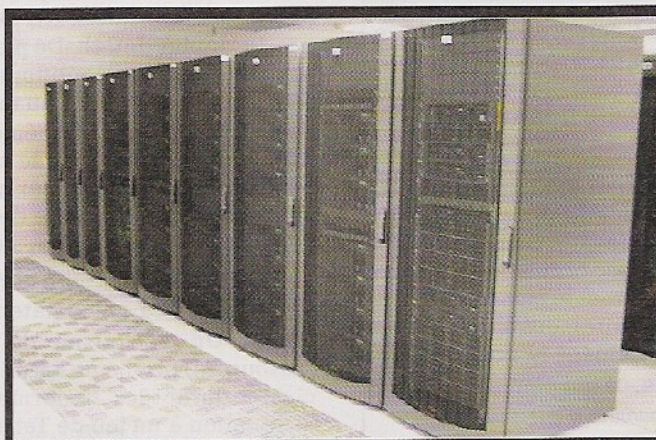


Figure 14

« Tantale a été une plate-forme innovante dans la mesure où cela nous a permis de tester et expérimenter la technologie Infiniband. Nous avons d'ailleurs été un des premiers sites HP à être équipé de ce type de cluster. Le portage de code de Tantale vers Platine est évidemment plus simple que d'une machine vectorielle vers Platine, puisque ce sont toutes les 2 des architectures SMP. Cependant, les coûts d'exploitation et de maintenance, ainsi que l'arrivée de Platine et de l'extension GENCI font qu'on devrait l'arrêter fin 2008. »

Tantale a pris son service en début 2005, et son prochain arrêt lui aura à peine permis quatre ans d'activité, malgré l'extension à la mi-2005. La vie des clusters est vraiment sans pitié.

13 Stockage

L'ensemble des machines partage un système de stockage à deux niveaux, connecté par 8 liens à 10 Gbps.

Depuis mars 2007, le premier niveau est constitué de 540 To de disques SATA redondants contrôlés par Acier; deux serveurs SGI Altix à base de processeurs Itanium2. Deux silos à bandes se chargent du deuxième niveau, de longue durée.

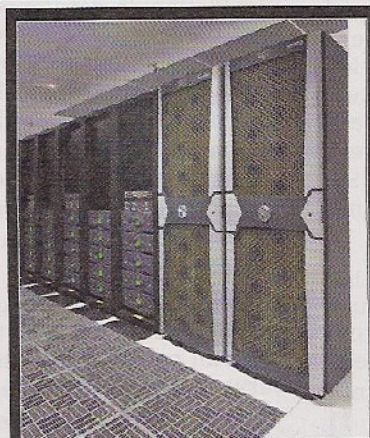


Figure 15 : Acier contrôle le stockage de premier niveau (crédit : CEA)

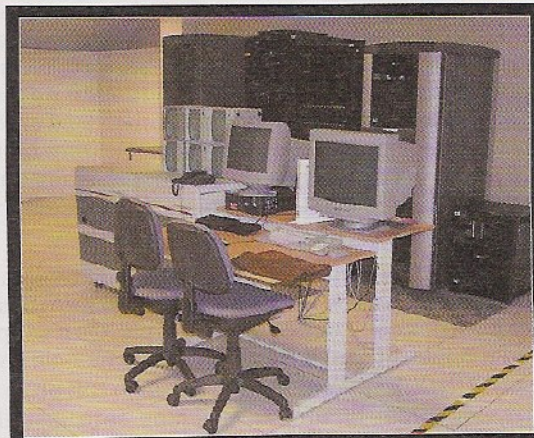


Figure 16 : Il faut bien tout cela pour contrôler le réseau interne et les archives

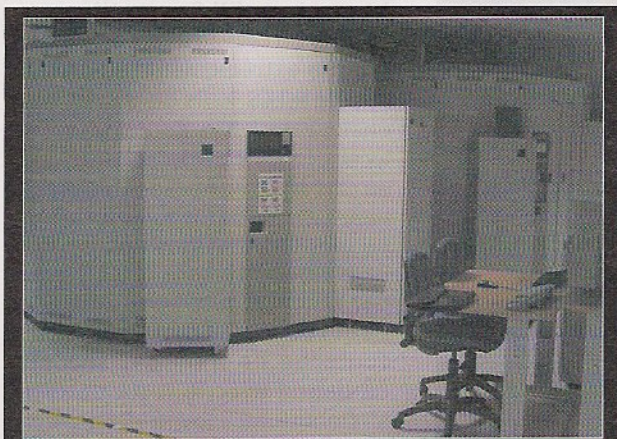


Figure 17 : Je connais des gens qui ont installé leur bureau dans d'anciens silos de stockage...

Comment nommer toutes ces machines ?

Le parc informatique du CEA comprend un nombre incroyable de machines, reliées par le protocole TCP/IP standard. Elles sont toutes accessibles (en interne) grâce à un DNS, qui oblige à faire très attention aux doublons.

La plupart des machines sont des stations de travail « classiques », baptisées par l'administrateur ou l'utilisateur. Par contre, cela devient beaucoup plus compliqué pour nommer un super-ordinateur, car il va être utilisé par tout le monde et tout le monde voudrait avoir son mot à dire. Et contrairement à l'IDRIS où le directeur choisit unilatéralement le nom, le CEA prend une décision collégiale, ce qui peut prendre des dimensions proportionnelles au nombre d'intervenants.

Traditionnellement, les machines d'un parc informatique suivent une thématique de nommage, souvent représentatives d'influences culturelles. Par exemple, à l'IDRIS, ce sont des noms issus des nouvelles de Jorge Luis Borges. Pour les machines du CCRT, ce sont des noms de métaux : Platine, Mercure, Tantale... Acier est un alliage, mais il devient nécessaire d'élargir la thématique quand tous les noms d'atomes sont déjà pris.

La question n'est pas très difficile lors de l'évolution d'une machine. Par exemple, le cluster des machines NEC est appelé Mercure, et les SX-8 reprennent le nom des SX-6 pour assurer la continuité du service. Cependant, lorsque les deux générations sont exploitées simultanément, les SX-6 sont renommées en « Mercure old », ce qui n'implique qu'une modification du DNS. Platine aussi a évolué : le cluster initial comprenant les premiers nœuds s'appelait « Argent ».

La future machine Bull n'a pas encore de nom, et le débat n'est pas clos. Aucun nom n'a obtenu de consensus et il y a des contraintes. Par exemple, Or n'est pas un nom valide, puisqu'il correspond à un mot-clé dans les shells UNIX. Espérons qu'une décision soit prise avant janvier 2009, lorsque la machine sera installée :-)

14

Comment utiliser un cluster ?

La différence majeure entre un ordinateur normal et un serveur de calcul intensif est que ce dernier n'est habituellement pas disponible en mode *interactif*. Un système de gestion de tâches (*jobs*) est installé sur un serveur frontal (pour les calculateurs vectoriels) ou dans un nœud d'administration d'un cluster. L'utilisateur travaille dessus (édition et compilation de son code source), puis ajoute son programme dans une file d'attente.

On peut n'utiliser qu'une partie des nœuds, mais les grands clusters imposent des unités d'allocation minimale (2 nœuds ou 16 processeurs au CCRT). Lorsque les ressources demandées (mémoire, durée, processeurs, etc.) sont disponibles, le programme est alors diffusé dans le réseau interne du cluster, puis exécuté. C'est le programme qui doit s'arranger pour que toutes ses instances puissent communiquer entre elles, souvent au moyen de l'interface logicielle standard MPI (*Message Passing Interface*).

Afin de garantir une bonne répartition des ressources, la durée d'exécution d'un job est limitée dans le temps (quitte à le reprendre ensuite). Par exemple, sur Platine, un job ne peut pas durer plus de 24h. De plus, les heures de CPU sont allouées par des comités scientifiques (pour les projets académiques) par blocs de centaines ou milliers d'heures, et, plus on utilise de processeurs simultanément, plus le quota expire vite. Une fois qu'un programme a le contrôle d'un processeur, il ne doit pas perdre un instant, car son quota de temps est décompté, quelle que soit l'activité demandée. Si le temps d'exécution dépasse la limite de temps de la file d'attente, le job est tué.

En fonction de l'activité et des demandes des utilisateurs, un job s'exécutera immédiatement ou devra attendre plusieurs jours... Un mécanisme de *fair share* (partage équitable des ressources) privilégie les utilisateurs qui ont le moins utilisé le cluster durant les 30 derniers jours. Mme Ménaché compare la situation à un jeu de Tetris, où les jobs seraient les briques qui tombent et l'espace de jeu serait le cluster : les situations ne sont ni prévisibles, ni reproductibles et le gestionnaire de jobs doit optimiser leur lancement. L'efficacité du système serait de 80% en moyenne.

Combien de temps durera l'exécution d'un programme ? C'est très difficile à prévoir par une simple analyse statique, comme Alan Turing l'a démontré il y a 60 ans. Le seul moyen de le savoir est de le faire tourner, mais il ne faut pas non plus gaspiller les précieuses heures de calcul ou monopoliser le cluster. Sur Platine, le développement et le *profiling* des programmes sont réalisés sur 256 processeurs dédiés, équipés d'une file d'attente limitant la durée des jobs à moins d'une heure. Grâce à un *turnover* plus rapide, les développeurs peuvent tester leurs programmes à petite échelle et augmenter progressivement le nombre de processeurs, pour vérifier la *scalabilité* de leurs algorithmes. À partir de ces informations, et si le code se comporte de manière linéaire, on peut facilement estimer la durée maximale d'exécution d'un job en fonction du nombre de processeurs.

Toutefois, les erreurs logicielles sont possibles et les pannes matérielles peuvent arriver. Le CCRT accueille des projets utilisant parfois des millions d'heures de CPU et un incident survenant à la fin de l'exécution du programme entraînerait de grosses pertes. Une sauvegarde, appelée *checkpoint*, est réalisée par les applications, toutes les heures par exemple. Tous les

processeurs vont alors écrire sur disque le contenu de leur mémoire, ce qui nécessite une bande passante énorme. Platine a été conçu justement pour fournir une vitesse

d'écriture et de lecture d'environ 20 Go par seconde, soit l'équivalent de 400 disques durs en parallèle, pour réduire au maximum la durée des checkpoints.

15

Le mur d'images tridimensionnelles

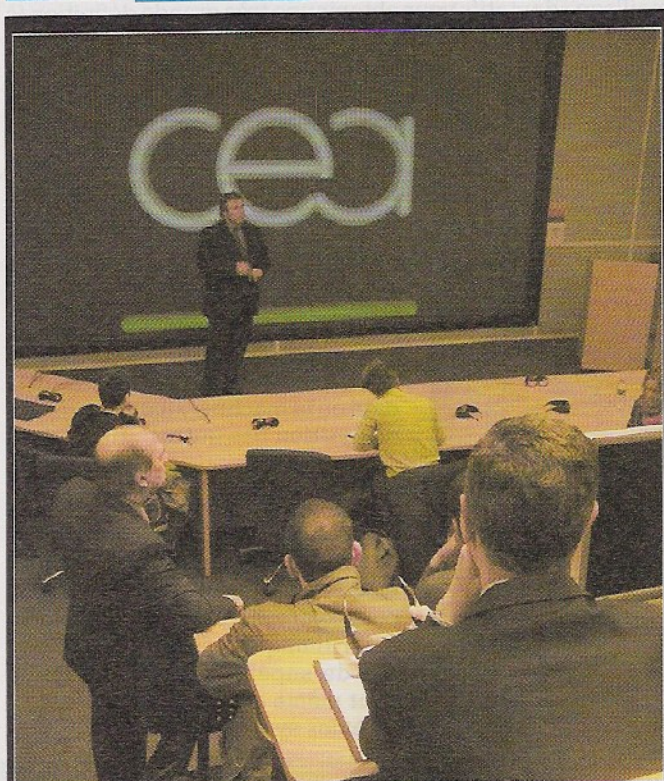


Figure 18 : Le mur d'images du CEA avant les présentations.

Après la visite de la salle des machines, le groupe se dirige dans la salle de projection. Le système *Mirage*, composé de 24 vidéoprojecteurs polarisés, y affiche en 3D des images générées par un ordinateur à 64 processeurs et 32 cartes graphiques. La surface (5,5m par 3m), la définition (13 M pixels) et le système optique créent des images d'une grande clarté, même au fond de la salle. Il faut cependant utiliser des lunettes spéciales pour profiter de l'effet stéréoscopique.

Ce système est un complément indispensable aux superclusters tout proches. Ces derniers génèrent des quantités de données gigantesques, qui sont inutiles si on ne peut pas les exploiter. L'affichage en 3D permet de mieux appréhender les phénomènes physiques simulés dans un volume. Pour illustrer cela, trois projets scientifiques sont présentés.

Le premier projet a consisté à simuler un tremblement de terre à proximité d'une ville située dans un bassin sédimentaire (semblable au frame de Mexico). Plusieurs phénomènes ont été étudiés simultanément dans un domaine de 11 km de côté : propagation des ondes dans les roches et dans les sédiments, éboulements, vibrations des bâtiments...

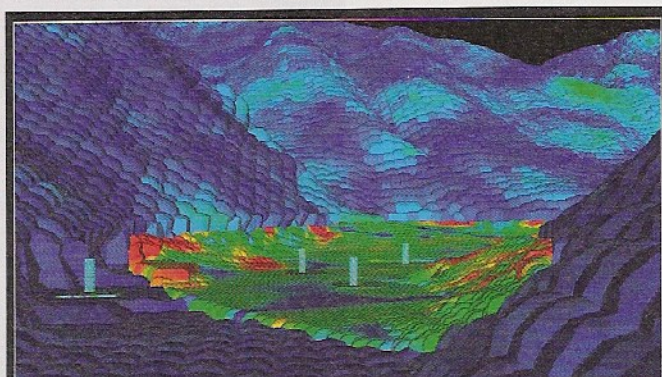


Figure 19 : Simulation d'un tremblement de terre (crédit : CEA)

Le deuxième projet a modélisé l'ensemble du corps d'un patient, ainsi qu'un appareil de TEP (Tomographie par Émission de Positons). L'objectif était de recréer l'image que le tomographe aurait fourni à partir des données du corps du patient, afin de mieux interpréter les images du tomographe réel.

Cette simulation a tourné sur Tera10 et est un excellent exemple des nouveaux types d'applications scientifiques. Les résultats n'ont pas été obtenus par résolution d'équations, mais par la méthode de Monte Carlo : des milliards de particules (des positons et des photons gamma), indépendantes, ont été simulées pour obtenir l'image finale. C'est donc une technique parfaitement adaptée aux ordinateurs massivement parallèles. Chaque particule étant influencée par le type du milieu physique traversé, il a fallu garder en mémoire la gigantesque description du corps et des tissus.

Dans le futur, quand les ordinateurs aussi puissants que Tera10 seront disponibles dans les hôpitaux, ils permettront d'améliorer les diagnostics et le dosage des produits traceurs radioactifs. En attendant, le calcul a nécessité l'ensemble des processeurs de l'ordinateur le plus puissant de France pendant trois heures.

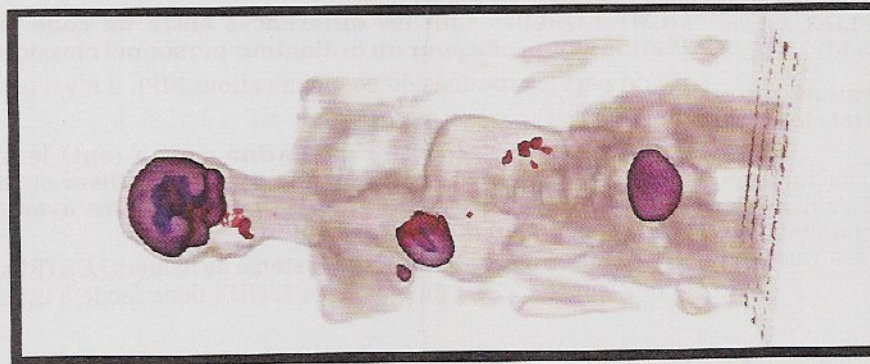


Figure 20 : Simulation d'une tomographie sur le corps numérisé d'un patient (crédit : CEA)

16 Le projet Horizon

Le troisième projet mis en avant lors du voyage de presse est le premier calcul à très grande échelle qui a tourné sur Platine. Cela a permis de valider l'architecture du système et sa réalisation, avec un calcul absolument énorme : s'il faut tester la machine, il vaut mieux que les calculs servent à quelque chose. Le CCRT a donc relevé le défi de simuler l'évolution de la moitié de l'univers observable, du Big Bang jusqu'à aujourd'hui [22] [23].

En pratique, 6144 cœurs de Platine ont été utilisés pour modéliser une grille tridimensionnelle gigantesque, comprenant 70 milliards de « particules ». Elles représentent du gaz qui s'assemble progressivement en étoiles, puis en galaxies, selon les règles et les paramètres déterminés par les astrophysiciens. Ce record mondial de complexité a été établi durant l'été 2007, lors de la phase de validation du cluster qui a précédé sa mise à la disposition des utilisateurs.

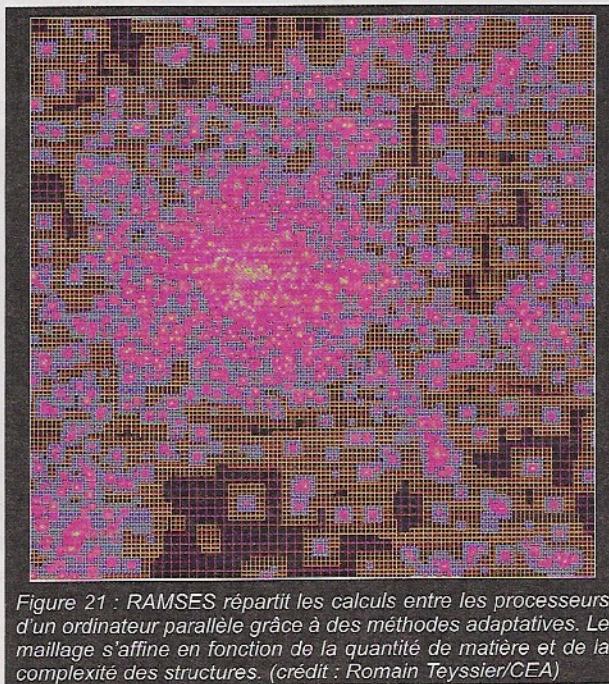


Figure 21 : RAMSES répartit les calculs entre les processeurs d'un ordinateur parallèle grâce à des méthodes adaptatives. Le maillage s'affine en fonction de la quantité de matière et de la complexité des structures. (crédit : Romain Teyssier/CEA)

Cette simulation a utilisé le code RAMSES [24] développé en Fortran90 par Romain Teyssier, et diffusé par le CEA sous licence CeCILL. C'est un code spécialement développé pour la cosmologie et il implémente des algorithmes parallèles de maillage adaptatifs pour obtenir une très grande définition (à l'échelle des étoiles) dans un très grand volume (l'Univers).

Il utilise le protocole MPI pour communiquer entre processeurs, et son efficacité a permis son utilisation simultanée sur 6144 processeurs.

Cela a quand même monopolisé Platine pendant deux mois, du vendredi midi au lundi midi. L'estimation de durée initiale (20 jours) a été dépassée à cause de l'incertitude initiale sur le code et quelques soucis de stabilité de la machine. Le calcul s'est heureusement terminé juste avant la date où la machine allait être mise à disposition des autres utilisateurs.

Une autre simulation similaire a lieu actuellement sur le supercalculateur Mare Nostrum, à Barcelone. Les données initiales sont identiques, mais l'algorithme utilisé est différent, pour comparer différentes approches algorithmiques. Cela crée en tout cas de superbes images [25], on peut même admirer une galerie de galaxies synthétiques...

Pour mieux comprendre dans quelles conditions on utilise un supercluster comme Platine, j'ai interrogé Romain Teyssier, du Service d'Astrophysique du CEA à Saclay.

GLMF : Comment le Projet Horizon est-il né et comment en est-il arrivé à pouvoir utiliser le cluster Platine ?

RT : Le Projet Horizon est né il y a 5 ans environ, lorsque les spécialistes de simulations numériques en cosmologie ont constaté le retard français dans le domaine. L'objectif était de fédérer les efforts pour pouvoir demander des moyens informatiques à l'étranger. Lorsque le CEA a acquis Platine, nous étions déjà en mesure de réaliser une simulation « dimensionnante », à savoir en utilisant plusieurs milliers de processeurs, grâce à notre expérience sur le calculateur espagnol MareNostrum. Après une première prise de contact, le CCRT a décidé de nous proposer ce grand challenge.

GLMF : Pouvez-vous décrire l'environnement de développement mis en place par le CCRT ?

RT : L'environnement de développement est du LINUX très standard, avec les compilateurs Intel et la librairie MPI BULL.

GLMF : Comment se déroule le développement d'un programme qui utilise toutes les ressources d'un tel cluster ? Des problèmes d'échelle sont-ils apparus ?

RT : Des problèmes sont effectivement apparus lorsque l'application a été déployée sur 6000 processeurs. Il s'agissait essentiellement de problèmes de communication qui nous ont conduit, en collaboration avec BULL, à réécrire les routines incriminées.

GLMF : Pouvez-vous expliquer ce qu'est le checkpointing, comment il fonctionne, et quelle est son influence sur le déroulement des calculs ?

RT : Le *checkpointing*, dans notre cas, consiste à écrire sur disque une copie quasi intégrale de la mémoire vive pour pouvoir, en cas de plantage intempestif, recommencer le calcul exactement à l'endroit où il s'était arrêté. Pour la simulation Horizon, ceci a demandé d'écrire 3 To de données toutes les heures environ. Grâce au système LUSTRE extrêmement performant installé au CCRT, cette opération prend 5 min environ. Le surcoût associé est donc de 5 min toutes les heures, soit 8%.

GLMF : Un bug sur une machine massivement parallèle est-il plus difficile à résoudre que sur un ordinateur classique ?

RT : Oui, beaucoup plus difficile, essentiellement à cause des communications MPI qui sont désynchronisées, ce qui conduit à des problèmes de chronologie dans le *debuggage*.

GLMF : Quelles sont les différences entre un code pour Platine et un code pour un ordinateur personnel classique ?

RT : À part les routines de communications MPI, il n'y a pas de différence.

GLMF : Quel(s) aspect(s) de Platine vous a (ont) le plus surpris ? Est-ce un environnement facile à utiliser ou bien y a-t-il de nombreuses contraintes à satisfaire avant de faire tourner un code efficacement ?

RT : La performance extrême du système de fichiers LUSTRE m'a le plus surpris. C'est du bon vieux LINUX donc facile à utiliser.

GLMF : Comment s'est déroulée la période de calcul durant l'été 2007 ? Est-ce un processus itératif ou bien le calcul a-t-il été effectué d'une traite ? Y a-t-il eu des incidents en cours de calcul, et de quels types ?

RT : La période de calcul a eu lieu pendant juillet et août 2007. À cause de problèmes de stabilité de la machine, il fallait surveiller le code en permanence, en réalisant des *shifts*, où chacun à tour de rôle assurait le baby-sitting du projet. Il fallait régulièrement rebooter le *cluster*, car l'application le mettait à rude épreuve.

GLMF : Combien de temps faut-il pour rebooter Platine ? Et comment cela se passe-t-il, comparé à un ordinateur classique ?

RT : Il faut environ 1/2 heure. Je ne connais pas les détails, mais ça doit sûrement être plus compliqué. Il faut aussi remettre en marche le *filesystem*.

GLMF : Quelles conclusions ou expériences tirez-vous de ce projet ?

RT : Une collaboration efficace entre informaticiens et scientifiques.

GLMF : D'autres projets de calcul à très grande échelle sont-ils prévus ?

RT : Oui, mais pas pour tout de suite : il faut d'abord exploiter ce calcul avant de se lancer dans d'autres projets.

GLMF : Il est difficile de traiter autant de données avec une station de travail. Comment exploitez-vous les 50 téraoctets de résultats générés ?

RT : Nous travaillons aussi sur Platine pour exploiter la simulation.

17 Linux met tout le monde d'accord

Après l'impressionnante séance de projection 3D haute résolution, tout le monde est convié à un petit buffet au sous-sol de l'État-major Ile de France de la Direction des Applications Militaires (rien que ça). Journalistes, responsables et même représentants de Bull sont rassemblés, le temps de déguster les préparations d'un traiteur. Mais, je suis encore dans l'excitation des présentations et j'en profite donc pour consolider mes informations en interrogeant deux ou trois personnes à la fois, oubliant ma faim et les tables qui se vident... Car, c'est une occasion unique et courte de parler avec de hauts responsables du CEA.

Après avoir obtenu des précisions sur comment fonctionne la soumission des *jobs* sur les *clusters*, la discussion tourne rapidement au *troll* sur les distributions Linux. Mais j'aurais dû me méfier du petit pingouin épinglé à la cravate d'un de mes interlocuteurs, car à *trolleur*, *trolleur* et demi. Le chef de laboratoire en face de moi utilise GNU/Linux au travail et chez lui. Pour certaines choses (administratives ou secrétariat), il faut encore utiliser MS Windows, mais les logiciels libres sont dominants dans les services informatiques du CEA. Cette culture ouverte et même passionnée tranche un peu avec l'atmosphère très pragmatique et raisonnée de l'IDRIS, sondé deux ans plus tôt.

Je croyais les responsables du CEA passionnés par Linux, mais je suis ensuite entré en contact avec un cadre de Bull... euphorique. C'est très étrange au début, surtout quand on se croit déjà un champion de la promotion des logiciels libres. Mais, on trouve toujours plus fort que soi et je ne m'attendais pas à ce qu'on puisse en faire son métier chez un constructeur français, dont on n'aurait d'ailleurs pas acheté une action il y a dix ans. J'ai déjà été étonné par l'annonce de la construction de Tera-10 par Bull, un industriel qui pour moi rimait avec systèmes informatiques pour l'administration (DPS8), cartes à puces pas toujours sûres (CP8), perfusions financières de l'État...

Mais, il semble que tout cela a changé depuis quelques années. Le turnover des employés semble comparable à celui d'autres entreprises (d'après leurs publications, « 30% du personnel présent aujourd'hui n'était pas dans l'entreprise il y a trois ans »), et la croissance est palpable, surtout dans les activités de services. Ce retour en grâce est-il coïncident avec une stratégie d'adoption de l'Open Source ?

M. Boris Auché s'est empressé d'éclairer ma lanterne. Dans le centre de services Bull à Paris, il assure la promotion des offres et produits Bull, qui sont majoritairement basés sur l'Open Source et Linux.

Un autre interlocuteur, d'un rang plus élevé, connaît un peu moins bien le fonctionnement d'un manchot ou d'un gnou, mais est tout aussi enthousiaste. Selon lui, le passage de l'environnement UNIX propriétaire de Compaq/HP (sur Tera) à l'environnement ouvert basé sur Red Hat Linux (sur Platine et Tera-10) a été « formidable ». Et tous mes interlocuteurs du CEA, de Bull et de NEC HPCE ont le même discours : GNU/Linux (même si le GNU n'est jamais mentionné) est choisi à chaque fois, non pas pour la liberté ou le prix, mais parce qu'il est le plus adapté et plus stable.

Évidemment, d'autres facteurs entrent en jeu et reviennent toujours à des conséquences de la liberté initiale. D'autres paramètres positifs participent aussi, comme l'abondance de jeunes diplômés qui ont utilisé Linux à l'école ou à l'université. Tous parlent le même langage dans un même environnement, ce qui évite la fragmentation des développements et permet l'utilisation des composants logiciels externes standardisés et hautement portables, sans oublier les bénéfices des mises à jour disponibles à un rythme soutenu. Les sociétés telles Red Hat et SuSE/Novell jouent aussi un rôle pérennisateur et décomplexant pour les « décideurs ».

GLMF : Quelle est la place de Bull dans l'univers open source, et quelle place prend l'Open Source dans les solutions Bull ?

BA : L'Open Source est partout. Côté intégrateur, tous nos projets sont réalisés, par défaut, en open source. Côté éditeur, Evidian intègre de l'Open Source (Bonita), Coriolis intègre et utilise largement les composants et produits open source. A contrario de nos concurrents, les activités de gestion des partenariats éditeurs sont marginales en regard des milliers de jours passés chaque année par nos ingénieurs sur l'Open Source.

GLMF : Sur quelles plateformes Bull propose-t-il ces environnements open source ?

BA : Toutes, NovaScale et Escala.

GLMF : Qu'est-ce qui a incité Bull à proposer des logiciels non propriétaires à ses clients ?

BA : Historiquement, nous sommes engagés, depuis très longtemps... C'est un pari gagnant ! De plus, la liberté que l'OSS apporte est dans nos gènes. D'ailleurs le slogan de Bull est « Architecte d'un monde ouvert » ! Et enfin, nous avons dû composer depuis toujours avec des concurrents

beaucoup plus gros et grands que nous, tout comme l'Open Source, nous étions donc faits pour l'open Source !

GLMF : Est-il difficile de proposer un environnement basé sur les logiciels libres ?

BA : Oui, car il faut savoir travailler seul, vendre la seule valeur ajoutée de Bull. Il n'y a pas de commerciaux, ni de budgets promotionnels fournis par les éditeurs. Quand on vend de l'OSS, la capacité à gagner l'affaire dépend du travail d'intégration.

GLMF : Est-ce que cela a permis à Bull de conquérir d'autres marchés ou à renforcer sa croissance ?

BA : Notre positionnement fort sur l'Open Source nous a permis de bien valoriser celui sur le marché de l'administration, et notre compétence reconnue nous permet aujourd'hui de remonter sur des marchés où nous étions peu présents. Quand ces marchés se mettent à regarder les apports de l'OSS, ils viennent naturellement nous voir. Par exemple, j'ai conduit des missions de conseil sur AVIVA, une compagnie d'assurance.

GLMF : Certains clients, comme le CEA, vous demandent-ils explicitement d'utiliser des logiciels libres ?

BA : Oui, nous avons des clients qui inscrivent l'OSS dans leurs cahiers des charges.

GLMF : Comment les logiciels open source cohabitent-ils avec des logiciels ou environnement propriétaires existants ou « legacy » ?

BA : Ils coexistent naturellement ! Novascale GCOS (système d'exploitation des mainframes Bull) tourne sur Bull Advanced Server, une distribution Linux de Bull.

GLMF : Y a-t-il parfois des conflits d'intérêt ou des contradictions à utiliser des logiciels libres dans un cadre commercial ou propriétaire ?

BA : Non, et d'ailleurs, y a-t-il un intérêt à vendre du propriétaire ? Il y a une vraie complémentarité, et vouloir faire du tout propriétaire ou du tout Open Source n'est pas possible. Dans la réalité, l'opposition entre les deux n'est pas bonne. C'est une bonne cohabitation qui fait que le système est stable et il n'y a pas de client qui bascule complètement du propriétaire au tout OSS. C'est pour cela que nous n'avons pas de position religieuse. Par contre, on fait parfois moins de marge, puisqu'on ne vend pas de licences.

GLMF : Des problèmes juridiques ont-ils été rencontrés ou identifiés ?

BA : Pas vraiment. Ceci dit, nos juristes sont tous aguerris aux licences OSS, pour éviter les conflits lors de développements applicatifs ou lors de nos activités éditeur. Des questions ont été remontées et tout le monde a été formé à ce sujet.

GLMF : Bull a-t-il diffusé en open source un ou des programmes initialement développés pour son usage interne ?

BA : Oui, Jonas, Bonita, Orchestra...

18

Conclusion

On peut voir encore aujourd'hui la coupole qui abritait le CDC6600 installé sur le campus de Jussieu dans les années 60. Depuis, la France a beaucoup dépendu des solutions conçues et construites aux USA, pour le calcul scientifique en général et les simulations atomiques en particulier, ce qui a parfois posé quelques soucis avec l'administration américaine. Les délais d'accès à la puissance de calcul se comptaient en années et les machines étaient exploitées jusqu'à l'usure comme le montre la période d'exploitation des ordinateurs CRAY du CEA.

Le monde a énormément changé depuis, la demande des utilisateurs scientifiques prenant le pas sur l'offre des constructeurs, chassés de leur tour d'ivoire. De nouvelles générations de machines arrivent encore tous les quatre ans environ, mais sont mises en service quasiment en flux tendu. Les performances deviennent astronomiques, mais les budgets ont du mal à suivre la demande des scientifiques.

Consciente de sa dépendance technologique et de s'être laissée distancer par les USA et l'Asie, l'administration française est résolue à passer à la vitesse supérieure, comme en témoignent les récents remembrements des centres existants ou la création de nouvelles structures. Car, ce n'est pas un ordinateur gigantesque qui résoudra tous les problèmes, puisque celui-ci sera forcément dépassé (puis démonté) un jour ou l'autre. Il faut un ensemble cohérent d'acteurs (constructeurs, exploitants et utilisateurs) pour que les scientifiques puissent exploiter rapidement, en continu et efficacement les dernières technologies. Nous verrons dans quelques années les résultats des projets annoncés en avril.

Encore une fois, et comme pour l'article sur Uqbar, j'ai été pris par surprise tout au long de cette enquête. Je voulais juste découvrir l'ordinateur civil le plus puissant de France et j'ai eu droit à bien plus ! Pourtant, le centre de calcul

de Bruyères-le-Châtel n'est pas vraiment hors norme. Il ressemble même beaucoup à d'autres salles de machines existantes. Ses particularités sont la mixité des applications (scientifiques et industrielles dans un environnement orienté « défense »), l'échelle des machines (Platine ressemble à d'autres clusters en armoire, il y a juste plus d'armoires) et l'environnement tourné vers l'Open Source (afin d'uniformiser le développement et de ne plus être dépendant des solutions propriétaires). Mais l'environnement humain, le rôle politique et surtout les gardes à l'entrée du site vous rappellent que ces machines servent à faire avancer la Science, pas à jouer à Doom.

Tout cela conduit à la question suivante : maintenant que le défi initial a été relevé, que reste-t-il ? Quel système à la pointe de la performance (français ou non) vaut-il la peine d'être visité pour faire l'objet d'un reportage ? Je crois qu'il va falloir attendre que le nouveau cluster du CCRT soit mis en place, à moins que l'IDRIS ne réserve encore d'autres surprises... Justement, on y augmente actuellement la puissance de 200 TFLOPS en installant de nouvelles machines IBM.

Remerciements

Cet article a bénéficié de l'aide (de près ou de loin) de très nombreuses personnes, toutes passionnées, à commencer évidemment par le personnel du CEA qui a organisé le voyage de presse et eu la patience de répondre à mes nombreuses questions. Des employés des constructeurs BULL et NEC HPCE ont aussi répondu présent et m'ont permis de solidifier cet article, car certaines informations disponibles en ligne sont incohérentes ou difficiles à vérifier. Il ne faut pas non plus oublier Diamond Editions qui m'a soutenu, et Laura qui m'a assisté !

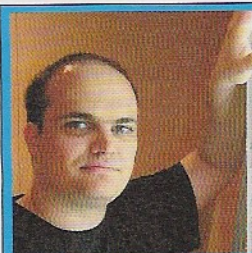
À tous, je vous dis « merci ! »

Liens

Internet regorge d'informations sur le CEA, le *High Performance Computing* (HPC) et les calculateurs scientifiques. Je vous invite à explorer les liens suivants, en faisant toutefois attention à la cohérence et l'exactitude des informations. Encore merci à toutes les personnes qui m'ont aidé à faire le tri !

- [1] http://www.thocp.net/hardware/nec_ess.htm
Petite présentation d'Earth Simulator
- [2] <http://www.nec.co.jp/press/en/0805/1601.html>
Annonce de l'augmentation de la puissance d'Earth Simulator
- [3] http://www.cea.fr/content/download/3409/16810/file/Tera10_janvier2006.pdf
Dossier de presse de la visite de Tera10 en janvier 2006
- [4] <http://fr.wikipedia.org/wiki/TERA-10>
(attention, certaines informations sont erronées)
- [5] http://www.guideinformatique.com/fiche-tera_10_lordinateur_le_plus_puissant_deurope-771.htm
Reportage sur les installations de Tera10
- [6] <http://www.idris.fr/IBM2008/Revue-de-presse/LeParisienEssonne-080108.jpg>
Photographie de la visite du Président Jacques Chirac (qui a soutenu le programme Simulation) sur le site de Tera10 en septembre 2006
- [7] <http://www2.english.uiuc.edu/cybercinema/forbin.htm>
- [8] <http://www-ccrt.cea.fr/fr/partenaires/index.htm>
Les partenaires industriels du CCRT : STMicro, EDF, les sociétés du groupe SAFRAN (SNECMA, Turbomeca, Techspace Aero), EADS...
- [9] <http://www.genci.fr/spip.php?article2>
Heures de calcul 2008, allouées pour GENCI le jeudi 7 février 2008
- [10] <http://www.idris.fr/info/entreprises.html>
« La direction de l'IDRIS dispose de la capacité d'engager environ 10 pour cent des ressources informatiques de l'Unité, dans des opérations de collaboration avec les entreprises. »
- [11] http://www.cea.fr/cea/content/download/548/3303/file/cea_dam_idf_map.pdf
Plan d'accès au CCRT par la route.
- [12] http://www-ccrt.cea.fr/fr/moyen_de_calcul/index.htm
Présentation des ressources informatiques du CCRT
- [13] http://www.cea.fr/content/download/5353/34954/file/CEA_CCRT.pdf
Dossier de presse de la visite du CCRT d'avril 2008

- [14] <http://www.genci.fr/IMG/pdf/GENCI-CEA-BULL-Avr08-fr.pdf>
<http://www.genci.fr/spip.php?article23>
GENCI et le CEA passent commande d'un Supercalculateur Bull destiné au CCRT
- [15] [http://en.wikipedia.org/wiki/Montecito_\(processor\)](http://en.wikipedia.org/wiki/Montecito_(processor))
Les Itanium2 comportent deux *cores*, chaque core supporte 2 *threads* et 12 MO de mémoire cache, pour 1,7 milliard de transistors. Le FSB travaille sur 256 bits à 533 MHz soit 17 GO/s de bande passante.
- [16] <http://www.idris.fr/comp/vecto/index-brodie.html>
Configuration de Brodie, le SX-8R de l'IDRIS
- [17] http://www-ccrt.cea.fr/fr/moyen_de_calcul/mercure.htm
Description de Mercure, le SX-8R du CCRT (un peu incohérent)
- [18] http://en.wikipedia.org/wiki/SX_architecture
Tableau comparatif des différentes générations de calculateurs SX
- [19] <http://code.google.com/p/sx-gcc/>
Un début de commencement d'effort pour porter GCC sur l'architecture SX...
- [20] <http://www.nec.de/hpc/hardware/scalar-smp/index.html>
Description des serveurs TX-7, qui servent d'ordinateurs frontaux pour les calculateurs vectoriels
- [21] http://www-ccrt.cea.fr/fr/moyen_de_calcul/tantale.htm
Description du cluster HP du CCRT
- [22] http://defis.cea.fr/defis/130/cea_defis130_04_10.pdf
Présentation du projet Horizon dans la revue du CEA
- [23] <http://www.projet-horizon.fr/>
La page d'accueil du projet Horizon
- [24] http://irfu.cea.fr/Phoce/Vie_des_labos/Ast/ast_sstechnique.php?id_ast=904
Le code RAMSES est téléchargeable sous licence CeCILL
- [25] <http://www.projet-horizon.fr/rubrique38.html>
Galerie d'images générées par le Projet Horizon



Yann Guidon

Électronique, musique et
informatique en folie^Wliberté
<http://ygdes.com>

Actualité : découvrez les

Nagios, comme tout bon programme, se bonifie au fil du temps en écoutant attentivement les attentes de ses utilisateurs. Développé comme un modeste programme de supervision du nom de Netsaint, il change pour devenir petit à petit la pierre angulaire des architectures de supervision dans le monde de l'Open Source.

1

Une évolution constante

1.1

Les ajouts de Nagios 2

Nagios 2 sorti mi-2006 apporte de nombreuses fonctionnalités à son prédécesseur. On note, outre l'amélioration des performances et un ajout massif de macros (variables suivant le contexte d'évaluation de la commande), l'arrivée des *services groups*. Ceux-ci permettent d'agréger des résultats de contrôle de services en une seule entité. Celle-ci pourra servir à alléger les écrans de supervision en ne présentant par exemple qu'un seul élément pour un ensemble de clusters de service Web. Elle peut être également utilisée afin de définir des dépendances plus simplement pour les environnements complexes (comme un service Web dépendant d'un cluster de base de données).

Nagios 2 est aussi l'occasion pour l'auteur de ne plus supporter, au sein de l'application, la gestion de configuration de l'outil sur des bases de données (MySQL et PostgreSQL). En effet, cette gestion était hasardeuse de l'aveu même de l'auteur. Fidèle à la notion de modularité qui a fait le succès de Nagios, il implémente au cœur de l'application

des possibilités pour le chargement de modules ayant accès aux données internes de Nagios. Cette notion a pour nom «event broker». Se développant doucement au long de la vie de Nagios 2, cette notion va prendre pleinement son sens au sein de Nagios 3. Nous aurons l'occasion d'en reparler par la suite.

1.2

Un avant-goût de Nagios 3

Tout en restant dans la continuité de ses prédécesseurs, Nagios 3 apporte principalement des solutions à la gestion des grands environnements (plus de 10000 services).

En effet, certaines limites conceptuelles de Nagios 2 rendaient la montée en charge difficile. Ces difficultés se ressentaient sur des délais d'ordonnancement non respectés et, surtout, par une configuration qui pouvait devenir une véritable épreuve de force avec l'outil lorsque le nombre d'éléments supervisés et d'utilisateurs augmentaient. Loin d'être linéaire en nombre de machines et d'utilisateurs, cette complexité avait un goût de problème NP complet...

2

Les avancées de Nagios 3 en termes de performance

En plus des habituelles améliorations des structures internes, cette nouvelle version nous réserve bien des surprises permettant d'alléger la charge de nos pauvres serveurs.

2.1

Problème de conception

Nagios 2 avait un problème de conception en ce qui concerne la vérification des hosts. En effet, même si la vérification des services était parallélisée, ce n'était pas le cas de celle des hosts. Il ne fallait pas non plus qu'il y ait une vérification de service au même moment ! Si Nagios souhaitait vérifier une machine, il devait arrêter d'alimenter le flux des vérifications de services, attendre que les vérifications en cours se terminent pour enfin lancer la vérification de l'host.

Cet état de fait obligeait les utilisateurs de Nagios à ne pas planifier de vérification de machines automatiquement afin de garder des performances acceptables. Nagios ne

planifiait le contrôle d'une machine que lorsqu'un service avait un problème. Ceci permettait de limiter les pertes de performances, mais lorsqu'un grand nombre de hosts avaient des problèmes, ces vérifications pouvaient créer une grande latence de traitement des vérifications normales. Dans cette nouvelle version, ces limitations ne sont plus de mise. Les vérifications d'hosts sont parallélisées entre elles et celles des services. Ceci permet de traiter un bien plus grand nombre de machines avec une seule instance de Nagios.

2.2

Problème de vérification de configuration

Le lancement de Nagios pouvait poser problème lorsqu'on avait un grand nombre de relations parents/enfants au sein de ses hosts. En effet, afin d'éviter de tourner en rond lors de son exécution, Nagios vérifie qu'il n'y a pas de cycle dans ces relations. L'algorithme utilisé n'étant pas vraiment

Supervision avancées de Nagios 3

optimal [1], cette vérification prend un temps très important (jusqu'à plusieurs dizaines de secondes à chaque lancement pour de grands environnements). Le traitement des fichiers de configuration est également très long. Les vérifications étant bien sûr bloquées pendant ce temps.

L'auteur a ainsi proposé un système de cache permettant de faire ces vérifications une seule fois lorsque l'application tourne encore. Il vérifie et sauvegarde les éléments configurés dans un format plus rapide à traiter que les fichiers textes. Aux prochains lancements de l'application, elle utilisera ce résultat en tant que source de configuration, et ce, quasi instantanément. Qui dit système de cache, dit bien sûr gestion de la cohérence du cache. Ici, il faut tout simplement que l'utilisateur n'oublie pas de lancer une vérification/génération de cache préalablement au lancement de l'application sous peine de ne pas voir ses dernières modifications prises en compte.

Un exemple de lancement devient donc :

L'argument **p** étant pour la génération du cache, le **v** est classiquement pour le lancement de la vérification de la configuration. L'argument **d** est le mode *daemon* classique, le **u** permet de charger le cache et enfin le **x** permet de passer outre la vérification des cycles.

```
#génération du cache
/usr/local/nagios/bin/nagios -pv /usr/local/nagios/
etc/nagios.cfg
#Lancement de l'application
/usr/local/nagios/bin/nagios -uxd /usr/local/nagios/etc/
nagios.cfg
```

2.3

Pour les installations load balancées

Il est courant de voir, sur les très gros environnements, plusieurs instances de Nagios travailler de concert : une armée de Nagios distribués un peu partout sur le réseau effectue les vérifications qu'elle remonte ensuite à un Nagios central. Cette transmission se fait selon le mode passif classique de Nagios : un daemon (NSCA) écoute et est chargé de transmettre ces informations au daemon Nagios. Il utilise pour cela le mécanisme de commandes externes envoyées dans un *pipe* nommé. Nagios « consomme » ces commandes en les plaçant en file d'attente au fur et à mesure. Si cette file d'attente est pleine, le pipe nommé n'est plus consommé et commence à se remplir petit à petit...

Avec la nouvelle version, la taille du *buffer* interne de commandes en attente est configurable et les utilisateurs ayant une grande installation peuvent mettre une valeur suffisamment grande pour ne pas avoir de problème de blocage de données.

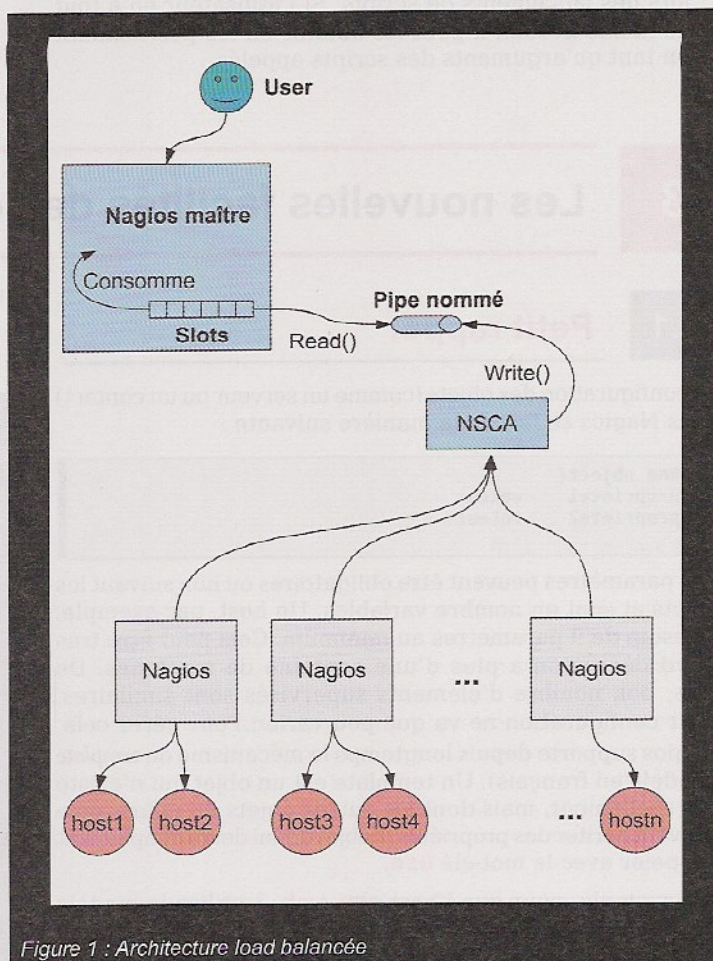


Figure 1 : Architecture load balancée

2.4

Mise en cache des résultats

Une nouvelle possibilité qui va intéresser fortement les personnes ayant de nombreuses relations de dépendances de machines ou de services est la possibilité, lors d'un lancement de vérification non prévue, (afin de vérifier l'état de l'objet dont on dépend) de réutiliser un ancien résultat. Bien sûr, si cet état est trop vieux, il peut être faux. Deux variables font alors leur apparition **cached_host_check_horizon** et **cached_service_check_horizon**. Elles sont respectivement l'âge maximum en secondes des états des services ou des hosts que Nagios s'autorise à utiliser. La valeur par défaut est de 15 secondes. L'utilisateur pourra fixer cette valeur en fonction de ses besoins de performances et de pertinence d'information.

2.5

Si tout ceci ne suffit pas

Nagios 3 propose une option l'autorisant à prendre certains raccourcis dans son exécution, afin d'alléger sa charge. Ce paramètre est **use_large_installation_tweaks**. Une fois activé, Nagios se permet de :

- Ne pas positionner en variable d'environnement certaines macros, coûteuses en termes de performance et peu utilisées, lors des lancements de scripts. Si l'utilisateur en a tout de même besoin, il peut les obtenir en les positionnant en tant qu'arguments des scripts appelés.

- Ne pas nettoyer l'espace mémoire des processus enfants effectuant les vérifications. Il laisse ce soin au système.
- Ne pas *double forker* avant de lancer un processus enfant. En temps normal, Nagios utilise ce moyen pour se protéger des possibles problèmes de programmation des plugins. En tant que ressource manquante, ce risque est acceptable.

3

Les nouvelles facilités de configuration

3.1

Petit rappel

La configuration des objets (comme un serveur ou un contact) dans Nagios se fait de la manière suivante :

```
define object{
    propriete1    valeur
    propriete2    valeur
}
```

Ces paramètres peuvent être obligatoires ou non suivant les objets et sont en nombre variables. Un host, par exemple, a besoin de 9 paramètres au minimum. Ceci peut être très lourd dès qu'on a plus d'une centaine de machines. De plus, bon nombre d'éléments supervisés sont similaires. Leur configuration ne va que peu varier. Pour gérer cela, Nagios supporte depuis longtemps le mécanisme de *template* (modèle en français). Un template est un objet qui n'existe pas réellement, mais dont les autres objets du même type peuvent hériter des propriétés. L'objet défini devra simplement l'appeler avec le mot-clé **use**.

Par exemple, pour que l'host **srv-web-1** utilise le modèle **serveur-linux**, il suffira de déclarer :

```
define host{
    use      serveur-linux
    host_name    srv-web-1
    alias    srv-web-1
    address  172.16.0.1
}
```

Ce mécanisme permet de diminuer la duplication de propriétés pour des objets se ressemblant.

3.2

Héritage multiple dans Nagios 3

Là où Nagios 2 s'arrêtait à un héritage simple des templates, Nagios 3 permet d'en utiliser plusieurs en les séparant par une virgule. Ceci permet de définir des schémas complexes d'héritages en autorisant encore plus la réutilisation de paramètres déjà positionnés et d'avoir au final moins de templates.

3.2.1

Exemple

Prenons le cas d'un serveur Linux de développement. Les serveurs Linux vont avoir en commun les méthodes de vérification (une vérification du port SSH par exemple). Les serveurs de développement vont être caractérisés principalement par des temps de vérification allongés, car ils sont beaucoup moins importants que la production.

On va donc définir deux templates :

- **generic-linux**
- **development-server**

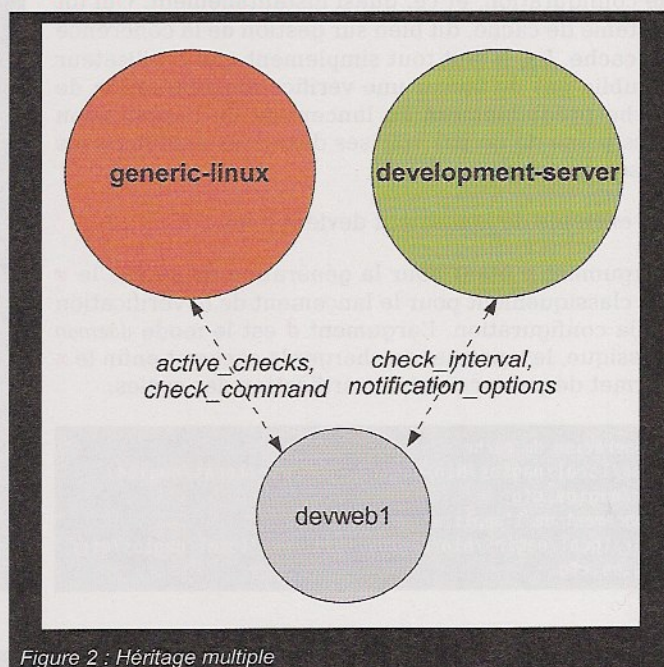


Figure 2 : Héritage multiple

La configuration est donc :

```
# Generic host template
define host{
    name      generic-linux
    active_checks_enabled 1
    check_command    check-ssh
    ...
    register    0
}

# Development web server template
define host{
    name      development-server
    check_interval  60
    notification_options d,u,r
    ...
    register    0
}

# Development web server
define host{
    use      development-server, generic-linux
    host_name    devweb1
    ...
}
```

Dans cet exemple, on arrive à une définition de **devweb1** équivalente à :


```
# Development web server
define host{
    host_name devweb1
    active_checks_enabled 1
    check_command check-ssh
    check_interval 60
    notification_options d,u,r
    ...
}
```

Ceci tout en limitant encore davantage que pour Nagios 2 la configuration nécessaire.

3.3

Rajout au lieu d'écrasement de valeur

Dans l'héritage par template classique, si on redéfinit une propriété, celle du template n'est pas prise en compte. Mais que faire si l'on veut avoir les deux ? Dans les versions précédentes de Nagios, on devait déclarer un template intermédiaire avec les bonnes propriétés, ce qui pouvait se révéler assez lourd à gérer. Un mécanisme a été mis en place pour gérer ce genre de cas : l'héritage additif. Si l'on veut ajouter sa valeur à celle du template, il suffit de lui ajouter le signe + devant. Ceci ne fonctionne bien sûr qu'avec les propriétés de type **string**.

Prenons par exemple l'ajout de toutes les machines dans un groupe **all-servers**. Il fallait dans Nagios 2 définir ce **host group** dans tous les templates et, en cas de changement de nom du groupe, il fallait modifier tous les templates ce qui pouvait prendre du temps... Avec le nouveau mécanisme, ceci se fait beaucoup plus simplement :

```
define host{
    hostgroups all-servers
    name generic_host_template
    register 0
}

define host{
    host_name linux_web_servers
    hostgroups +linux-servers,web-servers
    use generic_host_template
    register 0
}
```

Dans cette configuration, le host group **all-servers** va être ajouté aux host groups de **linux_web_servers**. Ceci va au final être équivalent à :

```
define host{
    host_name linux_web_servers
    hostgroups all-servers,linux-servers,web-servers
    use generic_host_template
    register 0
}
```

Avec plus de souplesse de gestion par la suite. En cas de modification du groupe global, il suffira de changer le nom dans la première déclaration pour qu'elle soit prise en compte sur tous ses descendants.

3.4

Problème des multiples définitions

Cette possibilité, ajoutée au fait qu'un template peut hériter ses propriétés d'un autre, offre la possibilité d'avoir une hiérarchie complète et complexe d'héritage. Cependant, il

faut bien observer ce qui se passe lorsqu'une propriété est définie dans plusieurs templates dont on hérite :

- Dans le cas d'un héritage vertical, c'est la dernière propriété définie qui est prise en compte ;
- Dans le cas d'héritages multiples, on doit faire attention à l'ordre de déclaration des templates parents. En effet, une fois rencontrée dans un parent, la propriété ne sera plus surchargée par les autres parents.

Dans l'exemple du serveur web de développement, si le template **development-server** avait proposé un autre type de commande de contrôle (comme un **check-icmp** par exemple), c'est elle qui serait prise en compte au lieu du **check-ssh** de **generic-linux**.

Le diagramme suivant présente l'ordre d'héritage lorsqu'on mélange les différents héritages par template :

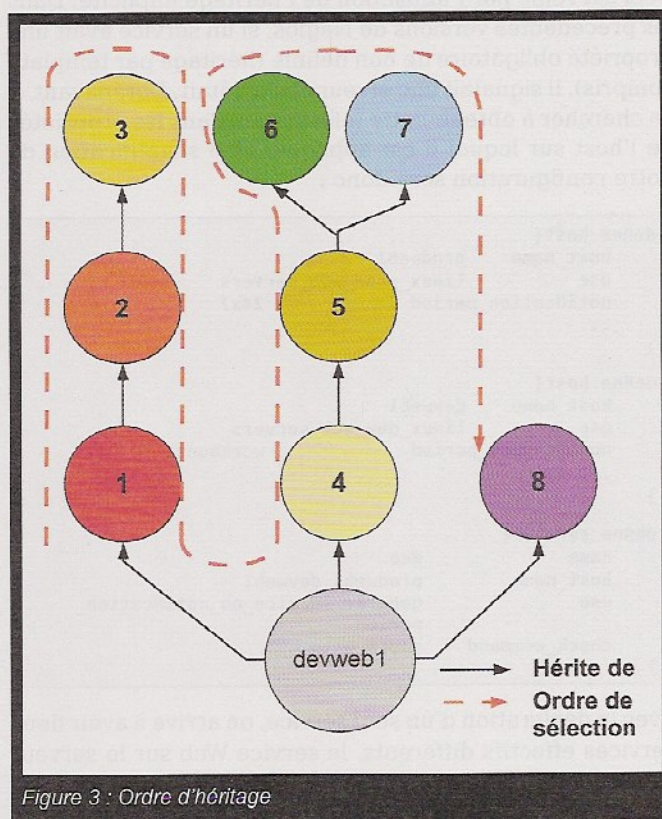


Figure 3 : Ordre d'héritage

3.5

Héritage implicite

On peut considérer le système de template comme un héritage d'un même type (un service hérite d'un autre service par exemple). On remarque cependant que certaines propriétés sont communes pour des types différents comme les **contact_groups** ou **notification_period** entre host et service. Dans une majorité de cas, ces valeurs seront les mêmes entre le service et l'host sur lequel il est appliqué. On duplique ainsi l'information, ce qui n'est pas optimal. Un nouveau mécanisme d'héritage a été introduit pour gérer ce genre de cas : l'héritage implicite.

Reprenons l'exemple de notre serveur web de développement. N'étant pas en production, il nous importe peu que la machine ou le service web ne soit pas accessible la nuit. On va donc positionner la propriété **notification_period** sur la machine à **workhours** et on va dupliquer cette information dans le service Web accroché à cette machine. Les serveurs de production auront eux une valeur de 24x7 pour la **notification_period** tant au point de vue de l'host que du service Web. Avec l'héritage classique, pour gérer la vérification du service Web sur ces différents environnements, il aurait fallu déclarer deux services (un pour les environnements de production et un autre pour les environnements de développement), alors que la seule propriété qui change est la période de notification, information déjà présente dans l'host. C'est dommage de devoir la spécifier à nouveau.

Ceci est réglé par l'utilisation de l'héritage implicite. Dans les précédentes versions de Nagios, si un service avait une propriété obligatoire de non définie (héritage par template compris), il signalait une erreur et s'arrêtait. Dorénavant, il va chercher à obtenir cette information dans les propriétés de l'host sur lequel il est appliqué. Une simplification de notre configuration sera donc :

```
define host{
    host_name    prodweb1
    use          linux_prod_web_servers
    notification_period    24x7
    ...
}

define host{
    host_name    devweb1
    use          linux_dev_web_servers
    notification_period    workhours
    ...
}

define service{
    name                Web
    host_name            prodweb1,devweb1
    use                  generic_service_no_notification_
                        period
    check_command        check_http
}
```

Avec la déclaration d'un seul service, on arrive à avoir deux services effectifs différents, le service Web sur le serveur de développement nous laissant tranquille la nuit... Il est à noter qu'ici, pour des raisons de lisibilité, la propriété **notification_period** a été placée sur les hosts, mais il est fortement recommandé de la placer dans les templates **linux_prod_web_servers** et **linux_dev_web_servers**. Ceci aurait également fonctionné si l'on avait appliqué le service sur un groupe de machines plutôt que directement sur les deux hosts. La granularité du mécanisme reste au niveau de la machine.

Cet héritage implicite fonctionne également avec les propriétés **contact_groups** et **notification_interval**. D'expérience, ce mécanisme permet de diviser par 2 ou 3 le nombre de services déclarés et permet une plus grande facilité de configuration, car, si l'on modifie l'host, tous ses services utilisant ce mécanisme vont être mis à jour automatiquement.

3.6 Héritage implicite et additif

Il est possible d'utiliser conjointement tous les types d'héritages dont notamment l'implicite et l'additif. Si l'on définit une propriété avec l'héritage additif sur une valeur non définie dans un template, on appliquera tout d'abord l'héritage implicite (récupération de la valeur dans l'host par exemple) à laquelle on ajoute la valeur que l'on vient de donner.

L'exemple ici sera notre bon vieux service Web sur nos machines de développement et de production. Ces machines sont administrées au niveau système par deux équipes différentes, mais un seul administrateur Apache s'occupe de tous les services Web. Les administrateurs de la production ne veulent pas entendre parler des serveurs de développement et réciproquement.

Si l'on veut dans ce cas ne déclarer qu'un seul service qui prévienne tout ce bon monde, il faut ajouter l'administrateur Apache dans les deux groupes système et utiliser l'héritage implicite sur ce service. Mais alors, il va aussi recevoir les erreurs système dont il se moque éperdument. Faut-il donc se résigner à faire deux services et ne pas profiter de l'héritage implicite ?

Non, on va l'agrémenter d'héritage additif : le contact **Admin_Web** va être ajouté dans le service précédé d'un signe +.

Ceci nous donne donc une déclaration de service :

```
define service{
    name                Web
    host_name            prodweb1,devweb1
    use                  generic_service_no_notification_period
    check_command        check_http
    contact_groups       +admin_web
}
```

Sur la machine de développement, ceci va être équivalent à la déclaration suivante :

```
define service{
    name                Web
    host_name            devweb1
    use                  generic_service_no_notification_period
    check_command        check_http
    contact_groups       admin_web,admin_dev
}
```

Alors que sur la machine de production ceci va donner :

```
define service{
    name                Web
    host_name            prodweb1
    use                  generic_service_no_notification_period
    check_command        check_http
    contact_groups       admin_web,admin_prod
}
```

Tout ceci en une seule déclaration et en gardant une grande souplesse de configuration. Merci Nagios !

3.7 Bilan de la configuration

Les mécanismes d'héritage dans les précédentes versions de Nagios se limitaient à l'héritage simple avec les templates. Bien que simplifiant la configuration, ils se révélaient insuffisants dans la gestion des grands environnements. Les nouvelles possibilités permettent de simplifier d'une

manière drastique la configuration de Nagios en diminuant le nombre d'éléments déclarés. Ceci en ayant beaucoup moins de duplication d'information. Les personnes recevant les alertes vont donc être plutôt contentes, car l'administrateur Nagios va beaucoup moins se tromper en oubliant ces personnes dans le maquis de ses templates.

Vous n'avez plus de raison de ne pas revoir votre configuration et de la simplifier au maximum. C'est un grand travail de *refactoring*, mais au combien agréable lorsqu'il est fini !)

4

Un aperçu (rapide) de l'avenir

Comme dit précédemment, Nagios a bien évolué et continue à le faire. Après un grand travail sur la configuration, une nouvelle ère arrive avec les prémices de l'utilisation de l'événement broker. La première utilisation à grande échelle de ce système est le support d'un *plugin* (NDO) exportant les informations de Nagios dans une base de données (MySQL ou PostgreSQL). Ces informations peuvent être par exemple l'état des machines et des services. Il n'est donc plus obligatoire de traiter le fichier d'état de Nagios afin d'obtenir ces informations. Les outils tiers en tirent une plus grande facilité d'accès et une grande amélioration de leurs performances. Une des principales interfaces de visualisation utilisant ce système est actuellement Nagvis [3] qui ne supporte d'ailleurs plus depuis ses dernières versions la méthode par *parcours* de fichiers. D'autres outils comme Centreon [4] commencent à supporter ces possibilités.

De plus, ces informations peuvent provenir de plusieurs instances de Nagios (chacune ayant un id unique). On peut ainsi avoir un unique point de convergence des états sans avoir besoin d'avoir un Nagios maître comme c'est le cas actuellement dans les architectures distribuées. Ces nouvelles fonctionnalités vont offrir une nouvelle manière pour accéder aux informations de Nagios sans pour autant complexifier à outrance son architecture. Gageons que l'auteur et la communauté sauront garder ce qui a fait le succès de Nagios au fil des années : sa simplicité et sa modularité.



Jean Gabès

Jean Gabès, ingénieur système pour une société développant des solutions technologiques intégrées dans la région de Bordeaux.

Références

- [1] Nagios,
<http://www.nagios.org>
- [2] Mailing list Nagios-Devel,
<http://news.gmane.org/gmane.network.nagios.devel>
- [3] Nagvis,
<http://www.nagvis.org>
- [4] Centreon,
<http://www.centreon.com>

Vous les avez manqués en kiosque ?



N° 35

N° 36

HORS-SÉRIE

Les hors-série 100% pratique pour gérer votre serveur WEB/MAIL



Ils sont toujours disponibles sur www.ed-diamond.com

Introduction à la supervision

De nos jours, dans le monde du Libre, beaucoup de personnes assimilent le mot supervision directement à Nagios. Certes, Nagios est sûrement l'outil de supervision le plus utilisé et le plus abouti du monde libre, mais il en existe beaucoup d'autres qui gagnent à être connus. C'est le cas de Zabbix [1], un superviseur créé en 2001 par la société qui porte le même nom et qui n'a pas grand-chose à envier à son grand frère comme va le démontrer cet article d'introduction.

1 Introduction

1.1 Qu'est-ce que la supervision ?

Phénomène très à la mode depuis maintenant quelques années, la supervision consiste à surveiller le bon fonctionnement d'un parc informatique. Elle se fait à travers divers moyens qui sont :

- supervision SNMP (*Simple Network Management Protocol*)

Le superviseur envoie des requêtes SNMP à des équipements pour récolter des informations sur leurs états. Les équipements peuvent également envoyer des *traps* SNMP au superviseur lorsque certains événements se produisent.

- supervision via l'analyse de journaux

Le superviseur analyse en temps réel les fichiers de *logs* pour obtenir des informations sur ce qui se passe sur une machine particulière et/ou sur le réseau.

- supervision active au travers d'actions

Le superviseur s'occupe d'aller chercher sur le réseau et sur les machines à superviser les informations dont il a besoin (charge CPU, processus,...).

Nous allons voir que Zabbix permet en natif d'intervenir sur ces trois moyens de supervision afin de surveiller efficacement tout ce que l'on désire...

1.2 Zabbix en quelques mots

En quelques mots, les deux fonctions principales de Zabbix sont de suivre l'évolution d'un parc informatique d'un point de vue statistique avec des graphiques générés par RRDtool et disponibles en temps réel depuis une interface web et de déclencher des alertes lorsqu'un seuil paramétrable est dépassé. Par exemple, un administrateur peut paramétrer Zabbix pour qu'il lui envoie un mail dès qu'une machine ne répond plus aux *pings* depuis un certain temps.

Dans cet article, nous allons commencer par expliquer le fonctionnement de Zabbix, ainsi que les fonctionnalités qui nous paraissent intéressantes, puis nous allons décrire un exemple de déploiement très simple.

2 Fonctionnement de Zabbix

Avant de nous lancer dans le déploiement, voyons voir le fonctionnement et les fonctionnalités intéressantes que propose la bête.

2.1 Architecture globale

Zabbix est avant tout un serveur chargé de la supervision qui est composé de trois parties :

- le moteur de supervision, écrit en C, qui s'occupe de collecter les informations à monitorer et de générer les alertes.

- une base de données pour stocker les informations monitorées, ainsi que la configuration de Zabbix.

- une interface web, écrite en PHP, qui permet de visualiser les graphiques, les alertes et l'administration du moteur de supervision.

Chaque partie peut être placée sur des machines séparées, vu que le lien entre le moteur de supervision et l'interface n'est que la base de données.

avec Zabbix

2.2 Récupération des données

Le moteur de supervision peut récupérer les informations à monitorer à travers ces différents moyens :

- un agent SNMP (*switch*, *routeur*,...)

Zabbix demande à l'agent SNMP ce dont il a besoin et agit en conséquence. De plus, un script `snmptrap.sh` qui utilise `snmptrapd` permet à Zabbix de récupérer les traps SNMP lancés par les agents SNMP.

- un agent Zabbix

Un agent Zabbix spécifique est installé sur chaque machine à surveiller, permettant ainsi le suivi de nombreux éléments (processus, mémoire, processeurs,...) qui sont récupérés par le superviseur à travers le réseau. Les agents Zabbix [2] sont disponibles pour de très nombreuses plateformes allant des Windows (tm) aux GNU/Linux en passant par les *BSD et les Solaris. Nous reparlerons plus en détail des agents Zabbix dans la suite de cet article.

- des tests externes

Le serveur Zabbix lance des tests externes au travers de scripts ou binaires locaux qui ont pour rôle de récolter l'information à monitorer. Ceci permet d'effectuer des tests simples, tels que tester la présence d'une page sur un serveur web ou la connexion à une base de données. Ils permettent surtout de s'affranchir de l'installation d'un agent sur l'équipement à monitorer.

- tests natifs

Le serveur Zabbix possède une batterie de tests que nous pouvons utiliser nativement. Parmi ces derniers, on retrouve la possibilité de faire des connexions TCP, des pings...

2.3 L'agent Zabbix

Comme nous l'avons vu précédemment, l'agent Zabbix est un moyen très performant pour récupérer le maximum d'informations pour le moteur de supervision, puisque les données proviennent directement des machines à monitorer. Le désavantage de ce procédé pour la collecte des données est que cela nécessite un passage sur toutes les machines à monitorer pour installer et configurer l'agent Zabbix. Cependant, ce passage qui vous coûtera 5 processus, 3 mo de RAM et une utilisation de 0.1% du CPU par machine en vaut sûrement la chandelle, puisque la quantité d'informations remontées par les agents est impressionnante. À titre d'exemple, un agent Zabbix pour un Linux 2.6 peut envoyer environ 180 informations au superviseur. Cela va des classiques utilisation du CPU, occupation mémoire, liste des processus à des informations plus exotiques comme les taux d'erreurs sur les transmissions réseau, les vitesses de lecture et d'écriture sur les disques...

Pour interroger un agent zabbix, tel que le fait le superviseur, nous pouvons utiliser le programme `zabbix_get` qui prend l'adresse de l'agent, ainsi que l'information à récupérer.

```
(clem1@clem1:~)$ zabbix_get -sgruik.clem1.be -k"system.cpu.load[all]"
0.01000
```

Ici, nous récupérons la charge CPU de la machine **gruik.clem1.be** sur laquelle tourne un agent Zabbix pour Linux 2.6.

Dans le sens inverse, sur les machines qui possèdent un agent Zabbix, nous avons le programme `zabbix_sender` qui permet d'envoyer des données au serveur de supervision Zabbix sans passer par l'agent. Il est possible d'utiliser cette commande dans des scripts pour envoyer des informations au moteur de supervision. Par exemple, si nous avons une sauvegarde quotidienne réalisée par l'intermédiaire de Bacula [3], nous pouvons remplacer l'envoi de mails émis lors de chaque sauvegarde dans la configuration par un script *shell* qui va renseigner au moteur de supervision si la sauvegarde s'est bien déroulée ou non.

Il est à noter qu'un agent Zabbix peut être installé sur le serveur de supervision.

2.4 L'interface web

Comme vous l'avez sûrement compris, l'interface web est un élément essentiel pour que l'administrateur puisse configurer son système de supervision et pour que les utilisateurs puissent visualiser les informations récoltées sous différentes formes que nous détaillerons un peu plus tard. Même si elle a fortement évolué depuis les dernières versions, l'interface web reste assez difficile à aborder au début et il faut quelques heures de manipulation pour arriver à faire ce que l'on veut. Son utilisation sera traitée dans la partie sur le déploiement.

2.5 Vocabulaire Zabbix

Avant de conclure cette partie, il nous semble crucial, pour bien comprendre le fonctionnement de Zabbix ainsi que sa configuration, de faire un tour d'horizon du vocabulaire utilisé sur l'interface web. Nous allons donc reprendre un par un chaque terme.

2.5.1 Item

Un item est une donnée mesurée. Nous définissons un item par une clef, puis nous pouvons spécifier les paramètres suivants :

- le délai de rafraichissement de la donnée mesurée (flexible) ;
- le temps que la donnée restera dans l'historique ;
- le type (numérique, caractère,...) de la donnée et son unité (bps, unixtime,...).

Par exemple, avec l'agent Zabbix, nous avons un item pour récupérer la sortie de la commande `uname` à travers la clef `system.uname`. Cet item, de type `character`, est, par défaut, récupéré toutes les heures et gardé 7 jours dans l'historique.

2.5.2 Trigger

Un *trigger* est une expression qui permet de faire des tests sur des items, afin de déclencher des alarmes. Il est impérativement défini par un nom et une expression. Cette dernière indique l'état du trigger. Elle peut renvoyer trois valeurs qui sont **TRUE** pour indiquer un problème, **FALSE** pour état normal ou **UNKNOWN** pour indiquer un problème lors de l'évaluation de l'expression.

Par exemple, pour créer un trigger qui renvoie **TRUE** lorsque la charge du CPU devient trop importante, on crée l'expression suivante :

```
{gruik.clem1.be:system.cpu.intr.last(0)}>5
```

Nous prenons la dernière valeur **last(0)** de l'item **system.cpu.intr** provenant de la machine **gruik.clem1.be** et nous vérifions que ce dernier ne dépasse pas les 5. Si c'est le cas, la valeur **TRUE** est renvoyée par le trigger.

Pratiquement, tous les opérateurs arithmétiques et logiques sont supportés dans les expressions qui disposent également de nombreuses fonctions. Nous avons vu ici la fonction **last()** qui permet de récupérer la dernière valeur calculée de l'item, mais nous avons également d'autres fonctions intéressantes telles que :

- **avg** pour récupérer la moyenne des valeurs d'un item sur une période donnée ;
- **change** pour retourner la différence entre les deux dernières valeurs d'un item ;
- **diff** pour savoir si les deux dernières valeurs d'un item sont différentes ou non ;
- **max** et **min** pour avoir la valeur maximale et minimale d'un item sur une période donnée.

Les autres fonctions sont disponibles dans la documentation officielle de Zabbix [4].

2.5.3 Template

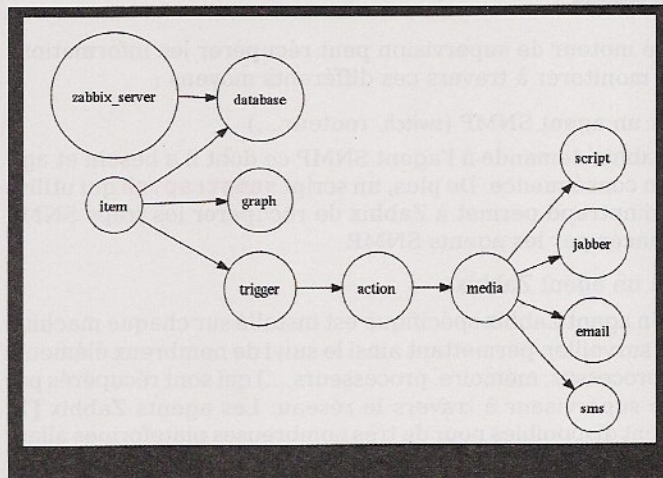
Un *template* permet de lier des hôtes avec des items, des triggers et des graphiques. C'est un excellent mécanisme pour administrer facilement des groupes de machines identiques dans le sens où une modification du template sera répercutée automatiquement sur les hôtes associées au template.

Ainsi, nous avons des templates pour les différents types d'équipements (machines Linux, routeur Cisco) qui contiennent les items, les graphes à afficher et les triggers qui vont bien.

2.5.4 Action

Comme son nom l'indique, une action permet d'exécuter quelque chose lorsqu'un événement se produit. Cet événement est défini sur la valeur de retour d'un ou plusieurs triggers et éventuellement en fonction du temps. Au niveau des actions, nous pouvons :

- lancer un script sur la machine distante ;
- envoyer un mail d'alerte ;
- envoyer un message à travers le protocole XMPP (Jabber) ;
- envoyer un SMS en passant par un modem GSM.



2.6 Fonctionnalités en vrac

Il serait difficile de décrire toutes les fonctionnalités que propose Zabbix dans cet article d'introduction. C'est pour cela que nous avons décidé de décrire dans cette partie uniquement celles qui nous paraissent intéressantes. La plupart de ces fonctionnalités sont apparues dans la version 1.4.

2.6.1 Découverte automatique

Zabbix intègre un module hautement configurable de découverte automatique des agents SNMP, des agents Zabbix et des équipements réseau (IP). Ce module se configure en lui spécifiant une plage d'adresses IP, un laps de temps entre chaque détection, les tests à réaliser, ainsi que les actions à réaliser lorsqu'une machine est détectée.

Ainsi, nous pouvons configurer Zabbix pour qu'il aille chercher, sur le réseau 192.168.0.0/24, toutes les machines où tourne un agent Zabbix. Pour chacune d'entre elles, on peut lui demander de récupérer la valeur de l'item **system.uname** pour ajouter la machine dans les machines à monitorer avec un template correspondant à sa plateforme.

2.6.2 IT services

La partie configuration de l'interface web de Zabbix possède un onglet appelé **IT services** qui permet de définir des contrats de niveau de service (SLA). Plus précisément, Zabbix fournit une interface de haut niveau qui corrèle différentes informations afin de fournir des données concises et exploitables compréhensibles facilement par des personnes non techniciennes.

Par exemple, nous pouvons monitorer une base de données fournie par un prestataire pour vérifier qu'il respecte bien

son contrat garantissant un taux de disponibilité supérieur à 85% par mois et avec une résolution des incidents en moins de 4 H. Les monsieurs Cravates pourront avec cela se rendre sur l'interface de Zabbix pour vérifier la couleur de l'indicateur SLA.

Un exemple concret d'utilisation sera décrit dans la partie déploiement.

2.6.3 Supervision de site web

Zabbix intègre un module de supervision de site web. Ce module permet de vérifier le bon fonctionnement d'un site web en simulant des scénarios. Il gère les requêtes **GET** et **POST** pour vérifier les formulaires, ainsi que les cookies pour se connecter à des pages nécessitant une authentification.

2.6.4 Supervision de journaux

Zabbix permet également de monitorer des fichiers de logs. Chaque ligne du fichier de log est alors considéré comme un item sur lequel on peut placer des triggers exactement comme ceux provenant des agents Zabbix.

On peut par exemple demander à Zabbix de monitorer le fichier **auth.log** afin de récupérer des statistiques sur les accès à distance ou de nous alerter dès qu'une tentative de *brute force* SSH est détectée.

3 Déploiement de Zabbix

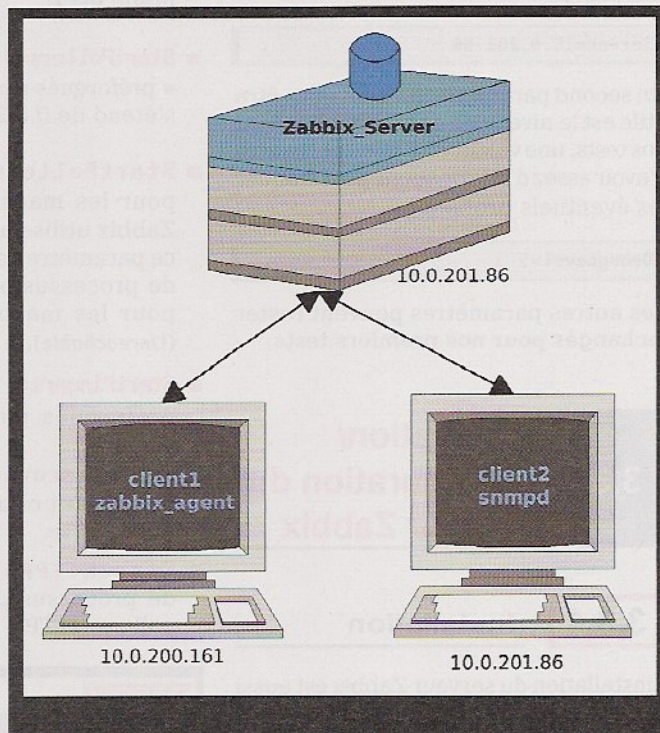
Maintenant que nous en savons plus sur le fonctionnement de Zabbix, voyons ensemble comment son administration se présente à travers un simple exemple de déploiement.

3.1 Pré-requis

3.1.1 Pré-requis matériels

Voici les spécifications matérielles que l'on peut trouver dans la documentation officielle de Zabbix :

				hosts
Small	Ubuntu Linux	P2 350MHz 256MB	MySQL MyISAM	20
Medium	Ubuntu Linux 64 bit	AMD Athlon 3200+ 2GB	MySQL InnoDB	500
Large	Ubuntu Linux 64 bit	Intel Xeon Dual Core 6400 4GB RAID	MySQL InnoDB or PostgreSQL	>1000
Very large	RedHat Enterprise	Intel Xeon 2 CPU 8GB RAID	MySQL InnoDB or PostgreSQL	>10000



Les trois machines tournent sous Ubuntu Server 8.04 : **client1** fait tourner un agent Zabbix, tandis que **client2** fait tourner un agent SNMP. Comme son nom l'indique, **Zabbix Serveur** est le serveur Zabbix qui intègre un serveur LAMP.

3.1.2 Pré-requis logiciels

Au niveau logiciel, il n'y a pas grand chose à dire : n'importe quel serveur LAMP ou BAMP fera l'affaire.

3.1.3 Environnement de test

Nous utiliserons au cours de cet article l'environnement de test suivant :

3.2 Installation/Configuration de l'agent Zabbix sur client1

3.2.1 Installation

L'installation de l'agent Zabbix ne pose aucun problème particulier que ce soit à partir des sources ou à partir d'un paquet binaire. Nous utilisons ici les paquets fournis dans la distribution Ubuntu Server.


```
# aptitude install zabbix-agent
```

3.2.2 Configuration

Le fichier principal de configuration se trouve dans `/etc/zabbix/zabbix_agentd.conf`. L'agent est d'ores et déjà fonctionnel avec la configuration par défaut, mais il y a tout de même certains points intéressants à aborder :

```
Server=localhost
```

Ce paramètre permet de définir avec quel serveur Zabbix, l'agent est autorisé à communiquer. C'est le seul paramètre par défaut qui s'avère bloquant pour la suite si on le laisse en l'état. Il faut indiquer l'adresse IP de notre serveur Zabbix.

```
Server=10.0.201.86
```

Un second paramètre qui peut nous être utile est le niveau de débogage. Pendant nos tests, une valeur de 5 nous permettra d'avoir assez d'informations pour traquer les éventuels problèmes.

```
DebugLevel=5
```

Les autres paramètres peuvent rester inchangés pour nos premiers tests.

3.3

Installation/ Configuration du serveur Zabbix

3.3.1 Installation

L'installation du serveur Zabbix est aussi simple que celle de l'agent. On s'en sort cette fois-ci avec la commande suivante :

```
# aptitude install zabbix-server-mysql  
zabbix-frontend-php zabbix-agent
```

Note

Le paquet **zabbix-server-mysql** permet de configurer la base de données Zabbix avec **debconf**. Si vous n'utilisez pas Debian, pas de panique, créez un utilisateur et une base de données Zabbix. L'installation des tables se fait plus tard dans l'interface web.

3.3.2 Configuration

Le fichier de configuration de **zabbix-server** se trouve dans `/etc/zabbix/zabbix_server.conf`.

Encore une fois, rien de bien méchant, le fichier de configuration n'est pas vraiment volumineux et chaque paramètre est précédé de quelques explications.

Tout comme pour l'agent, les valeurs par défaut sont correctes, mais l'étude de la configuration nous en apprendra davantage sur le fonctionnement de Zabbix.

■ **NodeID=0** : définit le numéro du nœud dans le cas où Zabbix est en mode distribué. Comme nous sommes en mode *standalone*, le numéro du nœud est 0.

■ **StartPollers=5** : nombre de processus « préforqués » pour le *poller* ; la valeur s'étend de 0 à 255.

■ **StartPollersUnreachable=1** : pour les machines indisponibles, Zabbix utilise un poller spécifique ; ce paramètre définit donc le nombre de processus préforqués du poller pour les machines indisponibles (*Unreachable*).

■ **StartPingers=1** : nombre de processus préforqués pour les *icmp pingers*.

■ **StartDiscoverers=1** : nombre de processus préforqués pour l'outil de découverte.

■ **StartHTTTPollers=1** : nombre de processus préforqués pour les pollers HTTP.

■ **UnreachablePeriod=45** : une machine ne donnant plus signe de vie pendant 45 secondes est considérée comme indisponible.

■ **UnavailableDelay=15** : pendant les 45 secondes où la machine ne donne plus signe de vie (*unreachable*), Zabbix va retenter de la contacter 15 fois avant de déclarer l'hôte comme indisponible.

■ **PingerFrequency=60** : fréquence du *pinger*.

3.4

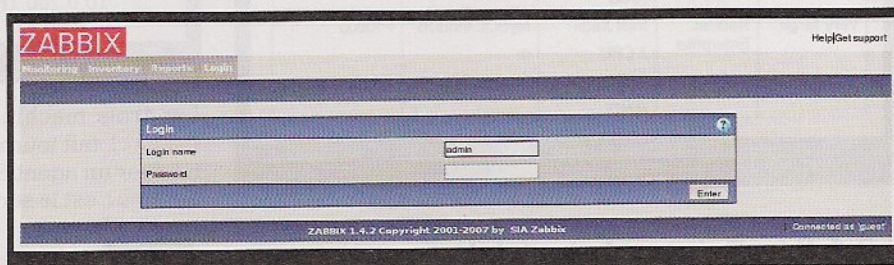
Premiers contacts avec l'interface

Ouf, c'est le moment pour les intégristes de la ligne de commande d'aller faire un tour. Nous allons maintenant nous attaquer à l'interface web de Zabbix. Contrairement à Nagios où l'interface n'est qu'un outil permettant de voir visuellement les paramètres du serveur et la santé des machines surveillées, l'interface de Zabbix est la clé de voûte de la configuration des paramètres de supervision.

3.4.1

Installer la base de données

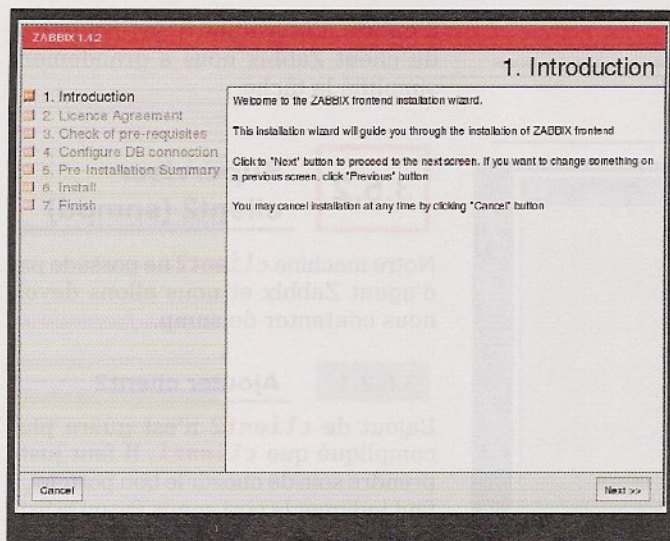
Nous ouvrons notre navigateur à l'adresse <http://mon-serveur/zabbix>. Si tout se passe bien, vous devriez voir l'écran d'authentification. Par défaut, le login est **admin** et il n'y a pas de mot de passe.



■ **HousekeepingFrequency=1** : fréquence de suppression des données inutilisées (*history table*).

■ **SenderFrequency=30** : quand des alertes ne sont pas envoyées, Zabbix fait une nouvelle tentative après un certain délai.

Maintenant que nous sommes authentifié, rendez-vous dans l'onglet administration, puis installation. Un écran va vous guider étape par étape dans l'installation de la base de données. Aucune difficulté n'est à noter à cette étape.



Par exemple, le trigger « *apache is not running* » déclenchera une alerte si ce trigger est à vrai.

■ Web

Permet de monitorer des applications web. Nous étudierons ce module par la suite.

■ Latest data

Correspond aux dernières données collectées.

■ Triggers

Affiche les triggers actifs.

■ Events

Liste tous les événements qui se sont produits.

■ Actions

Affiche les actions réalisées en fonction de l'état des triggers. Pour rappel, les actions possibles sont :

- envoyer un mail
- envoyer un message via Jabber
- envoyer un sms

■ Maps

Représente les équipements supervisés sur une carte.

3.4.2 Étude rapide de l'onglet monitoring

On peut se rendre maintenant dans le menu configuration, puis dans l'onglet *hosts*. Normalement, vous devriez voir votre serveur Zabbix (**localhost**) avec l'état **Not monitored**. On fait un petit clic pour l'activer et notre machine locale est maintenant monitorée.



Un passage par le menu monitoring s'impose pour apprécier les premiers résultats et expliquer brièvement quelques informations importantes que l'on peut obtenir dans cette partie.

■ Overview

Nous présente une vue complète des services supervisés par machine. À noter que « services » n'est pas vraiment le bon mot. On parlera plutôt ici de TRIGGERS. En effet, Zabbix définit les services à monitorer comme des triggers déclenchant une alerte.

■ Graphs

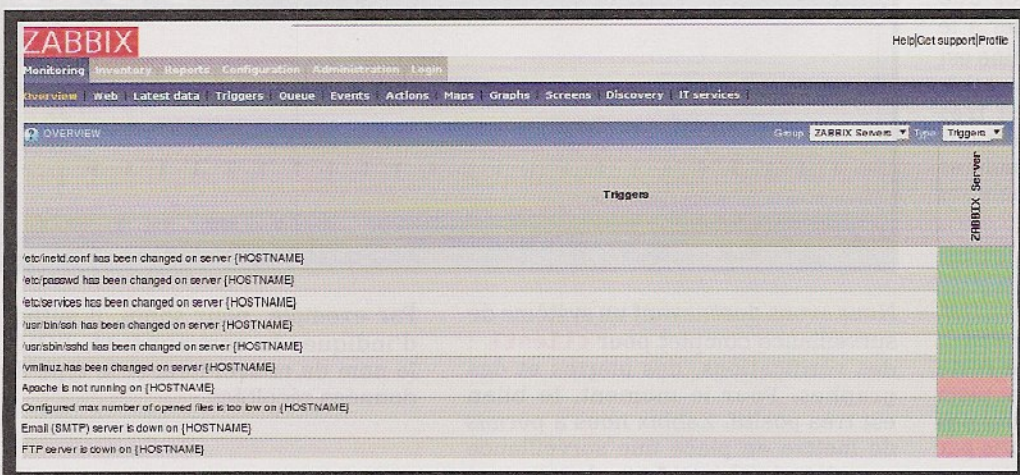
Graphes des données collectées par Zabbix.

■ Screen

Permet de créer son tableau de bord personnalisé avec ses graphes, ses cartes, etc.

■ Discovery

Permet la découverte automatique des machines à superviser.



3.5

Superviser client1 et client2

Revenons maintenant dans notre petit labo. Pour rappel, nous devons superviser **client1** et **client2**. L'un possède un agent Zabbix et l'autre ne possède qu'un démon snmpd.

3.5.1 Superviser client1

3.5.1.1 Ajouter client1

Dans notre cas, le groupe **linux-servers** nous convient parfaitement. Attention à choisir le bon port. Ici, nous ajoutons une machine qui possède un **agent_zabbix**.

Vous avez également le choix de lier la machine à un template. Le template contient un ensemble de triggers prédéfinis qui vont nous simplifier la vie. Pour une première découverte du logiciel, je ne saurais que trop vous conseiller d'utiliser un template.

3.5.1.2 Ajouter une alerte

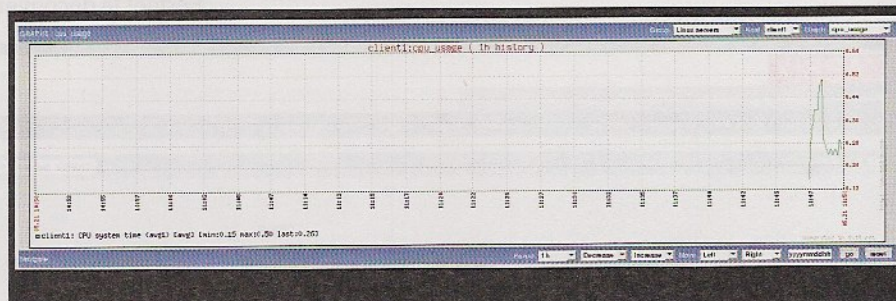
L'ajout d'alerte se fait via l'onglet action. Au sens de Zabbix, une alerte est une action déclenchée suite à un évènement (triggers).

Sur notre exemple, on définit une action « mail » qui va envoyer une alerte par mail lorsqu'un trigger a le statut « average » et que la machine est **client1**. Ce mécanisme d'opération logique entre les triggers est très intéressant, car il permet de créer des alertes personnalisées déclenchées suite à la corrélation de plusieurs évènements.

3.5.1.3 Ajouter des graphes

Pour que le plaisir soit total, il ne reste plus qu'à ajouter des graphes avec plein de belles couleurs. ;-) Ne soyez pas pessimiste, vous allez être surpris par la facilité déconcertante de Zabbix pour

ajouter un graphe. Pour rappel, Zabbix collecte un certain nombre d'informations (items) qui permettent de mettre en place les triggers et les actions. Pour les graphes, cela fonctionne de manière similaire. Il suffit d'indiquer le ou les items que l'on souhaite grapher et le tour est joué. Des options supplémentaires sont disponibles, telles que le type de graphe, les couleurs, etc...



Nous avons dorénavant un système de surveillance complet pour **client1** : des vérifications, des alertes et des graphes. Pour le moment, le bilan est très positif. Zabbix nous a permis de mettre en place une surveillance complète de **client1** et cela sans trop

d'efforts. L'utilisation d'un template et du client Zabbix nous a grandement simplifié la tâche.

3.5.2 Superviser client2 (snmpd)

Notre machine **client2** ne possède pas d'agent Zabbix et nous allons devoir nous contenter de snmp.

3.5.2.1 Ajouter client2

L'ajout de **client2** n'est guère plus compliqué que **client1**. Il faut juste prendre soin de choisir le bon port. Ici, il faut indiquer le port snmp. Parmi la liste des templates, on peut remarquer qu'il en existe un pour le snmp. Comme pour **client1**, je vous conseille d'utiliser un template. Notre machine **client2** est maintenant monitorée, mais les items du template snmp sont un peu moins intéressants que ceux récoltés par l'agent Zabbix. Il va donc falloir penser à créer ses propres items et triggers si le besoin s'en fait ressentir.

3.6

Allez plus loin dans la supervision

3.6.1 Créer un item

Si l'on souhaite récolter des informations bien spécifiques sur nos hôtes à superviser, il faut pouvoir créer de nouveaux items. Pour ne pas perdre les bonnes habitudes, l'ajout d'un nouvel item est simplissime.

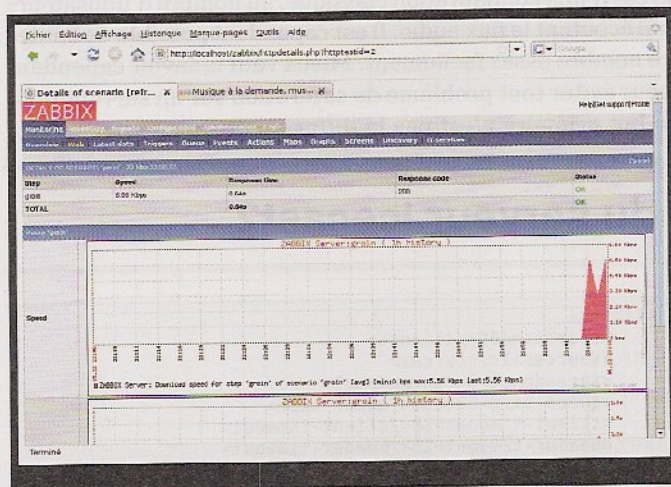
Par exemple, pour snmp, il suffira d'indiquer la communauté, l'OID, le nom du nouvel item et le type de données récoltées.

3.6.2 Créer un trigger

Il n'y a rien à dire de plus sur les triggers que ce qui a été dit dans la première partie, mis à part peut-être qu'il faut connaître un minimum leur fonctionnement pour en créer de nouveaux. Comme pratiquement tous les opérateurs arithmétiques et logiques sont supportés, les possibilités, lorsqu'elles sont couplées aux fonctions, sont quasi infinies.

Vous décrire les différentes étapes de la création d'un trigger serait ici un peu lourd. L'important est que vous ayez compris le système.

3.7 Web monitoring



Pour finir cette partie, nous abordons ici une fonctionnalité très intéressante de Zabbix qui permet de créer des scénarios web. Cela permet de simuler le parcours complexe d'un utilisateur en train de surfer sur un site web. Cet outil supporte les méthodes **GET** et **POST** offrant ainsi une flexibilité très appréciable. On peut par exemple imaginer simuler le remplissage d'un formulaire, l'inscription à un service en ligne ou même, pourquoi pas, vérifier le bon fonctionnement de l'interface web de Zabbix.

Le module web va périodiquement exécuter le scénario, créer et collecter des informations de performances :

- temps de réponse ;
- vitesse de téléchargement ;
- code de retour de la page.

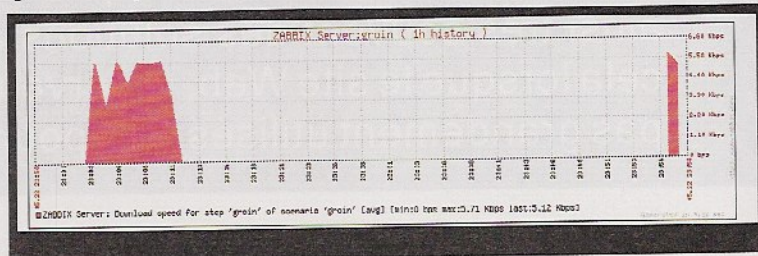
Il est également possible de vérifier la présence d'une chaîne de caractères dans la page.

Des items associés au scénario sont ensuite automatiquement créés.

- **web.test.in[Scenario, bps]** va renvoyer la valeur de la vitesse de téléchargement d'une étape du scénario ou du scénario complet.
- **web.test.fail[Scenario]** qui renverra 1 si le scénario échoue et 0 si tout s'est bien passé.

Il est ensuite possible de créer les triggers que l'on souhaite à partir de ces items.

Le résultat est assez sympathique. Chaque étape du parcours est graphée avec le temps de réponse et la vitesse de téléchargement. Par exemple, sur la capture ci-dessous, on peut voir que le site d'un des auteurs a eu quelques problèmes pendant un petit moment. Forcément, c'est ce qui arrive quand on a un serveur web en Python... :-)



4 Conclusion

Zabbix est un bon logiciel libre de supervision. Malgré sa jeunesse et son faible nombre de supporters, ce logiciel tout en un est facile à prendre en main et permet ainsi de mettre en place un système de supervision performant en peu de temps. Actuellement, Zabbix est en version 1.4. La prochaine version, déjà disponible en bêta, proposera des fonctionnalités alléchantes, telles que le support d'IPv6 et de LDAP, des améliorations sur les graphes (zoom, 3D)...

Le problème reste de savoir ce que l'on doit et ce que l'on ne doit pas monitorer, mais ce n'est pas le rôle de Zabbix. :-)



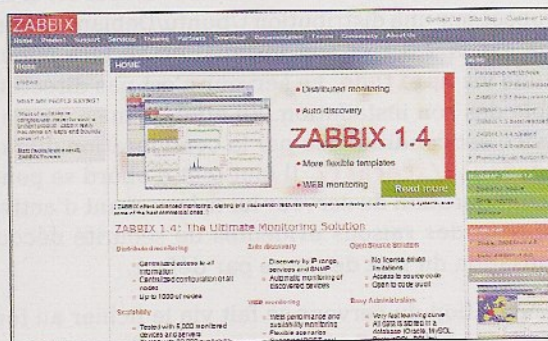
Sylvain Gallerand, Clément Lecigne

Sylvain Gallerand, utilise Linux depuis 2000. Etudiant en licence de sécurité informatique par apprentissage. Poste en entreprise : assistant RSSI

Clément Lecigne, apprenti ingénieur en sécurité informatique à l'université de Marne la Vallée.

Bibliographie

- [1] <http://www.zabbix.com>



- [2] <http://www.zabbix.com/download.php>
- [3] <http://www.bacula.org>
- [4] <http://www.zabbix.com/documentation.php>

Monter un service de diffusion

Les offres d'hébergeurs utilisent souvent comme argument de vente pour leurs offres la bande passante mise à disposition. On trouve ainsi très souvent des solutions à 100 Mb/s. Mais que faire de tout cela lorsque le site Web, le serveur de mail ou le FTP hébergé ne sont pas grandement utilisés ? Et pourquoi ne pas diffuser un flux audio ?

Certes, ceci revient à passer d'un extrême à l'autre. La diffusion de flux audio, le *streaming*, peut être un très grand consommateur de bande passante et éventuellement pénaliser les autres services en fonction sur la machine. La consommation de bande passante est proportionnelle, dans

le cas d'une configuration classique, au nombre d'utilisateurs qui écoutent le flux audio. Il est cependant possible de limiter ce nombre assez facilement. Mieux vaut le savoir cependant pour éviter tout problème de saturation sur un serveur dont le but primaire n'est pas la diffusion de flux audio.

1

Structure et configuration du serveur Icecast2

Icecast est un projet de développement d'une solution de streaming audio. Il se compose de trois éléments distincts :

- le serveur Icecast chargé d'envoyer le flux audio au travers d'Internet vers les applications clientes des auditeurs ;
- la bibliothèque **libshout**, fournissant une API permettant aux applications de communiquer avec le serveur ;
- le programme IceS, qui envoie les données audio au serveur qui va les *streamer*.

Le serveur Icecast est capable de streamer aussi bien en MP3 qu'en Ogg. Notez cependant que le streamer IceS ou plus exactement IceS2 ne permet que le stream Ogg depuis une liste de lecture (*playlist*) ou une source live. Pour diffuser des données au format MP3, vous devrez donc vous tourner vers un autre streamer.

Préparons tout d'abord l'installation du serveur Icecast2 tel que proposé sur une distribution Ubuntu/Debian. La version actuelle est la 2.3.1 et son installation se fera comme à l'habitude via **aptitude**. Le serveur n'est pas directement utilisable dès son installation. En effet, la mise en marche du service est conditionnée par la configuration du fichier **/etc/default/icecast2**. Il faut tout d'abord se pencher sur la configuration du serveur lui-même avant d'activer le service pour des raisons évidentes de sécurité découlant de l'utilisation du mot de passe par défaut.

La configuration du serveur se fait via le fichier au format XML **/etc/icecast2/icecast.xml**. Nous y trouvons plusieurs sections clairement organisées. La configuration elle-même est comprise entre les balises **<icecast>** et **</icecast>**. Nous avons tout d'abord les éléments imposants les limites :

```
<limits>
  <clients>100</clients>
  <sources>2</sources>
  <threadpool>5</threadpool>
  <queue-size>524288</queue-size>
  <client-timeout>30</client-timeout>
  <header-timeout>15</header-timeout>
  <source-timeout>10</source-timeout>
  <burst-on-connect>1</burst-on-connect>
  <burst-size>65535</burst-size>
</limits>
```

Les balises **clients** et **sources** parlent d'elles-mêmes. Remarquez qu'il est possible d'avoir plusieurs sources distinctes. Nous en reparlerons plus loin. Le nombre de clients détermine implicitement la bande passante maximum qui sera utilisée. Il ne s'agit pas simplement de charge de serveur, mais également de respect des conditions générales imposées par votre hébergeur. Il n'est pas rare d'y trouver, en effet, des mentions concernant l'utilisation « raisonnable » de la bande passante parmi les restrictions plus courantes (interdiction de *peer-to-peer* et/ou de services IRC).

threadpool détermine le nombre de *threads* à lancer pour accueillir les connexions clientes. Si votre serveur est susceptible de faire face à un grand nombre de connexions simultanées, il peut être intéressant d'augmenter cette valeur afin de réduire le délai d'attente découlant de la création d'un nouveau thread. Il ne faut pas oublier, cependant, la charge induite si votre serveur n'est pas très « musclé ».

queue-size fixe la taille maximum de la file d'attente en octet. Lorsqu'un client est connecté, il reçoit normalement les données audio sous forme de flux constant. Un problème réseau peut toutefois survenir entraînant une latence. Un tampon est donc utilisé pour déterminer à partir de quelle limite le client doit être éliminé du *stream*. Cette valeur par défaut est normalement suffisante pour supporter n'importe quel petit problème réseau classique.

audio avec Icecast2

client-timeout, **header-timeout** et **source-timeout** permettent respectivement de fixer un délai d'attente maximum en secondes pour l'expiration d'une connexion, le temps entre la connexion du client et une requête et l'expiration d'un flux source. Ces valeurs permettent de réagir rapidement aux problèmes afin de ne pas maintenir dans les listes de connexions des sources ou des clients injoignables ou déconnectés de force.

burst-on-connect est une option très intéressante pour les clients utilisant une mémoire tampon (*buffering*). Il s'agit ici d'envoyer une certaine quantité de données d'un coup pour remplir de tampon du client. Il en découle un gain significatif pour les auditeurs puisque le flux audio est plus rapidement audible. Le volume de données envoyées de cette manière est fixé par **burst-size** (ici 64 ko).

```
<authentication>
  <source-password>coucou</source-password>
  <relay-password>coucou</relay-password>
  <admin-user>admin</admin-user>
  <admin-password>coucou</admin-password>
</authentication>
```

Cette section est relativement claire. Là où les sources audio doivent s'authentifier auprès du serveur Icecast avant d'envoyer les données. Ce type d'authentification légère pourra, bien entendu, être renforcée via la création d'un VPN ou d'un tunnel entre la source et le serveur. Selon la configuration, le streamer, la source, fonctionnera sur le même hôte que le serveur ce qui simplifie grandement la gestion de la sécurité. Ceci n'est envisageable que dans le cadre de l'utilisation d'une liste de lecture. On imagine difficilement pouvoir utiliser une source live (entrée *line-in* de la carte son) sur un serveur et a fortiori sur le serveur loué par un hébergeur.

Un serveur Icecast est en mesure de relayer les flux audio d'un autre serveur. L'utilisateur par défaut est **relay**, mais pourra être changé avec l'élément **relay-user**. Un serveur esclave, qui va relayer les flux, se connecte au serveur maître pour obtenir la liste des flux et les relayer. Cette architecture permet d'utiliser plusieurs serveurs afin de répartir la charge système, mais également la bande passante.

Enfin, nous avons **admin-user** et **admin-password** qui permettent respectivement de définir un nom d'utilisateur et un mot de passe pour la page d'administration du serveur. En effet, le serveur Icecast dispose d'une interface HTML permettant de gérer les connexions et les flux.

Nous avons ensuite les informations concernant la connectivité du serveur :

```
<hostname>localhost</hostname>
<listen-socket>
  <port>8000</port>
</listen-socket>
```

hostname définit le nom de la machine tel qu'il sera utilisé dans les URL de l'interface Web. Celle-ci n'est pas uniquement disponible pour l'administrateur, mais également pour les auditeurs, leur permettant ainsi d'avoir une vue d'ensemble des informations sur les flux disponibles. **listen-socket** et, à l'intérieur, **port** déterminent le port utilisé à la fois pour les connexions des clients, mais également pour l'interface

Web et les connexions du ou des streamers. Il est possible d'attacher le serveur Icecast uniquement à une interface en spécifiant une adresse IP via **bind-address**. Il est possible de spécifier plusieurs adresses en multipliant les blocs **listen-socket**. En l'absence de la directive **bind-address**, le serveur est en attente de connexions sur toutes les interfaces disponibles.

Viennent ensuite les configurations des différents chemins :

```
<paths>
  <basedir>/usr/share/icecast2</basedir>
  <logdir>/var/log/icecast2</logdir>
  <webroot>/usr/share/icecast2/web</webroot>
  <adminroot>/usr/share/icecast2/admin</adminroot>
  <alias source="/" dest="/status.xml"/>
</paths>
```

basedir n'est utilisé que dans le cas d'un serveur *chrooté*. On utilisera alors :

```
<security>
  <chroot>1</chroot>
</security>
```

pour activer cette option. Bien entendu, il faut au préalable avoir préparé le répertoire spécifié pour le **chroot**. **logdir** renseigne sur l'emplacement des journaux d'activité (voir plus loin). Les deux éléments de configuration, **webroot** et **adminroot** déterminent respectivement l'emplacement des éléments HTML pour les interfaces Web générale et d'administration. Notez qu'avec Debian et Ubuntu les fichiers XSL contenus dans **/usr/share/icecast2/web** par exemple sont des liens symboliques vers des fichiers placés dans **/etc/icecast2/web**. Ceci évite de modifier des données dans **/usr**, ce qui serait contraire aux règles d'usages.

alias peut s'avérer très intéressant, puisque cette directive permet d'éventuellement simplifier les liens vers des pages de l'interface Web. Ici, par défaut, la racine est un alias vers la page présentant l'état du serveur et des flux.

Enfin, pour terminer, nous avons les différentes informations sur les journaux d'activité :

```
<logging>
  <accesslog>access.log</accesslog>
  <errorlog>error.log</errorlog>
  <loglevel>4</loglevel>
  <logsize>10000</logsize>
  <logarchive>1</logarchive>
</logging>
```

accesslog et **errorlog** se passent de commentaire. Le niveau d'information spécifié par **loglevel** sera choisi entre 1 et 4 définissant, respectivement, les messages de débogage, d'information, d'avertissement et d'erreur. Afin d'éviter un grossissement incontrôlé des fichiers journaux, **logsize** permet de spécifier une taille maximum en Ko. Si l'option **logarchive** est, comme ici, spécifiée, une rotation des journaux aura lieu automatiquement via un ajout de date en guise d'extension. Dans le cas contraire, l'actuel journal est renommé avec une extension **.old** et un nouveau sera créé. À la rotation suivante, le journal **.old** sera écrasé.

Notre serveur est maintenant prêt à fonctionner. Il ne nous reste qu'à modifier le fichier **/etc/default/icecast2** pour changer la ligne :


```
ENABLE=false
en
ENABLE=true
```

puis à lancer le service avec `/etc/init.d/icecast2 start`. Dès lors, le serveur est en route et accessible via le port spécifié dans la configuration (8000). Pointez un navigateur sur **localhost:8000** et vous obtiendrez la page d'état (Figure 1).



Figure 1 : Interface Web d'Icecast

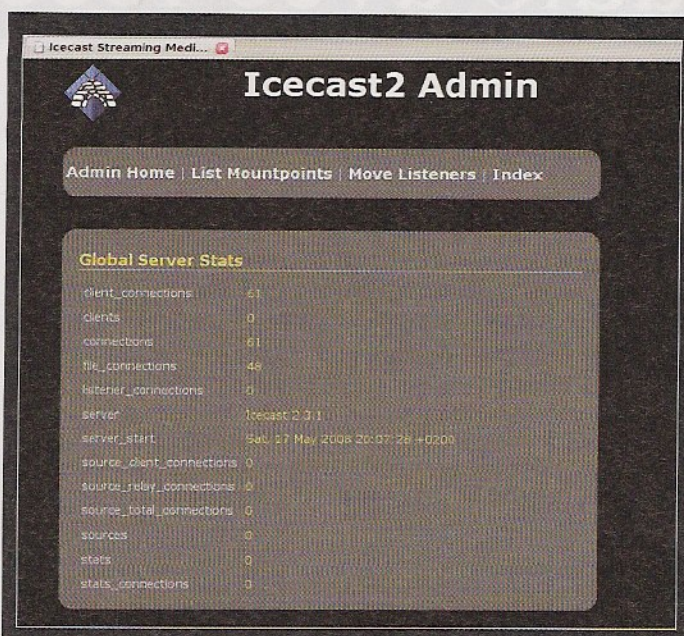


Figure 2 : Interface Web d'administration d'Icecast

2

Configuration du streamer IceS

Nous disposons maintenant d'un serveur capable de diffuser un flux audio pour quelques 100 clients. Encore faut-il diffuser quelque chose. Nous devons nous pencher sur la configuration du streamer IceS2. Celle-ci est également au format XML et reprend une syntaxe très proche du fichier de configuration du serveur Icecast. La configuration d'IceS est comprise entre les balises `<ices>` et `</ices>`. Tout d'abord, nous avons les éléments de configuration concernant le fonctionnement du streamer à proprement parler :

```
<background>0</background>
<logpath>./logpath>
<logfile>ices.log</logfile>
<loglevel>4</loglevel>
<consolelog>1</consolelog>
```

background indique un fonctionnement en arrière-plan. Dans la phase d'expérimentation, on préférera naturellement ne pas détacher le streamer de la console. **logpath** et **logfile** indique l'emplacement et le nom du fichier journal. **loglevel** est identique à l'élément du même nom pour le serveur. Enfin, **consolelog** permet d'envoyer les messages sur la console en lieu et place du fichier journal, parfait pour les premiers essais.

Vient ensuite la configuration des différents flux. En effet, IceS est en mesure de produire plusieurs flux dont la configuration est comprise entre `<stream>` et `</stream>`. Là, nous avons tout d'abord les métadonnées, les informations qui apparaîtront dans la page d'état de l'interface Web :

```
<metadata>
  <name>Stream exemple</name>
  <genre>Divers pour test</genre>
```

```
<description>Votre premier stream avec IceS2</description>
</metadata>
```

Nous avons ainsi respectivement le nom du flux, une description du genre de diffusion et une description du flux lui-même. Nous configurons ensuite la source d'entrée des données audio. La source est définie par le module utilisé. Dans ce premier exemple, nous utiliserons le module **playlist** permettant d'utiliser une liste de lecture regroupant des morceaux au format Ogg :

```
<input>
  <module>playlist</module>
  <param name="type">basic</param>
  <param name="file">playlist.txt</param>
  <param name="random">0</param>
  <param name="restart-after-reread">0</param>
  <param name="once">0</param>
</input>
```

Nous retrouvons dans la configuration le module à utiliser et différents paramètres de configuration lui étant spécifiques. Le paramètre **type** détermine la manière d'obtenir la liste des pistes. Cette liste est un fichier contenant simplement le chemin et le nom des fichiers Ogg à utiliser. **type** peut être **basic** pour l'utilisation d'un nom de fichier ou **script**. Dans ce dernier cas, le script est lancé pour obtenir le nom d'un fichier Ogg à lire.

Le paramètre **file** renseigne sur le fichier contenant la liste de lecture. Ce paramètre n'est utilisable que dans le cadre du type **basic**. Dans le cas de l'utilisation d'un script, on précisera le chemin et le nom avec un paramètre **program**.

Note sur les scripts

La gestion des scripts est relativement basique, même avec les dernières versions d'IceS2. Un script ne doit retourner qu'un seul nom de fichier à la fois et le streamer n'acceptera pas de jouer deux fois le même fichier. Comme le précisent les développeurs sur les listes de diffusion, le support des scripts n'est, pour l'heure, qu'un *proof of concept* et mériterait d'être implémenté de manière plus complète. Vous pouvez cependant développer des scripts complexes en respectant ces indications.

Le reste des paramètres concerne le type **basic**. **random** permet de jouer les morceaux de manière aléatoire. **restart-after-reread** indique qu'il faut recommencer la lecture de la liste depuis le début si cette dernière a été mise à jour. Enfin, **once** permet de ne lire qu'une seule fois les morceaux contenus dans la liste, puis de quitter plutôt que de boucler.

La balise **<instance>** permet de préciser la manière de se connecter au serveur. Notez qu'il est possible d'utiliser plusieurs instances. Dans ce cas, IceS enverra le ou les flux vers les différents serveurs ou vers plusieurs points de montage sur un même serveur.

Nous n'avons pas encore parlé des points de montage. Il s'agit tout simplement des points d'accès vers les flux audio sur le serveur Icecast. À l'instar des systèmes de fichiers, vous pouvez considérer que les streams ou flux configurés dans IceS sont des systèmes de fichiers contenant les données. Pour les représenter et les rendre accessibles aux auditeurs, ils sont « montés » dans des points de montage. Nous allons configurer ici deux instances différentes pour que vous compreniez l'intérêt des points de montages d'une même source audio vers un seul serveur (et donc dans la même portée **stream**).

```
<instance>
  <hostname>127.0.0.1</hostname>
  <port>8000</port>
  <password>coucou</password>
  <mount>/example1.ogg</mount>
  <reconnectdelay>2</reconnectdelay>
  <reconnectattempts>5</reconnectattempts>
  <maxqueueLength>80</maxqueueLength>
  <encode>
    <nominal-bitrate>64000</nominal-bitrate>
    <samplerate>44100</samplerate>
    <channels>2</channels>
  </encode>
</instance>
```

Les directives **hostname**, **port** et **password** se passent de commentaire. **mount** détermine le nom du point de montage. Celui-ci apparaîtra dans l'interface Web. Ici, **example1.ogg** donnera un flux accessible via <http://localhost:8000/example1.ogg> et une liste de lecture au format M3U accessible via <http://localhost:8000/example1.ogg.m3u>.

reconnectdelay et **reconnectattempts** permettent de gérer les problèmes de connexion. Lorsque le serveur est coupé ou crashe, IceS se déconnecte. Le premier argument est un délai en secondes à laisser entre deux tentatives de reconnexion. Le second paramètre indique le nombre de tentatives à faire. **maxqueueLength** détermine la taille du

buffer à remplir avant d'envoyer les données au serveur Icecast. Il n'est généralement pas nécessaire de changer cette valeur.

Nous arrivons enfin à l'encodage du flux, objet primaire du streamer. Les paramètres sont placés entre les balises **<encode>** et **</encode>**. Il est possible d'ajuster avec précision la qualité du flux. La manière la plus simple est d'utiliser une facilité incluse dans IceS via la directive **quality**. On choisira alors une valeur entre -1 et 10, 3 étant la valeur par défaut. Une valeur de 0 avec un échantillonnage à 44100 Hz et deux canaux (stéréo) donne un **bitrate** autour de 64 Kbps. Vous pouvez vous limiter à l'utilisation de **quality** ou apporter des éléments de configuration complémentaires.

IceS encode les flux avec un **bitrate** variable (VBR), mais vous pouvez influencer sur différents points. **nominal-bitrate** indique le **bitrate** qu'IceS essaiera de maintenir. Les directives **maximum-bitrate** et **minimum-bitrate** permettent de définir les limites basse et haute. Pour utiliser ces options et contrôler ainsi la gestion du **bitrate**, vous ajouterez également **<managed>1</managed>**. Attention, la consommation CPU s'en trouve augmentée. Enfin, **samplerate** indique la fréquence d'échantillonnage et **channels** le nombre de canaux.

On détermine un second point de montage avec d'autres options :

```
<instance>
  <hostname>127.0.0.1</hostname>
  <port>8000</port>
  <password>coucou</password>
  <mount>/example2.ogg</mount>
  <reconnectdelay>2</reconnectdelay>
  <reconnectattempts>5</reconnectattempts>
  <maxqueueLength>80</maxqueueLength>
  <encode>
    <nominal-bitrate>32000</nominal-bitrate>
    <samplerate>22050</samplerate>
    <channels>1</channels>
  </encode>
</instance>
```

Ici, le **bitrate** est réduit de moitié, ainsi que la fréquence d'échantillonnage. Nous diffusons également un flux **downmixé** en mono (**channels** à 1). Il est d'usage de proposer plusieurs solutions pour les auditeurs. Bien que la majorité d'entre eux disposent sans doute de connexions haut débit, il faut penser aux problèmes de charge réseau, aux connexions RTC et aux liens Wifi poussifs.

Avant de nous lancer, nous devons créer le fichier de liste de morceaux **playlist.txt**. Rien de plus simple, il suffit de copier les fichiers Ogg dans le répertoire courant et un petit **/bin/ls *.ogg > playlist.txt** fera l'affaire. Si vous disposez de fichiers MP3, utilisez simplement le sympathique **mp32ogg** provenant du paquet du même nom. Notez que nous utilisons ici un fichier de configuration, des fichiers audio et un fichier de liste de morceaux placés dans le répertoire courant. Nous sommes en phase d'expérimentation, mais il faudra, bien entendu, préciser tous les chemins lorsqu'il s'agira de faire fonctionner réellement le streamer.

Il ne nous reste plus qu'à lancer notre streamer avec :

```
% ices2 ices-playlist.xml
INFO ices-core/main IceS 2.0.1 started...
signals/signal_usr1_handler Metadata update requested
playlist-basic/playlist_basic_get_next_filename
Loading playlist from file "playlist.txt"
```



```
INFO playlist-builtin/playlist_read
    Currently playing "the_well_tempered_clavier_bwv_846.ogg"
INFO stream/ices_instance_stream
    Connected to server: 127.0.0.1:8000/example1.ogg
DEBUG reencode/reencode_page
    Reinitialising reencoder for new logical stream
INFO encode/encode_initialise
    Encoder initialising in VBR mode:
    2 channels, 44100 Hz, nominal 64000
stream/ices_instance_stream
    Connected to server: 127.0.0.1:8000/example2.ogg
DEBUG reencode/reencode_page
    Reinitialising reencoder for new logical stream
INFO encode/encode_initialise
    Encoder initialising in VBR mode:
    1 channels, 22050 Hz, nominal 32000
INFO audio/resample_initialise
    Initialised resampler for
    1 channels, from 44100 Hz to 22050 Hz
INFO audio/downmix_initialise
    Enabling stereo->mono downmixing
```

On retrouve dans les informations s'affichant sur la console toutes les indications données dans la configuration. Nous avons bel et bien deux flux encodés comme il se doit. Côté interface Web du serveur Icecast, on retrouve nos points de montage (Figure 3).

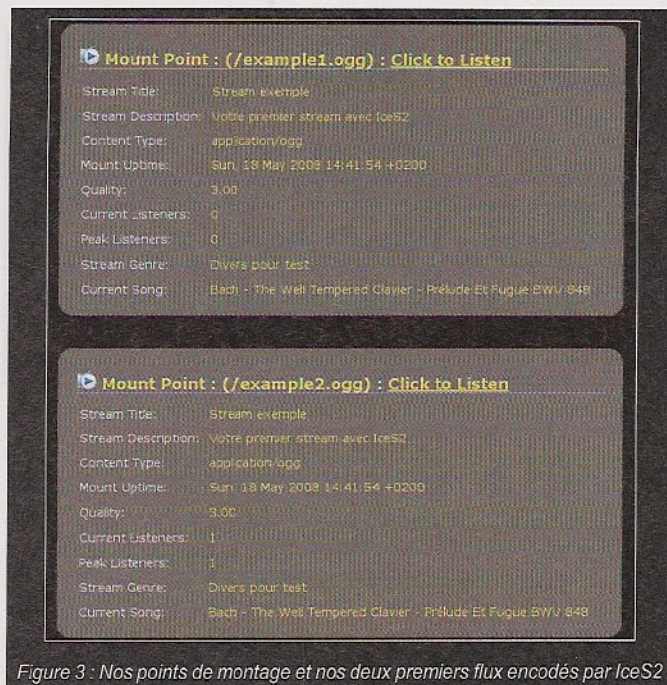


Figure 3 : Nos points de montage et nos deux premiers flux encodés par IceS2

3 Le cas du live

Le module **playlist** n'est, bien entendu, pas le seul disponible. Il est par exemple parfaitement possible de diffuser un flux audio live. Plusieurs modules permettant cela existent, mais le plus utilisé est sans le moindre doute **alsa**. Rappelons au passage qu'ALSA (Advanced Linux Sound Architecture) est le composant noyau remplaçant OSS pour la gestion du son.

Entrons dans le vif du sujet avec la section **input** de la configuration :

```
<input>
  <module>alsa</module>
  <param name="rate">44100</param>
  <param name="channels">2</param>
  <param name="device">plughw:0,0</param>
  <param name="periods">2</param>
  <param name="buffer-time">500</param>
  <param name="metadata">1</param>
  <param name="metadatafilename">metadata</param>
</input>
```

Le module **alsa** est ici spécifié, ainsi que ses paramètres. **rate** et **channels** fixent la fréquence d'échantillonnage et le nombre de canaux. On précise ensuite, via **device**, le périphérique ALSA à utiliser. **plughw:0,0** désignant directement un périphérique est la solution la plus directe. On peut cependant utiliser **dsnoop** et permettre ainsi à d'autres applications d'accéder au périphérique d'enregistrement. Dsnoop est, tout simplement, l'équivalent de Dmix pour l'enregistrement.

buffer-time permet d'indiquer un tampon (buffer) en millisecondes. Paramètre plus intéressant, **metadata** permet d'activer les métadonnées sur le flux. Dans le cas d'une liste de lecture, les informations affichées proviennent des tags présents dans chaque fichier. Bien entendu, avec un flux audio live, ces informations n'existent pas. IceS permet cependant d'ajouter des métadonnées en provenance d'un fichier spécifié avec le paramètre **metadatafilename**. Ces données peuvent être modifiées et mises à jour. Il suffit, en effet, de modifier le contenu du fichier et d'envoyer un SIGUSR1 au processus **ices2** avec, par exemple :

```
% kill -USR1 `pidof ices2`
```

Les sources d'enregistrement ALSA

Le support des périphériques son via ALSA est très avancé. Les périphériques disposent habituellement de plusieurs sources d'entrées (micro, line-in, CD, etc.). Il n'est pas rare de voir, par exemple, un tuner branché à la carte son (ici une Ensoniq 5880 AudioPCI), de lancer un enregistrement et de n'obtenir qu'un magnifique fichier de... silence.

Il faut, en effet, préciser la source d'enregistrement. Les différents mixers disponibles permettent de simplement réaliser cela. Cependant, dans le cas d'une machine réduite, sans interface console ou graphique, il devient délicat d'utiliser quelque chose comme **alsamixer**. Heureusement, les outils utilisateur d'ALSA comprennent **amixer** :

```
% amixer sget Line,0
Simple mixer control 'Line',0
Capabilities: pvolume pswitch pswitch-joined cswitch
cswitch-exclusive
Capture exclusive group: 0
Playback channels: Front Left - Front Right
Capture channels: Front Left - Front Right
Limits: Playback 0 - 31
Front Left: Playback 0 [0%] [off] Capture [off]
Front Right: Playback 0 [0%] [off] Capture [off]
```

On voit ici clairement que ce périphérique, l'entrée **Line**, n'est pas actif pour l'enregistrement. On corrige alors le problème avec :

```
% amixer sset Line,0 0%,0% cap
```

Remarquez que nous en profitons pour descendre le volume à 0%, ce qui ne change en rien le volume d'enregistrement (c'est Line-in). Nous pouvons maintenant streamer joyeusement tout ce qui sort du tuner.

Le contenu du fichier spécifié, ici **metadata**, est un simple fichier texte UTF-8 (attention, IceS ne vérifie pas l'encodage des caractères). Le fichier utilise une ligne par tag :

```
artist=GLMF
title=In Unix We Trust
```

Les tags utilisables sont **artist**, **title**, **album**, **genre**, **track**, **year** et **comment**. Notez que les applications clientes n'affichent généralement que le contenu des tags **artist** et **title**.

Nous mettons ensuite à jour notre section **encode** :

```
<encode>
  <nominal-bitrate>64000</nominal-bitrate>
  <samplerate>48000</samplerate>
  <channels>2</channels>
</encode>
```

Et le tour est joué !

4 Mes auditeurs veulent du MP3 !

Hé oui ! Bien que le Ogg fasse doucement son petit bonhomme de chemin, il ne représente pas encore le format le plus utilisé, loin de là. D'autres formats concurrencent le tenant du titre, j'ai nommé, le MP3. C'est le cas du AAC, par exemple, qui arrive en force depuis quelque temps. Il n'est donc pas étonnant de voir des utilisateurs/auditeurs souhaiter du MP3 ou de l'AAC plutôt que du Ogg (je sais, « mauvais utilisateurs, changer d'utilisateurs », mais je vais être gentil cette fois-ci).

IceS2 ne permet pas d'encoder en MP3. Il faudrait pour cela se tourner vers IceS0, mais nous avons une meilleure solution : DarkIce. Derrière ce petit nom se cache un véritable petit bijou. Ce streamer est capable d'encoder en Ogg, MP2, MP3 et AAC une source OSS, ALSA, Jack et même uLaw via un port série. Mieux encore, il est en mesure de streamer tout ce petit monde vers des serveurs Icecast, Icecast2, ShoutCast et Darwin Streaming Server (Apple). Pour jouer avec tous ces formats, vous devrez installer le paquet **darkice-full**.

La configuration du streamer est relativement simple. Voici un premier exemple. Nous commençons par les éléments de configuration généraux :

```
[general]
duration      = 0
bufferSecs    = 1
reconnect     = yes
```

duration détermine la durée de la transmission. 0 indique ici une diffusion continue. **bufferSecs** permet de spécifier un tampon en secondes et **reconnect** permet de se reconnecter automatiquement en cas de problème ou d'interruption de liaison. Nous embrayons directement sur la partie consacrée au périphérique audio.

```
[input]
device        = plug:dsnoop
sampleRate    = 22050
bitsPerSample = 16
channel       = 2
```

On retrouve les éléments de configuration classiques avec le périphérique (*plugin* ALSA Dsnoop), la fréquence d'échantillonnage, la taille des échantillons et le nombre de

canaux. Viennent ensuite le ou les profils de configuration spécifiques aux points de montage :

```
[icecast2-0]
bitrateMode    = cbr
format         = mp3
bitrate        = 32
server         = 127.0.0.1
port           = 8000
password       = coucou
mountPoint     = darkice.mp3
name           = Essai DarkIce MP3
description    = Stream DARKIce exemple MP3
url            = http://192.168.0.84
genre          = misc
public         = no
```

Les sections débutent par **[icecast-x]**, **[shoutcast-x]** ou **[icecast2-x]** où x est un numéro entre 0 et 7. Ce qui nous intéresse ici, c'est l'envoi vers un serveur Icecast2. Les différents paramètres intéressants sont :

- **format** : peut être choisi entre **vorbis**, **mp2**, **mp3** et **aac**.
- **bitrateMode** : peut être défini en **cbr** pour un bitrate fixe, **vbr** (Ogg, MP2, MP3) pour un bitrate variable et **abr** (MP2, MP3) pour un bitrate moyen.
- **bitrate** : n'est utile que dans le cas où **bitrateMode** = **cbr** et permet de fixer le bitrate en Kbps. Pour un autre mode, on utilisera **quality** avec une valeur comprise entre **0.0** et **1.0**.

Vous pouvez donc multiplier les sections et obtenir jusqu'à huit points de montage Icecast2 dans des formats différents pour une même source. De quoi réjouir vos utilisateurs qui peinent à choisir un format et un lecteur digne de ce nom. Attention cependant dans vos copier/coller à bien changer le nom des points de montage. L'erreur affichée par DarkIce n'est pas très explicite et parle d'un problème d'ouverture de bibliothèque. Il en va de même si votre serveur est configuré avec un nombre insuffisant de sources autorisées.

Le lancement du streamer se fait simplement via :

```
% darkice -c darkice.cfg
```

Il peut être utile de lancer la commande en tant que super utilisateur **root** afin de changer le mode de *scheduling* et d'éviter ainsi des « sauts » dans le flux audio.

5 Conclusion

Cet article fait l'impasse sur pas mal d'éléments de configuration avancée qui sortiraient du cadre de ce hors-série. Le fait de mixer liste de lecture et live peut, par exemple, être un élément important. Cependant, on parle alors de Webradio et non plus d'un simple service annexe d'un serveur dédié. Pour conclure, je ne saurais que trop vous conseiller l'écriture d'un script d'init basé sur **/etc/init.d/skeleton** afin de démarrer le streamer comme un service. Ces scripts ne sont, malheureusement, pas fournis avec les distributions Debian, mais cela fait un petit exercice intéressant après tout !



Denis Bodor

Rédacteur en chef de GLMF.
Utilisateur GNU/Linux depuis 1994.
Randonneur du jardin magique.

Chrootez vos connexions SSH

OpenSSH en version 5.0 a été mis à disposition il y a peu de temps. C'est l'occasion rêvée pour traiter un peu de la configuration avancée de ce shell distant sécurisé et en particulier de sa capacité à fonctionner de manière chrootée.

Un **chroot** est une technique d'emprisonnement de services. Plutôt que de les laisser fonctionner dans l'arborescence normale du système ayant ainsi accès à l'ensemble des fichiers qu'ils ont le droit de lire ou modifier, les services sont enfermés dans une prison constituée d'une fausse racine du système de fichiers. C'est la commande **chroot** qui permet de lancer une commande, quelle qu'elle soit, dans un environnement ainsi truqué.

L'intérêt principal d'un chroot est de limiter l'accès au système de fichiers. Ainsi un **shell** chrooté n'offrira à l'utilisateur

qu'un aperçu de la configuration et de l'arborescence. De la même manière, un service ou une application contenant une faille de sécurité ne pourra offrir qu'un accès limité à l'attaquant. Attention toutefois, un chroot n'est pas une solution parfaite. Il existe des techniques avancées permettant à un attaquant de sortir de sa prison en utilisant des failles dans le mécanisme de chroot ou dans le système utilisé comme hôte. Un chroot apporte de la sécurité certes, mais pas la sécurité absolue (qui n'existe de toutes façons pas).

1

Construire un environnement pour le chroot

La commande **chroot** permet de très simplement se familiariser avec un environnement chrooté. Il faut tout d'abord construire l'environnement lui-même. Un grand nombre de solutions existent allant de la copie de fichiers à la main aux scripts et outils divers en passant par l'installation d'un système minimal.

La solution 100% Debian au problème de la construction d'un environnement chrooté consiste à utiliser l'outil **debootstrap**. Il s'agit d'un script destiné originellement à créer, dans un répertoire, une arborescence complète d'un système Debian GNU/Linux. On peut ainsi installer une distribution Debian depuis un système déjà existant. C'est le cas, par exemple, pour les systèmes embarqués x86 (ou autres) utilisant une carte de type CompactFlash. On monte le système de fichiers fraîchement initialisé et on utilise **debootstrap** pour créer la base du système. On ajoute ensuite un noyau et un **bootloader**, puis on utilise la carte sur la plateforme embarquée.

On peut rapidement avec **debootstrap** créer un système minimal dans lequel chrooter :

```
% sudo -s
% debootstrap --variant=minbase etch \
/mnt/six1/CHR http://ftp.debian.org/debian/
I: Retrieving Release
I: Retrieving Packages
I: Validating Packages
I: Resolving dependencies of required packages...
I: Resolving dependencies of base packages...
```

```
I: Checking component main on http://ftp.debian.org/debian...
I: Retrieving adduser
I: Validating adduser
[...]
```

debootstrap prend en argument une cible (profil de distribution), un répertoire de destination pour l'installation des composants et un chemin vers un miroir contenant les paquets Debian. Ce dernier argument étant, le plus souvent, un miroir officiel sur le Net peut également être un chemin (**file:///**) ou une liaison SSH (**ssh:///**). L'option **--variant=minbase** permet d'obtenir un système le plus minimal possible. Le résultat a cependant une taille relativement conséquente, quelques 120 Mo, mais est parfaitement fonctionnel :

```
% chroot CHR
root@raven:/# dpkg -l | wc -l
83
root@raven:/#
```

Nous avons ici un environnement composé de quelques 83 paquets qui ne sont pas tous forcément utiles, mais qu'il n'est pas possible de sérieusement désinstaller en raison des dépendances du système de base (oui **gnupg** n'est pas très utile dans un chroot). Le principal avantage de cette méthode est le fait de pouvoir simplement installer et supprimer des composants logiciels dans l'environnement chrooté avec **apt-get**.

2

OpenSSH 5.0

Comme vous le savez sans doute à force de lire ce magazine, je suis utilisateur de distributions Debian. Malheureusement, la version 5.0 d'OpenSSH n'existe pas encore en version empaquetée. Il n'est pas question, bien sûr, d'installer tout cela à la main dans **/usr/local** à grand coup de **./configure && make && make install** (c'est mal !). L'astuce consiste

donc à utiliser les sources du dernier paquet Debian et à les combiner avec les sources de la dernière version amont. Dans le dialecte Debian, une version amont est la version développée par les personnes du projet d'origine.

On commence donc par récupérer les sources Debian, non sans avoir au préalable installé les dépendances de compilation :


```
% apt-get build-dep ssh
% apt-get source openssh-server
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Nécessité de prendre 1210ko dans les sources.
Réception de: 1 http://ftp.fr.debian.org unstable/main openssh
1:4.7p1-10 (dsc) [1490B]
Réception de: 2 http://ftp.fr.debian.org unstable/main openssh
1:4.7p1-10 (tar) [1009kB]
Réception de: 3 http://ftp.fr.debian.org unstable/main openssh
1:4.7p1-10 (diff) [199kB]
1210ko réceptionnés en 2s (410ko/s)
dpkg-source: extraction de openssh dans openssh-4.7p1
dpkg-source: info: unpacking openssh_4.7p1.orig.tar.gz
dpkg-source: info: applying openssh_4.7p1-10.diff.gz
```

Puis les sources amont :

```
% wget ftp://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-5.0p1.tar.gz
% tar xfvz openssh-5.0p1.tar.gz
```

On copie ensuite les informations concernant la construction du paquet de l'ancienne version dans la nouvelle :

```
% cp -r openssh-4.7p1/debian openssh-5.0p1
```

Dès lors, nous avons des sources qui sont presque prêtes à être compilées pour obtenir un paquet Debian. Il faut cependant modifier quelques informations. En effet, en y regardant de plus près, on remarque que la version du paquet dépend de la version spécifiée dans le **changelog**.

On ajoute donc une entrée au début du fichier **openssh-5.0p1/debian/changelog** :

```
openssh (1:5.0p1-0) unstable; urgency=low

* Major version

-- Denis Bodor <lefinnois@lefinnois.net> Thu, 3 Apr 2008 21:21:05 +0200
```

Autre problème à régler, celui des pages de manuel. Le script de construction **debian/rules** manipule des fichiers *manpage* qui n'existent pas. Il nous faut donc modifier ce script et commenter la ligne concernant le déplacement (**mv**) du fichier **authorized_keys.5**. Dès lors, nous pouvons lancer la construction :

```
% cd openssh-5.0p1
% dpkg-buildpackage -b -rfakeroot

% ls ../*.deb | cat
../openssh-client_5.0p1-0_i386.deb
../openssh-server_5.0p1-0_i386.deb
../ssh_5.0p1-0_all.deb
../ssh-askpass-gnome_5.0p1-0_i386.deb
../ssh-keyb5_5.0p1-0_all.deb
```

Il ne nous reste plus qu'à installer les paquets **openssh-client_5.0p1-0_i386.deb** et **openssh-server_5.0p1-0_i386.deb** avec **dpkg -i**. Notez au passage que, dans mes tests, j'ai été obligé de désinstaller les précédentes versions avec **apt-get remove**, avant l'installation des nouvelles.

3 Utilisation de la directive Match

Ce n'est pas une nouveauté de la version 5.0. Depuis plusieurs versions, le serveur OpenSSH est en mesure de réagir conditionnellement et d'adapter sa configuration en fonction de l'utilisateur, du groupe auquel il appartient, de l'hôte distant et de son adresse IP. Ainsi, on peut très simplement tester en modifiant **/etc/ssh/sshd_config** et en ajoutant en fin de fichier :

```
Match user lefinnois
ForceCommand /bin/date
```

On redémarre ensuite le serveur, puis on teste :

```
% ssh lefinnois@192.168.0.200
lefinnois@192.168.0.200's password:
mardi 20 mai 2008, 11:32:48 (UTC+0200)
Connection to 192.168.0.200 closed.
```

La directive **ForceCommand**, comme son nom l'indique permet de forcer le lancement d'une commande quand bien même l'utilisateur en aurait spécifié une autre. La connexion sous l'identité d'un autre utilisateur donnera, bien entendu, accès à un shell.

Passons à ce qui nous intéresse dans cet article, un shell chrooté pour l'utilisateur **lefinnois**. La section magique à ajouter dans votre **/etc/ssh/sshd_config** est la suivante :

```
Match user lefinnois
ChrootDirectory /var/JAIL_%u
```

Nous préparons ensuite l'environnement chrooté dans **/var/JAIL_lefinnois**. **%u** sera automatiquement développé en le nom d'utilisateur. Il nous faut également faire quelques ajustements dans l'environnement :

- Nous avons un utilisateur **lefinnois** existant déjà dans le système, ainsi qu'un utilisateur **denis** avec respectivement les UID 1001 et 1000 :

```
% id lefinnois
uid=1001(lefinnois) gid=1001(lefinnois)
groupes=1001(lefinnois)
```

- Nous chrootons dans l'environnement :

```
% sudo -s
% chroot /var/JAIL_lefinnois
```

- Le système minimal n'intègre pas l'utilitaire **adduser**. Nous l'installons donc via **apt-get** dans l'environnement :

```
$ apt-get update
$ apt-get install adduser
```

- Nous pouvons maintenant créer le groupe 1001 nommé **lefinnois** :

```
$ addgroup --gid 1001 lefinnois
Adding group 'lefinnois' (GID 1001) ...
Done.
```

- Puis l'utilisateur avec l'UID adéquate :

```
$ adduser --uid 1001 --gid 1001 lefinnois
Adding user 'lefinnois' ...
Adding new user 'lefinnois' (1001) with group 'lefinnois' ...
Creating home directory '/home/lefinnois' ...
Copying files from '/etc/skel' ...
```

- Toujours dans l'environnement chrooté, nous vérifions :

```
$ id lefinnois
uid=1001(lefinnois) gid=1001(lefinnois) groupes=1001(lefinnois)
```

- Tout fonctionne, nous pouvons sortir :


```
% exit
```

Il est important de respecter la correspondance entre les UID/GID sur le système réel et sur l'environnement chrooté pour éviter toute confusion. Si nous avons simplement créé l'utilisateur **lefinnois** dans l'environnement, celui-ci se serait vu automatiquement attribuer l'UID 1000, ce qui ne correspond en rien à la réalité et aurait introduit une confusion dans l'installation. Nous finissons en copiant quelques fichiers bien utiles du système réel dans l'environnement :

```
% cp /etc/hostname /etc/resolv.conf \
/etc/hosts JAIL_lefinnois/etc/
```

Après redémarrage du serveur OpenSSH, nous pouvons vérifier :

```
% ssh lefinnois@192.168.0.200
lefinnois@192.168.0.200's password:
$ tail -n 3 /etc/passwd
clamav:x:106:105::/var/lib/clamav:/bin/false
munin:x:107:106::/var/lib/munin:/bin/false
lefinnois:x:1001:1001:::/home/lefinnois:/bin/bash
$ ping localhost
-bash: ping: command not found
$ echo "echo JAILED" >> .bash_profile
```

```
$ exit
```

```
% ssh lefinnois@192.168.0.200
lefinnois@192.168.0.200's password:
JAILED
$exit
```

Dans l'ordre, nous nous connectons sous l'identité **lefinnois** et constatons dans le contenu de **/etc/passwd** que l'utilisateur **denis** est bien absent. La commande **ping** est introuvable. Enfin, nous modifions le **.bash_profile** pour y glisser un petit message qui s'affiche lors de la prochaine connexion. Nous sommes bien dans l'environnement chrooté.

À présent, quelques réflexions sur la méthode de construction de l'environnement. Pourquoi donc utiliser tellement d'espace ? La réponse tient en plusieurs points :

- Nous pouvons facilement ajouter/supprimer des outils via **apt-get** pour chaque utilisateur et leur environnement.
- Nous pouvons fournir bien plus qu'un simple shell. À l'aide de liens symboliques, nous pouvons utiliser des fichiers de configuration propres à chaque utilisateur. Il en va de même pour les répertoires de données. Idéal pour un serveur Web.
- Nous pouvons maintenir facilement le contenu des environnements et les réinstaller au besoin, très rapidement.

4 sFTP en toute simplicité

Cette fonctionnalité de chroot permet également de simplifier la mise en œuvre d'un service de FTP over SSL. Il faut tout d'abord ajouter le support sFTP interne d'OpenSSH en ajoutant une ligne **Subsystem sftp internal-sftp** dans votre **sshd_config** en lieu et place de **/usr/lib/openssh/sftp-server**.

On configure ensuite le service via **Match** :

```
Match group sftpgroup
  ChrootDirectory /home/%u
  AllowTcpForwarding no
  X11Forwarding no
  ForceCommand internal-sftp
```

Ici, plutôt que de chercher à « matcher » sur un utilisateur, on utilise un groupe spécifique pour les utilisateurs sFTP. Il nous faut ensuite préparer l'environnement, qui n'a rien à voir avec le précédent. Nous n'avons pas besoin de **debootstrap** :

```
% addgroup sftpgroup
% chown root.root /home/plop
% usermod -d / plop
% adduser plop sftpgroup
```

Le répertoire dans lequel OpenSSH chroot après la connexion doit appartenir au **root**. De plus, il change de répertoire et

se place dans le répertoire personnel de l'utilisateur après le chroot, relativement à la nouvelle racine. On définit donc comme répertoire personnel la racine, puis on ajoute l'utilisateur **plop** dans le groupe **sftpgroup**. On peut également faire appartenir le répertoire **/home/plop** au groupe **plop** avec **chown root.plop plop**, puis donner la permission d'écriture au groupe avec **chmod -R g+w /home/plop**. Ceci évite les problèmes de permission pour l'ajout et la suppression de fichiers via sFTP. La connexion se fait avec :

```
% sftp plop@192.168.0.200
Connecting to 192.168.0.200...
plop@192.168.0.200's password:*****
sftp> ls
sftp> put essai
Uploading jap to /essai
essai 100% 117 0.1KB/s 00:01
sftp> ls
essai
sftp>
```

Et sur le serveur :

```
# ls -la /home/plop
total 12
drwxrwxr-x 2 root plop 4096 mai 21 09:57 .
drwxr-xr-x 5 root root 4096 mai 21 09:48 ..
-rw-r--r-- 1 plop plop 117 mai 21 09:56 essai
```

5 Conclusion

Nous avons fait un petit tour des fonctionnalités de **chrootage** offertes par OpenSSH. Je vous laisse le soin d'étudier d'autres techniques pour créer un environnement chrooté. Un petit coup d'œil aux différents paquets disponibles dans Debian vous montrera qu'il existe un certain nombre de solutions plus ou moins automatisées. Bien entendu, la rédaction d'un script personnalisé est aussi une possibilité. Il n'existe pas de méthodes universelles. Enfin, on peut également se tourner vers des solutions plus complexes comme LVM2 et/ou des systèmes de fichiers montés en **loopback** (éventuellement chiffrés ou utilisant **unionFS**). Tout est possible.



Denis Bodor

Rédacteur en chef de GLMF.
Utilisateur GNU/Linux depuis 1994.
Randonneur du jardin magique.

DU NOUVEAU DANS LA PRESSE LINUX !

TOUJOURS CHEZ VOTRE MARCHAND DE JOURNAUX

LINUX PRATIQUE ESSENTIEL JUIN - JUILLET

NOUVEAU!! NOUVEAU!! NOUVEAU!! NOUVEAU!! NOUVEAU!!

N°2

JUIN - JUILLET 2008

LINUX

PRATIQUE

ESSENTIEL

L'ESSENTIEL DE L'ACTUALITÉ LINUX ET DES LOGICIELS LIBRES

TRUCS & ASTUCES

- Configurer Emacs
- Personnaliser la commande ls
- Serveur lent : pensez à changer vos dépôts !
- Téléchargez et convertissez les vidéos YouTube
- Changez votre résolution d'écran en un clic
- Des brosses supplémentaires pour GIMP
- Etc.

DOSSIER

INTERNET

L'ESPRIT LIBRE !

WEB, MAIL, CHAT, TÉLÉCHARGEMENT

Firefox vs Internet Explorer : lequel est le plus sûr ?... Migrer facilement d'Outlook vers Thunderbird... Messagerie instantanée : quels sont les risques ?... P2P, BitTorrent, Usenet, comment ça marche ?... MLDonkey, un client P2P multiprotocole...

 GNOME	 KDE	 XFCE
■ Que nous réservent Firefox 3 et Epiphany ? ■ Gnome et les applications mobiles	■ La prise en main à distance sur KDE ■ KDE 4.1 alpha 1, premier aperçu du futur bureau KDE	■ Découvrez les nombreux plug-ins à votre disposition pour personnaliser votre panel !

 L 17521-2-F 6.50 €

N°2

WEB, MAIL, CHAT, TÉLÉCHARGEMENT

Firefox vs Internet Explorer : lequel est le plus sûr ?... Migrer facilement d'Outlook vers Thunderbird... Messagerie instantanée : quels sont les risques ?... P2P, BitTorrent, Usenet, comment ça marche ?... MLDonkey, un client P2P multiprotocole...

et sur www.ed-diamond.com

Architecture Haute Disponibilité

Lighttpd est le petit serveur web qui monte. Simple, élégant et surtout très performant, de plus en plus de sites l'utilisent, spécialement ceux qui servent essentiellement des données statiques. Dans une architecture Java/J2EE, où les données sont essentiellement dynamiques, le serveur Apache est souvent utilisé en frontal. Le propos de cet article est d'étudier si Lighttpd ne pourrait pas remplacer avantageusement l'indien dans ce contexte.

1 Architecture J2EE « classique » avec Apache et JBoss

1.1 Répartition de charge

Avec la démocratisation d'Internet, le nombre d'utilisateurs simultanés d'une application Web a augmenté de manière exponentielle. Désormais, une application d'eCommerce doit pouvoir gérer jusqu'à plusieurs centaines de milliers d'utilisateurs simultanés. Comme les machines ont des limites de mémoire et de performance qui ne leur permettent pas d'obtenir de tels résultats, il est courant de mettre plusieurs serveurs en parallèle pour augmenter la capacité de réponse d'un système.

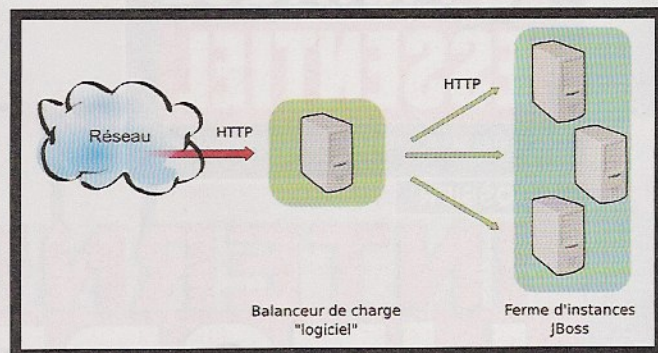
Dans la plupart des cas, on place en frontal un serveur Web, tel que Apache, pour assurer une bonne répartition de charge. Le serveur pourrait être aisément remplacé par un répartiteur de charges matériel (une « *appliance* »), mais ces derniers sont souvent très onéreux. Par souci d'économie, il est souvent pratique d'utiliser Apache à la place.

1.2 Simple répartiteur de charge

Dans ce contexte, Apache se contente de faire suivre la requête à chaque instance, en alternant entre chaque serveur JBoss, de manière à conserver une « charge » équilibrée entre les instances. On notera donc que si l'instance JBoss qui reçoit la requête est inaccessible (parce qu'elle est déjà trop chargée, par exemple), cette requête ne sera pas traitée et il est fort probable que l'utilisateur reçoive en retour une page d'erreur ! Bref, on a donc du *load balancing*, mais pas de reprise sur panne (*failover*).

Les sources d'enregistrement ALSA

Comme le serveur Apache est généralement peu chargé, on peut aussi lui demander de faire office de cache pour les instances JBoss et de servir les fichiers statiques, tels que les images, à la place du Tomcat embarqué dans JBoss. Tomcat devenant de plus en plus performant pour servir du contenu statique, cette pratique est devenue de moins en moins pertinente (surtout face à la relative complexité de l'opération et au faible gain de performance en retour).



1.3 Reprise sur panne

En conservant la même architecture, il est possible d'assurer une reprise sur panne. Dans notre première architecture, Apache et JBoss utilisaient simplement le protocole HTTP pour communiquer. Quand une requête parvient à Apache, il applique simplement son algorithme de répartition de charge et redirige la requête vers l'instance choisie. À la place de HTTP, il est possible d'utiliser un protocole dédié, nommé AJP. Dans un Apache, le module **modJK** implémente ce protocole et permet, entre autres donc, les fonctionnalités suivantes :

- Si l'instance JBoss choisie n'est pas en mesure de répondre, le protocole AJP permet à Apache d'en prendre compte et de rediriger la requête vers une autre instance, disponible.
- Si une instance JBoss doit être redémarrée ou simplement subir une opération de maintenance, le protocole permet de retirer une instance de la ferme. Les sessions entamées sur l'instance y seront achevées, mais aucune nouvelle requête ne sera redirigée vers l'instance. Une fois les sessions achevées, on peut redémarrer l'instance, sans que le moindre utilisateur ne se voit refuser l'accès à l'application.

Dans le cadre de cet article, nous allons donc essayer d'obtenir le même genre de fonctionnement, mais en utilisant non plus Apache mais Lighttpd.

avec Lighttpd et JBoss

2 Mise en place de la ferme d'instances JBoss

2.1 Démarrage et application de test

Avant tout, pour pouvoir mettre en place une telle architecture, il vous faut disposer d'une ferme d'instance JBoss, soit d'au moins 2 instances. Une fois la dernière version de la série 4.2 de JBoss téléchargée sur le site <http://labs.jboss.com/projects/download/> et décompressée, il vous suffit de démarrer une instance **default**, ainsi :

```
$ unzip jboss-4.2.X.zip
$ cd jboss-4.2.X
$ ./bin/run.sh

=====
JBoss Bootstrap Environment
JBoss HOME: /home/rpelisse/Desktop/jboss-4.2.2.GA/jboss-4.2.2.GA
JAVA: /opt/java/java/bin/java
JAVA_OPTS: -Dprogram.name=run.sh -server -Xms128m -Xmx512m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.net.preferIPv4Stack=true
CLASSPATH: /home/rpelisse/Desktop/jboss-4.2.2.GA/jboss-4.2.2.GA/bin/run.jar:/opt/java/java/lib/tools.jar
=====
14:12:38,178 INFO [Server] Starting JBoss (MX MicroKernel)...
14:12:38,180 INFO [Server] Release ID: JBoss [Trinity] 4.2.2.GA (build: SVNTag=JBoss_4_2_2_GA date=200710221139)
14:12:38,181 INFO [Server] Home Dir: /home/rpelisse/Desktop/jboss-4.2.2.GA/jboss-4.2.2.GA
14:12:38,226 INFO [Server] Home URL: file:/home/rpelisse/Desktop/jboss-4.2.2.GA/jboss-4.2.2.GA/
14:12:38,227 INFO [Server] Patch URL: null
14:12:38,227 INFO [Server] Server Name: default
...
14:13:26,581 INFO [Server] JBoss (MX MicroKernel) [4.2.2.GA (build: SVNTag=JBoss_4_2_2_GA date=200710221139)] Started in 48s:349ms
```

Le propos n'étant pas de déployer une application au sein de JBoss, on va utiliser une application déjà embarquée par le serveur d'application. Les Web Services étant très à la mode, nous allons donc utiliser l'application de gestion de la pile Web service de Jboss : <http://localhost:8080/jbossws/>. Ce choix étant fait, on constate qu'une instance JBoss occupe près de 128 Mo de mémoire par défaut ; comme on souhaite en faire tourner 2 sur le même poste, en plus d'un Lighttpd, on va commencer par réduire un peu la consommation, en termes de ressources de nos instances.

2.2 Configuration de l'instance JBoss

Un serveur d'application J2EE étant un arsenal de guerre, offrant l'ensemble des services définis dans la spécification, mais aussi un ensemble de services supplémentaires, spécifiques à JBoss, on va essayer de le configurer plus finement, l'objectif étant de réduire l'empreinte mémoire du serveur pour faciliter la mise en place de notre maquette sur une seule machine. Sans entrer dans les détails de l'architecture interne de JBoss AS, on peut retirer des services simplement en retirant des fichiers/répertoires du répertoire **deploy** de l'instance déployée. Pour le moment, on a démarré l'instance **default** qui se trouve dans le répertoire **server/default**. On ne va pas modifier cette instance, mais en créer une nouvelle :

```
$ cp -R server/default server/light
```

Dans cette nouvelle instance, on retire du répertoire **light/deploy**, les fichiers suivants :

- **bsh-deployer.xml**
- **client-deployer-service.xml**
- **ear-deployer.xml**
- **ejb3.deployer**
- **ejb3-interceptors-aop.xml**
- **ejb-deployer.xml**
- **jboss-aop-jdk50.deployer**
- **jboss-ha-local-jdbc.rar**
- **jboss-ha-xa-jdbc.rar**
- **jms**
- **jsr88-service.xml**
- **mail-ra.rar**
- **management**
- **quartz-ra.rar**

On pourrait peut être en retirer davantage, mais là c'est déjà pas suffisant :). On démarre cette nouvelle instance allégée :

```
./bin/run.sh -c light

=====
JBoss Bootstrap Environment
JBoss HOME: /home/rpelisse/Desktop/jboss-4.2.2.GA/jboss-4.2.2.GA
JAVA: /opt/java/java/bin/java
JAVA_OPTS: -Dprogram.name=run.sh -server -Xms128m -Xmx512m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.net.preferIPv4Stack=true
CLASSPATH: /home/rpelisse/Desktop/jboss-4.2.2.GA/jboss-4.2.2.GA/bin/run.jar:/opt/java/java/lib/tools.jar
=====
13:59:09,658 INFO [Server] Starting JBoss (MX MicroKernel)...
13:59:09,660 INFO [Server] Release ID: JBoss [Trinity] 4.2.2.GA (build: SVNTag=JBoss_4_2_2_GA date=200710221139)
13:59:09,661 INFO [Server] Home Dir: /home/rpelisse/Desktop/jboss-4.2.2.GA/jboss-4.2.2.GA
13:59:09,704 INFO [Server] Home URL: file:/home/rpelisse/Desktop/jboss-4.2.2.GA/jboss-4.2.2.GA/
13:59:09,705 INFO [Server] Patch URL: null
13:59:09,705 INFO [Server] Server Name: light
...
13:59:47,767 INFO [Server] JBoss (MX MicroKernel) [4.2.2.GA (build: SVNTag=JBoss_4_2_2_GA date=200710221139)] Started in 38s:57ms
```

Le démarrage de JBoss a déjà gagné 10s sur ma machine. En retirant simplement quelques fichiers, JBoss démarre 20% plus rapidement ! Ce n'est pas compliqué certes, mais il faut tout de même savoir ce que l'on fait et surtout, ce qui est souvent étrangement plus dur, de quoi on a vraiment besoin...

2.3 Configuration Mémoire

Comme indiqué au boot, la machine Java associée à JBoss se voit allouer 128 Mo de mémoire par défaut (-Xms128m) et peut, si cette limite est atteinte, consommer jusqu'à 512 Mo (-Xmx512). Pour un serveur de production, hébergeant une application chargée, ce paramétrage est justifié. Dans notre cas, nous n'avons pas besoin d'autant de mémoire et nous allons donc réduire, au minimum, l'empreinte mémoire de JBoss. Il suffit pour cela de modifier le fichier **bin/run.conf** qui définit les paramètres de lancement de JBoss AS :

```
JAVA_OPTS="-Xms32m -Xmx32m -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000"
```

On redémarre notre instance JBoss qui désormais n'occupe plus que 32 Mo de mémoire ! (Quoi ? Seulement ? Mais, on m'avait dit que les applications Java avaient besoin de beaucoup de mémoire ? On m'aurait menti ? :)).

2.4 Deuxième instance

Comme tout serveur J2EE, JBoss AS, même avec une configuration réduite comme ci-dessus, ouvre un certain nombre de ports associés à certains services (JNDI, HTTP/HTTPS,...). Si vous lancez une deuxième instance JBoss AS, sans changer la configuration, vous aurez des conflits de ports à régler. Il est possible de retrouver ces numéros de port dans les différents fichiers de configuration de JBoss AS, mais c'est long, pénible et certainement pas garanti de succès (vous pouvez en oublier).

La solution la plus simple est de disposer de plusieurs interfaces réseau, et de démarrer chaque instance de JBoss AS sur une carte réseau dédiée :

```
$ {JBoss_HOME}/bin/run.sh -c 194.3.4.97
```

Malheureusement, il n'est pas courant d'avoir 2 cartes réseau sur un simple poste de travail ou de développement. Heureusement, JBoss vient avec un service, nommée le *Binding manager*, qui résout, de manière très efficace (et pertinente), cette problématique.

En effet, le Binding Manager s'occupe de traduire l'ensemble des ports utilisés par défaut par votre instance d'une certaine valeur (par exemple 100). Ainsi, Tomcat, le container de *servlet* embarqué par JBoss AS, qui contient les applications Web, démarrera en utilisant le port 8180, et non plus 8080. Ainsi, l'URL d'accès à la seconde instance de notre application sera <http://localhost:8180/jbossws/>.

Comment mettre en place ce service ? Commençons simplement par créer une nouvelle instance :

```
$ cp -R server/light server/light2
```

Dans cette nouvelle instance, éditons le fichier **server/light2/conf/jboss-service.xml**. Il suffit de décommenter l'extrait de code XML suivant :

```
176
191
193 ports-01
194 ${jboss.home.url}/docs/examples/binding-manager/
   sample-bindings.xml
195
196 org.jboss.services.binding.XMLServicesStoreFactory
197
198
199
```

Et le tour est joué. Au démarrage, cette instance chargera le fichier **docs/example/sample-bindings.xml** qui lui décrit, pour chaque composant nécessitant un port d'écoute, la valeur du nouveau port. C'est simple, élégant, et surtout non intrusif (pas besoin de changer des dizaines de fichiers de configuration dans la deuxième instance). Vous pouvez maintenant ouvrir deux terminaux et lancer, sur votre machine, 2 instances de JBoss, qui occuperont chacune 32 Mo de mémoire. Notre ferme de serveurs applicatifs est prête. Passons maintenant au cœur de cet article : Lighttpd.

3 Mise en place de Lighttpd

3.1 Installation

Pour installer Lighttpd, vous pouvez le recompiler depuis ses sources ou simplement utiliser le système de packaging de votre distribution (**apt-get** pour les distributions à base de Debian ou **yum/rpm** pour les distributions utilisant **rpm**). Notez néanmoins que pour que le **mod_rewrite** de Lighttpd fonctionne, il vous faut disposer de **prce**. Pour l'exemple, supposons que l'installation se fasse sous Fedora :

```
# yum install pcre zlib lighttpd
```

Si vous utilisez déjà un serveur Web, pensez à le désactiver ou simplement à changer le port d'écoute de Lighttpd. Dans le cas de Fedora, le fichier de configuration, nommé **lighttpd.conf**, atterrit dans **/etc/lighttpd/**, et il est déjà pourvu d'une bonne configuration par défaut.

3.2 Mise en place du mode proxy

Lighttpd tourne. Reste donc à rediriger vers une instance JBoss, dans un premier temps, les requêtes HTTP. Pour cela, il suffit de démarrer Lighttpd avec le **mod_proxy** activé :

```
server.modules = ("mod_proxy")
```

Et définir ainsi la redirection :

```
proxy.server = ( "/jbossws/" => ( "localhost" => (
    "host" => "127.0.0.1", "port" => 8080 ) ) )
```

Détail important : notez bien l'usage de **127.0.0.1** et non de **localhost** pour indiquer l'hôte de redirection. En effet, il s'agit d'une limite de la version courante (1.4.18) : on ne peut pas préciser de *hostname* pour la redirection. Il faut utiliser des adresses IP. Cette limitation est clairement documentée dans la page du *wiki*, au sujet du **mod_proxy**.

Pour la petite histoire, au moment de la rédaction de cet article, la configuration par défaut, suggérée par l'installation Fedora, vous enduisait d'erreurs :

```
proxy.server = ( "/jbossws/" =>
    ( "localhost" =>
        (
            "host" => "localhost",
            "port" => 80
        )
    )
)
```


Voilà qui montre bien que, d'une part, la documentation de Lighttpd est plutôt bien faite et, d'autre part, que les logiciels de packaging facilitent grandement le travail, mais ne dispensent pas de la lecture de la documentation.

3.3 Répartition de charge

Lighttpd fait désormais office de proxy entre une de mes instances JBoss AS et les utilisateurs, ce qui est pratique (et peut servir à beaucoup de choses), mais ça ne remplit pas du tout notre cahier des charges, qui était, pour rappel : « *load balancing and failover* ». Alors, première étape, comment faire de la répartition de charge avec Lighttpd ?

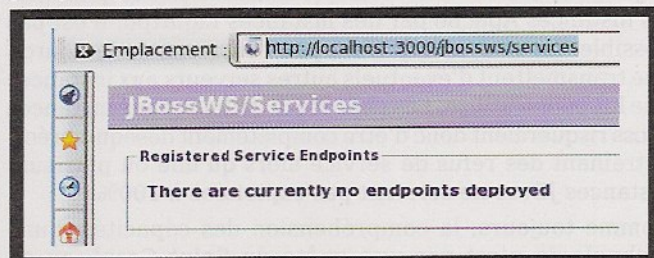
Pour mettre en place la répartition de charge, il suffit simplement d'ajouter des hôtes :

```
proxy.server = ( "/jbossws/" => ( ( "host" =>
"127.0.0.1", "port" => 8080 ), ( "host" => "127.0.0.1",
"port" => 8180 ) ) )
```

Pour voir sur quelle instance atterrissent nos requêtes, on peut simplement modifier le fichier `server/light/deploy/jbossws.sar/jbossws-context.war/index.html`. C'est un simple fichier HTML. Il suffit donc de rajouter un texte **Running on node 1**, par exemple. Lancez votre navigateur et accédez à l'URL <http://localhost/jbossws/> :



Il ne s'agit que d'une page statique servie par JBoss ; pour valider un peu plus, on peut cliquer sur **View a list of deployed services**. Là, le serveur d'application va lister l'ensemble des Services Web déployés, ce qui est un traitement un peu plus pertinent :



Maintenant, testons la reprise sur panne. Si vous arrêtez le nœud sur lequel vous êtes connecté, puis que vous rechargez la page, vous tombez sur une erreur de type 503. Mais, si un autre nœud est encore actif, il suffit de faire recharger une fois pour retomber sur une autre instance active. La reprise sur panne est assurée, moyennement tout de même la perte d'une requête.

En effet, avec le **mod_proxy**, Lighttpd n'est pas « conscient » de l'état des instances JBoss qu'il sert. Il ne découvre donc qu'une instance n'est plus en état de traiter des requêtes que lorsqu'il renvoie une page d'erreur au client. Néanmoins, le **mod_proxy** n'est pas si idiot : une fois qu'une instance a retourné une erreur, il la retire des nœuds actifs. Malheureusement, l'utilisateur aura reçu, au pire une page d'erreur, au mieux une page d'erreur personnalisée pour indiquer que le site est temporairement indisponible.

Et si jamais le nœud est de nouveau disponible ? Si l'instance JBoss a été redémarrée ou simplement que le pic de charge qu'elle subissait est fini et qu'elle peut accepter de nouveau des connexions, est-ce que **mod_proxy** intègre de nouveau l'instance ? Si vous arrêtez la deuxième instance et que vous rechargez la page, vous recevrez une erreur « 503 – Service Not Available ». Redémarrez une instance, rechargez la page et, de nouveau, Lighttpd redirige de nouveau les requêtes vers l'instance redémarrée.

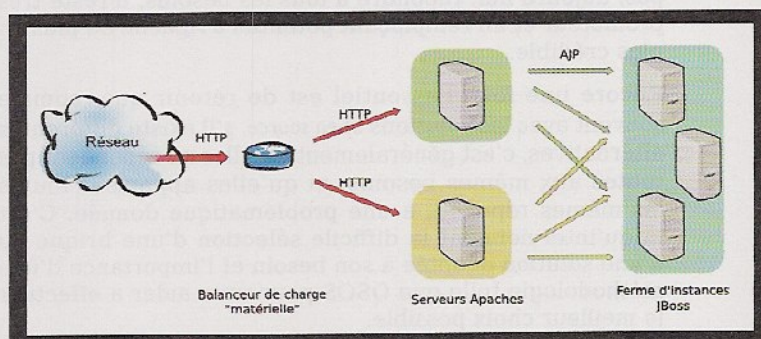
4 Bilan des courses ?

4.1 Apache et mod_jk ne servent donc à rien ?

Loin de là ! En effet, le protocole AJP, implémenté par le module **mod_jk** d'Apache2 offre un niveau de service, en termes de répartition de charge, supérieur à ce que peut offrir un simple **mod_proxy** comme le propose Lighttpd. Pour s'en rendre compte, il faut construire une architecture plus élaborée, où le répartiteur de charge logiciel lui-même devient redondant et assure donc une reprise sur panne. En effet, le protocole AJP permet à plusieurs serveurs web de répartir les requêtes sur une même ferme de JBoss. Avec **mod_jk**, l'architecture décrite ci-dessous permet donc d'assurer les points suivants :

- **Répartition de charge équilibrée entre chaque nœud**, grâce aux informations véhiculées par le protocole AJP, les deux instances Apache sont « conscientes » des charges transmises aux différentes instances JBoss et tiennent alors compte de leur répartition de charge.

- **Reprise sur panne**, si une des instances Apache est défaillante, l'autre pourra assurer la répartition de charges sur l'ensemble des requêtes mais, surtout, elle les répartira entre l'ensemble des instances JBoss. L'absence du second répartiteur n'exclura aucune instance JBoss de la ferme.



4.2

Quelle est la bonne solution alors ?

Aujourd'hui, une telle architecture, où l'on remplacerait les instances Apache par des instances Lighttpd, n'est pas possible. Le **mod_proxy** n'est pas conscient de la charge que transmettent d'éventuels autres serveurs aux instances que lui-même utilise. Les charges distribuées aux instances JBoss risqueraient donc d'être complètement déséquilibrées, entraînant des refus de service alors qu'une ou plusieurs instances JBoss ne seraient pas exploitées à 100%.

Comme toujours, la compréhension des capacités d'une technologie n'est pas une quête du Saint Graal, où on trouverait, in fine, une solution parfaite et absolue. L'objectif est plutôt de bien comprendre les avantages et les limites des composants de notre système pour bien les agencer. Il faut donc utiliser, à bonne escient, c'est-à-dire en tenant compte de ce qu'il sait faire et surtout de ce qu'il ne sait pas faire.

Lighttpd est un serveur très puissant dont cet article n'expose certainement pas, loin de là, toutes les finesses. S'il en ressort que Lighttpd ne peut pas remplacer Apache dans certains cas, il ressort aussi que sa simplicité et sa légèreté le placent comme une solution de remplacement très élégante à l'indien dans d'autres cas. Pour illustrer ce propos, nous allons retenir les deux points suivants :

- Lighttpd dispose d'une très faible empreinte mémoire et il est très simple à utiliser. C'est donc un outil simple et très peu coûteux à mettre en place.
- Le **mod_proxy** de Lighttpd permet de l'utiliser comme répartiteur de charge logiciel, mais l'implémentation ne permet pas d'utiliser plusieurs instances pour permettre une reprise sur panne.

Ces deux points ne sont certainement pas un résumé complet des capacités du serveur, mais au regard de notre cas d'étude (répartition de charges en frontal de serveurs web J2EE), ils synthétisent bien les caractéristiques de Lighttpd. Nous allons maintenant voir un cas où l'utilisation de Lighttpd, en

frontal de serveurs J2EE, offre un réel intérêt et se justifie probablement plus qu'Apache.

4.3

Utiliser au maximum son serveur

Vous souhaitez mettre en production une application J2EE que vous déployez sur JBoss. Vous avez à votre disposition une seule machine physique, une machine multiprocessus qui possède 4 Go et une carte réseau. Pour déployer votre application Java, vous avez choisi le serveur d'application JBoss et vous utilisez la machine Java 5 de sun.

Seulement, la machine Java ne peut utiliser plus de 2 Go de mémoire et, surtout, vous avez fait des tests de charge et vous savez que, même en période de charge extrême, une 1 Go de mémoire suffit amplement pour votre instance JBoss (surtout que vous l'avez déjà personnalisée un peu pour réduire son empreinte mémoire). Vous voilà donc avec un serveur loin d'être utilisé à 100%.

Entre le nombre de processeurs et la mémoire disponible, nous pourrions donc peut-être en profiter pour disposer de plusieurs instances et de pouvoir alors répondre à plus de charge. Mais nous ne disposons pas de répartiteur de charge physique et nous n'avons qu'une seule carte réseau. On a déjà vu plus haut comment démarrer plusieurs instances de JBoss sur la même machine et nous savons aussi comment utiliser Lighttpd comme répartiteur logiciel !

La faible empreinte mémoire de Lighttpd apportera ici une réelle plus value. En effet, avec 3 instances JBoss occupant chacune 1 Go de mémoire, il n'est pas sûr qu'il reste assez de mémoire pour le système et un Apache, et, de toute manière, plus performant sera notre répartiteur, mieux se sera. Néanmoins, il reste à valider notre architecture avec une série de tests de charge et de performance pour être sûr que la machine peut bien héberger 3 instances de JBoss AS. En effet, peut-être faudra-t-il se contenter de seulement 2 instances ou peut-être que, simplement, la ou les bases de données utilisées par notre application ne tiendront pas la charge.

5

Le mot de la fin

La prise en main de Lighttpd (nettement plus rapide, à mon sens, qu'Apache) ainsi que le côté performant du serveur en fait une solution parfaite dans bien des situations. Si, comme l'expose le thème de cet article, le serveur ne peut pas, aujourd'hui, répondre à tous les besoins, il reste très promoteur et un remplaçant potentiel à Apache de plus en plus crédible.

Encore une fois, l'essentiel est de retenir que, comme souvent avec les solutions *open source*, s'il existe différentes alternatives, c'est généralement qu'elles ne répondent pas toutes aux mêmes besoins, ni qu'elles apportent toutes les mêmes réponses à une problématique donnée. C'est là qu'interviennent la difficile sélection d'une brique ou d'une solution adaptée à son besoin et l'importance d'une méthodologie telle que QSOS pour vous aider à effectuer le meilleur choix possible.



Romain Pelisse

Romain Pelisse, Architecte Open Source chez AtosOrigin (JBoss, XRadars, PMD). Chargé de TD à l'ESME Sudria

Liens

- <http://wiki.jboss.org/wiki/Wiki.jsp?page=ConfiguringMultipleJBosInstancesOnOneMachine>
- <http://trac.lighttpd.net/trac/wiki/Docs:ModRewrite>
- <http://qsos.org/>

VoIP illimitée avec Asterisk

Parmi les lecteurs de GLMF, je suis sûr qu'une bonne partie a déjà entendu parler de la Voix sur IP ou VoIP dans le langage des lutins. Si ce n'est pas le cas, accrochez-vous bien, car vous êtes à quelques minutes de découvrir la liberté.

Pour faire de la publicité des discussions rincées de Gintonix, la VoIP est tout simplement le service de téléphonie offert par la majorité des FAI. De manière vulgarisée, votre BlahBox va s'authentifier via le net sur un serveur dit « registrar », ce qui vous permettra ensuite de passer des appels en France, en Europe et ailleurs pour des prix variant de 0 à x (x étant indéterminable, mais tendant souvent vers 0).

Cet article a pour but de vous guider, tout au long de la mise en place de l'architecture nécessaire, sur votre serveur dédié, pour récupérer votre ligne de téléphone fixe où que vous soyez dans le monde. Cette installation consistera au couplage d'OpenVPN, qui nous servira à ériger un réseau privé chiffré, et d'Asterisk, qui fera de notre serveur un commutateur téléphonique.

1 Scénario

Vous voilà parti sur la route de l'inconnu équipé bien entendu de votre joli portable. Vous êtes bien trop loin de chez vous, vous allumez votre portable pour y transférer quelques photos, et là, *surprise*, il y a un lien wifi ouvert qui rôde dans les parages ! Merveilleux ! Vous vous y connectez, vous lancez OpenVPN (qui se connecte sur votre serveur dédié) et votre *softphone* préféré (qui se connecte sur votre Asterisk dédié), et vous voilà téléporté chez vous ! Vous pouvez alors appeler votre famille & vos amis de votre ligne fixe française. Magique !

Viennent alors les questions. Mais, tout ce système, je pourrais le mettre sur ma machine à la maison et m'y connecter quand

je me déplace. Mais, qui a envie de laisser son ordinateur allumé 24h/24 pendant qu'il va gambader dans le monde à des centaines et des centaines de kilomètres de là ? Et si votre système prenait feu ? Et s'il y avait une coupure de courant ? Un dégât des eaux ? Une canicule qui fait surchauffer mon matériel ? Et j'en passe. La solution ultime qui, au-delà des problèmes de sécurité physique, règle aussi les problèmes de bande passante, est le serveur dédié.

Oui. Votre serveur dédié aura une bande passante bien supérieure (en débit montant et descendant), ainsi qu'une adresse IP fixe, et sera posé bien au frais dans une salle toute climatisée dans laquelle il se sentira si bien qu'il ne voudra pas se reposer une seule seconde. :)

2 Installation et configuration d'OpenVPN

Note

Étant donné que les *BSD et les Linux n'installent pas les ports ou *packages* selon la même arborescence, dans la suite de cette partie, j'emploierai la variable **\$PREFIX** qui prendra les valeurs suivantes :

Pour Linux : **PREFIX="/etc/openvpn"**

Pour FreeBSD : **PREFIX="/usr/local/etc/openvpn"**

Un des premiers bonheurs à mettre en place lorsqu'on configure un serveur dédié est de monter un réseau chiffré privé entre vos différentes machines à l'aide d'OpenVPN.

Cette partie de l'article décrit une configuration générique d'OpenVPN et peut donc être utile dans divers autres schémas d'utilisation que celui d'Asterisk.

Dans un premier temps, nous verrons la génération des clés et certificats liés à votre serveur et à vos clients. Puis, nous remplirons les fichiers de configurations de chacun.

Commençons tout d'abord par installer OpenVPN.

Sur une Debian (ou Ubuntu) :

```
sudo apt-get install openvpn
```

Sous FreeBSD :

```
cd /usr/ports/security/openvpn && make install clean
```

2.1

Installation & Génération des clés

La première étape sera de créer les dossiers, qui vous serviront par la suite pour y mettre vos fichiers de configurations.

Pour Linux et/ou FreeBSD : **mkdir \$PREFIX && mkdir \$PREFIX/work && mkdir \$PREFIX/keys.**

À ce moment précis, c'est avec grand plaisir que vous découvrirez qu'OpenVPN propose des scripts tous prêts qui vous serviront à générer les clés et certificats nécessaires à la sécurisation des échanges. Étant donnés les emplacements différents, une 2^{ème} variable est nécessaire : **\$SCRIPTS.**

Pour Debian (et Ubuntu) : **SCRIPTS="/usr/share/doc/openvpn/examples/easy-rsa/2.0".**

Pour FreeBSD : **SCRIPTS="/usr/local/share/doc/openvpn/easy-rsa/2.0".**

Copions alors ces fichiers dans le dossier **\$PREFIX/work** que nous avons créé précédemment :

```
cp -Rv $SCRIPTS/* $PREFIX/work/ && cd $PREFIX/work/
```

Pour faciliter la tâche qui suit, nous allons commencer par éditer le fichier **\$PREFIX/work/vars** qui sera lu lors des différentes générations. Voilà ce que vous avez à changer :

```
(...)
export EASY_RSA="$PREFIX"
(...)
export KEY_SIZE=2048 # <- pour les paranos
(...)
export KEY_COUNTRY=FR
export KEY_PROVINCE=NA
export KEY_CITY=PARIS
export KEY_ORG="Ghantoos OpenVPN"
export KEY_EMAIL="ghantoos@ghantoos.org"
```

Pour prendre en compte les nouvelles variables d'environnement définies dans le fichier **vars**, tapez :

source ./vars (Rappel : nous sommes là dans **\$PREFIX/work/**)

```
./clean-all (au cas où)
```

Commençons à tout calculer.

2.1.1

Certification Authority

Une *Certification Authority* est une paire de clés publique/privée servant à signer les autres clés publiques générées (auto-certification) :

./build-ca (vous y trouverez les variables éditées dans le fichier **//vars//**)

2.1.2

Certificat du serveur

Créons maintenant le certificat du serveur. La commande suivante ajoutera **server.crt** **raoul-work.csr** et **raoul-work.key** dans **\$PREFIX/work/keys/** :

```
./build-key-server server
```

Ce certificat est bien sûr signé par le CA généré au-dessus.

2.1.3

Construction des paramètres Diffie Hellman

Les paramètres Diffie Hellman nous fournissent un moyen de sécuriser la négociation initiale effectuée sur un lien non sécurisé. Cette commande peut durer quelques minutes :

```
./build-dh
```

2.1.4

La clé TLS

L'authentification TLS rajoute une signature HMAC à tous les packets de *handshake* afin de vérifier l'intégrité de ceux-ci :

```
openvpn --genkey --secret $PREFIX/keys/ta.key
```

2.1.5

Les clés des clients

Ce sera la dernière étape de préparation de notre sécurisation d'OpenVPN. À l'intention de chaque client qui se connectera sur votre VPN, vous allez devoir construire une clé :

```
./build-key pinpin
./build-key ghantoos
./build-key stariles
```

2.1.6

Envoi des clés aux clients

Afin d'accomplir cette tâche, on utilisera SCP. Pour le client **pinpin**, il vous faudra envoyer **ca.crt** **pinpin.crt** **pinpin.key** et **ta.key** :

```
cd $PREFIX/openvpn/keys/ && scp ca.crt pinpin.crt pinpin.key ta.key IP_de_pinpin:$PREFIX/
```

Maintenant que nous avons les clés en main, nous pouvons nous lancer dans la configuration de notre serveur et de nos clients.

2.2

Les fichiers de configuration d'OpenVPN

2.2.1

Configuration du serveur dédié

Pour configurer notre serveur, il nous faudra tenir compte de :

- nos clés et certificats générés dans la première partie de l'article ;
- l'IP et le port sur lequel le serveur VPN écoutera ;
- l'IP privée du serveur VPN ;

■ l'intervalle d'IP allouées aux clients.

J'ai choisi de vous montrer la configuration de réseau « routé » d'OpenVPN, l'autre choix étant le mode « bridgé » [2]

Les autres options seront commentées dans le fichier de configuration type que voici :

```
# Server Config
local xxx.xxx.xxx.xxx      # mettre l'IP de votre serveur
port 5050                  # port de messagerie instantanée,
                           # souvent ouvert : )

proto tcp
dev tun                    # mode routé
server 10.10.10.0 255.255.255.0 # IP du serveur 10.10.10.1

# Sécurité & Certificats
ca $PREFIX/keys/ca.crt
cert $PREFIX/keys/server.crt
key $PREFIX/keys/server.key
dh $PREFIX/keys/dh2048.pem
tls-auth $PREFIX/keys/ta.key 0

# Autre config
keepalive 10 120
comp-lzo                # compression LZ0
user nobody
group nogroup
persist-key # ne pas relire les clé en cas de perte de connectivité
persist-tun # ne pas éteindre le tun en cas de perte de connectivité
log openvpn.log
verb 3

# Clients configuration
client-config-dir ccd # dossier de configuration par client
client-to-client      # permet la communication entre clients

# Routes annoncées par les clients
route 192.168.50.0 255.255.255.0 # déclaration des route à
                                # venir (voir config client)
```

2.2.2 Configuration des clients

Maintenant que nous avons configuré notre serveur OpenVPN, nous allons configurer les différents clients qui auront accès à notre « réseau privé ». Cela pourra être votre ami en Chine, votre copain ou copine encore ailleurs ou bien tout simplement toutes vos différentes machines (bureau, maison, parents, etc.). Ainsi, vous aurez accès au service téléphonique de votre FAI où que vous soyez !

2.2.2.1 Chez le client

Pour configurer notre client, il nous faudra tenir compte :

- des clés et certificats générés dans la première partie de l'article ;
- de l'IP et du port sur lequel nous nous connecterons au serveur VPN.

Les autres options seront commentées dans le fichier de configuration type que voici :

```
# Config client
client
dev tun                    # réseau routé
proto tcp
remote xxx.xxx.xxx.xxx 5050 # IP et port du serveur

# Sécurité & Certificats
ca /home/ghantoos/.openvpn/ca.crt
cert /home/ghantoos/.openvpn/raoul-home.crt
```

```
key /home/ghantoos/.openvpn/raoul-home.key
tls-auth /home/ghantoos/.openvpn/ta.key 1
ns-cert-type server
```

```
# Autres options
comp-lzo
resolv-retry infinite # reconnections automatiques "infinie"
persist-key
persist-tun
verb 3
mute 20                # limitation des répétitions dans les logs
ping 10
```

2.2.2.2 Sur le serveur dédié

La configuration d'un client sur le serveur servira à :

- fixer son adresse IP privée ;
- lui permettre d'annoncer des routes qu'il connaît (vers son réseau local par exemple.) ;
- lui annoncer les routes privées d'autres clients OpenVPN.

Voici le fichier de configuration type. Celui-ci devra se placer dans le dossier **\$PREFIX/ccd** et portera le même nom que notre client :

```
# IP statique
# adresse IP fixe de pinpin: 10.10.10.20
ifconfig-push 10.10.10.20 10.10.10.21

# Les routes de mon réseau local
iroute 192.168.50.0 255.255.255.0

# Les routes du réseau local d'autres clients:
push "route 10.29.31.0 255.255.0.0"
```

2.3 Démarrage d'OpenVPN

Nous voilà donc en présence d'un serveur OpenVPN prêt à être utilisé lors de la deuxième partie de l'article.

Démarrons donc le serveur, puis le client.

Sur le serveur OpenVPN :

Linux :

```
/etc/init.d/openvpn start
```

FreeBSD :

```
/usr/local/etc/rc.d/openvpn start
```

Sur le client (je préfère démarrer le client dans un **screen**.. c'est bien plus pratique..) :

Linux :

```
/usr/bin/screen -dmS plop
/usr/bin/screen -x plop
/usr/sbin/openvpn /etc/openvpn/pinpin.conf
```

FreeBSD :

```
/usr/local/bin/screen -dmS plop
/usr/local/bin/screen -x plop
/usr/local/sbin/openvpn /usr/local/etc/openvpn/pinpin.conf
```

OpenVPN reste un monde à part entière dans lequel je vous conseille vivement de vous plonger. Le site officiel [1] est très riche en documentation et bien sûr un *man* OpenVPN pourra vous apporter des réponses et de nombreuses idées.

3

Installation et configuration d'Asterisk

3.1

Un peu de théorie ne fait jamais de mal

3.1.1

Architecture

Ainsi que je l'ai annoncé au début de l'article, nous allons bâtir notre infrastructure autour du protocole de VoIP SIP. Voici l'architecture que nous voudrions mettre en place (voir Figure 1) :

- Un OpenVPN précédemment configuré pour avoir votre réseau privé chiffré (sur serveur dédié).
- Une patte de l'Asterisk ira s'enregistrer auprès de votre FAI (sur serveur dédié).
- Une patte de l'Asterisk écoutera sur l'IP OpenVPN pour enregistrer vos utilisateurs (sur serveur dédié).
- Trois utilisateurs seront configurés et enregistrés (sur machines clientes).

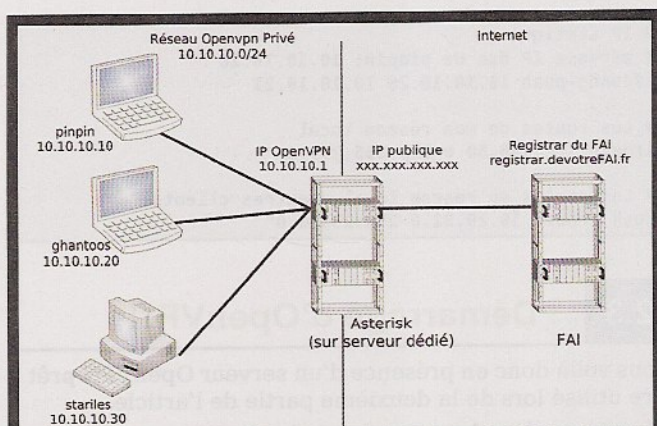


Figure 1 : Architecture

3.1.2

Enregistrement

Il est bien entendu qu'il faudra que vos utilisateurs s'enregistrent afin de passer ensuite leurs appels. Voyons comment cette authentification aura lieu.

Afin de ne pas envoyer votre mot de passe en clair à travers le réseau, la méthode d'authentification employée est le DIGEST over SIP (à l'image du DIGEST over HTTP) :

- L'utilisateur envoie un message REGISTER sans mot de passe.
- Le serveur lui renvoie une réponse 401 en lui donnant une clé de chiffrement.
- L'utilisateur fait alors un calcul « irréversible » à partir de la clé et de son mot de passe qu'il renvoie dans un nouveau REGISTER au serveur.
- Le serveur fait à son tour ce calcul, et le compare avec le contenu du message reçu. Si le résultat est identique, alors il vous renvoie un message 200 OK : vous êtes enregistrés.

Voici le *callflow* typique d'un enregistrement réussi :

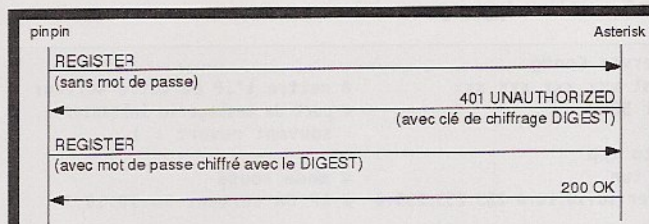


Figure 2 : Enregistrement

3.1.3

Appel sortant

En dehors du message REGISTER que nous avons vu, voici la liste des messages SIP les plus fréquemment rencontrés : INVITE, 100 TRYING, 180 RINGING, BYE et 200 OK. Leurs noms résument assez bien leur fonction lors de l'établissement d'un appel.

Voici le *callflow* d'un appel allant de l'intérieur de notre VPN vers un numéro national :

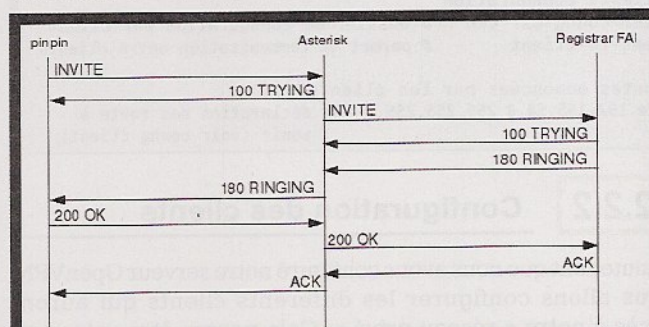


Figure 3 : Appel sortant

3.1.4

Appel entrant

Dans notre configuration, 3 utilisateurs seront enregistrés sur notre Asterisk, mais seuls **pinpin** et **ghantoos** pourront recevoir des appels entrants. Ci-dessous, le *callflow* d'un appel entrant récupéré par **ghantoos** :

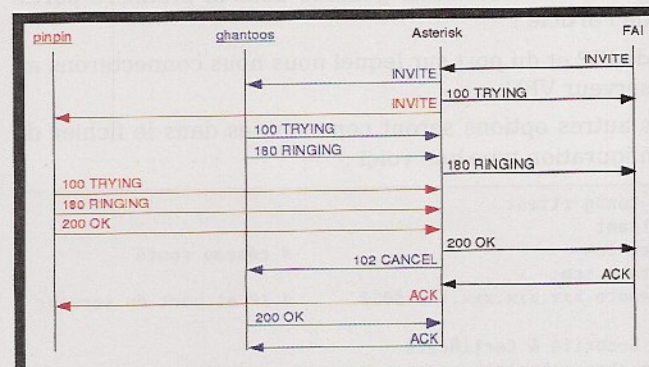


Figure 4 : Appel entrant

3.1.5 La problématique du NAT

Le problème du NAT dans la VoIP est un coup classique. En effet, le message SIP contient des informations relatives à l'adressage IP des clients. Quand A appelle B en passant par un serveur intermédiaire, le routage des messages de signalisation (SIP) se fait de manière classique. Jusqu'ici tout va bien, les téléphones sonnent, mais on n'entend rien. En fait, la destination (IP et port) des flux vocaux est donnée à l'intérieur du message SIP (plus précisément dans la SDP ou *Session Description Protocol*). Eh oui, vous voyez bien le souci.. Si B a une adresse publique et qu'il essaie de joindre A sur son adresse privée NATée (e. g. 10.10.10.10), la voix n'arrivera jamais à destination !

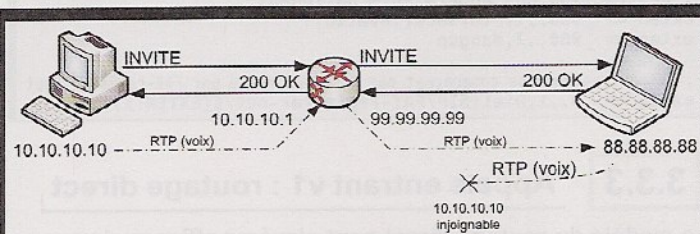


Figure 5 : Problématique NAT

Afin de pallier cette complication, il est impératif que le *registrar* soit capable d'éditer les messages SIP et plus particulièrement la SDP des messages pour y modifier les données relatives à l'adressage IP. Il faut donc remplacer toutes les occurrences de 10.10.10.10 au niveau de la couche applicative en 99.99.99.99. Heureusement, Asterisk est capable de faire ce travail là. Il nous faudra tout simplement déclarer notre réseau privé. Ainsi, quand Asterisk recevra un message venant de la plage d'IP en question, il saura que faire. Cette configuration est d'une simplicité déroutante ! Voici la directive à renseigner dans le fichier **sip.conf** :

```
localnet=10.10.10.0/255.255.255.0
```

3.2 Modélisation

Afin de pouvoir débiter la configuration de notre cher IPBX, il est nécessaire de poser les règles de numérotations que nous allons suivre.

Nous avons besoin de choisir :

- un préfixe de sortie (ou escape code) afin de contacter les numéros nationaux.
- les utilisateurs pouvant recevoir des appels de l'extérieur.
- les restrictions d'appel affectées aux utilisateurs.

Voici notre choix :

- préfixe de sortie : 9
- recevant les appels de l'extérieur : **pinpin** et **ghantoos**
- seul **stariles** aura des restrictions d'appel :
 - interdiction d'appeler sur les mobiles (906).
 - interdiction d'appeler à l'étranger (900).

3.3 Installation & Configuration

Note

Puisque que les *BSD et les Linux n'installent pas les ports ou *packages* selon la même arborescence, dans la suite de cette partie, j'emploierai la variable **\$PREFIX** qui prendra les valeurs suivantes :

Pour Linux : **PREFIX="/etc/asterisk"**

Pour FreeBSD : **PREFIX="/usr/local/etc/asterisk"**

Commençons tout d'abord par installer Asterisk.

Sur une Debian (ou Ubuntu) :

```
sudo apt-get install asterisk
```

Sous FreeBSD :

```
cd /usr/ports/net/asterisk && make install clean
```

3.3.1 Utilisateurs et registrar FAI

La configuration de l'enregistrement auprès de votre FAI et celle de vos utilisateurs sur votre Asterisk se fait dans le fichier **\$PREFIX/sip.conf** :

- Enregistrement sur le registrar du FAI :
 - Utilisateur (et numéro de téléphone fixe) : **0912345678**
 - Mot de passe : **votremotdepasse**
 - Domaine : **fai_registrar.com**
- Déclaration de l'utilisateur **pinpin** :
 - mot de passe : **lecodeestlibre**
 - nom de domaine : **asterisk** (**ndd** ou **realn** par défaut)
 - **context** : **monreseau** (la configuration des **context** sera expliquée par la suite)
- Déclaration de l'utilisateur **stariles** :
 - **context** : **monreseau-restreint**

Afin d'éviter d'écrire le mot de passe de **pinpin** en clair dans notre fichier de configuration, nous allons utiliser le paramètre **md5secret**.

Pour calculer le mot de passe en format **md5secret**, nous utiliserons la commande suivante :

Linux :

```
echo -n "utilisateur:nomd_de_domaine:motdepasse" | md5sum
```

FreeBSD :

```
echo -n "utilisateur:nomd_de_domaine:motdepasse" | md5
```

Pour **pinpin** :

```
echo -n "pinpin:asterisk:lecodeestlibre" | md5
f52697913d853a7216428779d4631a1e
```

Nous pouvons alors remplir notre fichier de configuration :


```
[general]
defaultexpiry=1800 ; timeout des enregistrements
dtmfmode=auto ; gestion des DTMF
qualify=yes ; vérification de présence avant l'envoi des appels

bindaddr=10.10.10.1 ; IP d'écoute de la patte privée d'Asterisk
bindport=5060 ; port d'écoute de la patte privée d'Asterisk
localnet=10.10.10.0/255.255.255.0 ; déclaration de notre réseau local

register => 0912345678:votremodepasse@fai_registrar.com ;
e.g. freephonie.net

; On refuse tous les codecs à part ceux cités ci-dessous
disallow=all
allow=ulaw
allow=alaw
allow=speex
allow=gsm

[FAI-registrar-out] ; point d'accès SIP pour les appels sortants
type=peer
host=fai_registrar.com ; e.g. freephonie.net
username=0912345678
fromuser=0912345678
secret=votremodepasse

[FAI-registrar-in] ; point de sortie des appels
type=peer ; déclaration du nœud avec votre FAI
context=fromFAI ; utile pour le plan de numérotation & routage
host=fai_registrar.com ; votre FAI

[pinpin]
type=friend ; "friend" autorise un dialogue bidirectionnel au contraire de "peer"
username=pinpin
md5secret=f52697913d853a7216428779d4631a1e
callerid=Pinpin (GCU-Squad)
host=dynamic
context=monreseau ; utile pour le plan de numérotation & routage

[ghantoos]
(...)
```

```
[stariles]
type=friend
username=stariles
md5secret=36f3b2748cea3f5714d849889bb4a0c7
callerid=Stariles
host=dynamic
context=monreseau-restreint
```

3.3.2 Numéros internes & appels sortants

Dans la partie modélisation, nous avons vu qu'à la différence de **pinpin** et **ghantoos**, **stariles** avait des restrictions d'appel. Afin de mettre en place ces restrictions, nous allons créer 2 groupes d'utilisateurs :

- **monreseau** : pour **pinpin** et **ghantoos** ;
- **monreseau-restreint** : pour **stariles**.

Cette variable se fixe lors de la déclaration de l'utilisateur à l'aide du champ **context**.

Nous allons maintenant déclarer les groupes, fixer les numéros de téléphone des utilisateurs et leurs règles de numérotation :

```
[monreseau]
exten => 10,1,Dial(SIP/pinpin) ; numero interne de pinpin: 10
```

```
exten => 11,1,Dial(SIP/ghantoos) ; numero interne de ghantoos: 11
exten => 12,1,Dial(SIP/stariles) ; numero interne de stariles: 12

; tous les numéros commençant par 9 (d'où le _9) seront routés
; par FAI-registrar-out
exten => _9.,1,Dial(SIP/FAI-registrar-out/${EXTEN:1})

[monreseau-restreint]
exten => 10,1,Dial(SIP/pinpin)
exten => 11,1,Dial(SIP/ghantoos)
exten => 12,1,Dial(SIP/stariles)

; Restriction des appels vers les mobiles
exten => _906.,1,Playback(invalid)
exten => _906.,2,Hangup

; Restriction des appels 0800
exten => _908.,1,Playback(invalid)
exten => _908.,2,Hangup

; Restriction des appels internationaux
exten => _900.,1,Playback(invalid)
exten => _900.,2,Hangup

; Les autres numéros commençant par 9 seront routé par FAI-registrar-out
exten => _9.,1,Dial(SIP/FAI-registrar-out/${EXTEN:1})
```

3.3.3 Appels entrant v1 : routage direct

Le modèle du routage direct peut s'avérer efficace si vous avez plusieurs téléphones dans la maison et/ou que vous voulez tout simplement faire sonner plusieurs téléphones en même temps lorsqu'un appel arrive (voir Figure 4). Lorsque votre Asterisk reçoit un appel entrant venant de votre FAI, il lui faudra router l'appel vers une ou plusieurs destinations. La configuration d'un tel scénario est assez simple. Dans **\$PREFIX/extension.conf**, il faudra rajouter :

```
[fromFAI] ; context utilisé par FAI-registrar-in dans sip.conf
exten => s,1,Dial(SIP/pinpin&SIP/ghantoos)
```

Ainsi, quand un appel arrive sur votre Asterisk, les téléphones de **pinpin** et **ghantoos** vont sonner en même temps. Le premier qui décroche aura gagné !

3.3.4 Appels entrants v2 : portail vocal

Imaginez que lorsqu'on vous appelle, vous donnez le choix à l'appelant de composer l'extension de l'utilisateur qu'il cherche à joindre. La première extension pourra être votre box à la maison, la deuxième celle du bureau, la troisième celle d'un ami, etc. Avec ceci, vous êtes sûr que tout le monde vous enverra lors de vos soirées branchées dans le bistrot du coin ! Regardons techniquement parlant comment mettre en place un tel service.

Le routage via un serveur vocal interactif est très intéressant. Voici ce que nous allons faire :

- enregistrer un message d'accueil ;
- générer un menu basique qui sera joué à la réception des appels entrants ;
- utiliser les touches [DTMF] du téléphone pour router l'appel vers le bon destinataire.

3.3.4.1 Enregistrement du message d'accueil

Il vous est possible d'enregistrer un fichier son en local en utilisant votre application préférée, puis de le convertir au format GSM. Il est possible d'utiliser l'application **sox** pour faire cette conversion [7] :


```
sox votre_message.wav -r 8000 -c 1 votre_message.gsm
```

Après cette conversion, le fichier devra être envoyé sur votre serveur dédié dans le dossier :

Linux (Debian) : **/usr/share/asterisk/sounds**

FreeBSD : **/usr/local/share/asterisk/sounds/**

Ceci étant dit, Asterisk nous offre une solution bien plus élégante. En effet, nous allons premièrement mettre en place un SVI (Serveur Vocal Interactif) basique qui nous permettra d'enregistrer les annonces que nous utiliserons par la suite. Il vous suffira alors d'appeler sur votre numéro fixe pour enregistrer votre message d'accueil après le bip et de taper la touche [#] à la fin de votre message pour le réécouter :

```
[fromFAI]
exten => s,1,Goto(menu_enregistrement,s,1)

[menu_enregistrement]
exten => s,1,Answer
exten => s,2,Wait(2)
exten => s,3,Background(hello-world)
exten => s,4,Record(mes-enregistrements%d:gsm)
exten => s,5,Wait(2)
exten => s,6,Playback(${RECORDED_FILE})
exten => s,7,Wait(2)
exten => s,8,Hangup
```

Vous pourrez alors trouver les fichiers enregistrés (**mes-enregistrements0**, **mes-enregistrements1**, etc.) dans le dossier **sounds** d'Asterisk (voir plus haut pour l'emplacement exact). Nous allons donc enregistrer 2 messages, le premier (**mes-enregistrements0**) donnera les choix qu'aura l'appelant en tombant sur notre SVI, le deuxième (**mes-enregistrements1**) préviendra l'utilisateur que la personne qu'il essaie de joindre n'est pas disponible. Vous voyez où je veux en venir. Quand, par exemple, l'appelant fera son choix de joindre **pinpin**, nous allons d'abord vérifier que **pinpin** est bien enregistré avant de transférer l'appel. Si ce dernier n'est pas joignable, un message sera joué, et l'utilisateur sera renvoyé au menu principal. :)

3.3.4.2 Notre SVI d'accueil

Maintenant que nous avons enregistré nos messages, nous allons mettre en place un SVI (Serveur Vocal Interactif) qui accueillera tous nos appels entrant en donnant le choix suivant à l'appelant :

- Tapez 1 pour joindre **pinpin**.
- Tapez 2 pour joindre **ghantoos**.
- Tapez 4 pour joindre nos trois lutins.
- BONUS : Tapez 5 pour joindre le premier utilisateur enregistré.

Notre menu principal devra alors ressembler à ce qui suit :

```
[fromFAI]
exten => s,1,Goto(menu_principal,s,1)

[menu_principal]
exten => s,1,Answer
exten => s,2,Wait,2
exten => s,3,SetMusicOnHold(default)
exten => s,4,Set(TIMEOUT(digit)=5)
exten => s,5,Set(TIMEOUT(response)=5)
exten => s,6,Background(mes-enregistrements0)
exten => s,7,Goto(s,2)
exten => 0,1,Directory(default|maison)

; On met le nom de l'utilisateur à joindre dans
```

```
; ${MYUSER} une variable globale (g), puis on
; envoie l'appel au testeur de "joignabilité": [test_presence]
exten => 1,1,Set(MYUSER="SIP/pinpin",g)
exten => 1,2,Goto(test_presence,s,1)
exten => 2,1,Set(MYUSER="SIP/ghantoos",g)
exten => 2,2,Goto(test_presence,s,1)

exten => 4,1,Dial(SIP/pinpin&SIP/ghantoos)

; BONUS: Dans ce cas là, c'est le premier utilisateur
présent qui sonnera
exten => 5,1,Set(MYUSER="SIP/pinpin&SIP/ghantoos",g)
exten => 5,2,Goto(test_presence,s,1)

[test_presence]
; Nous vérifions la présence de ${MYUSER}
exten => s,1,ChanIsAvail(${MYUSER})

; Un "CUT" doit être fait pour récupérer le nom de
l'utilisateur (sans son numéro
; de session) s'il est présent, dans le cas contraire,
la variable sera vide.
; Le "?5:3" signifie: si le "if" renvoie 1 aller à "s,5"
=> prévenir que l'utilisateur
; n'est pas joignable. Sinon aller à "s,3" => appeler
l'utilisateur en question
exten => s,2,GotoIf("${CUT(AVAILCHAN,,1)}" = "" )?5:3)

exten => s,3,Dial(${CUT(AVAILCHAN,,1)})
exten => s,4,Hangup

exten => s,5,Playback(mes-enregistrements1)
exten => s,6,Goto(mainmenu,s,1)
```

3.4 Démarrage d'Asterisk

Nous avons tout configuré sur notre serveur dédié afin de pouvoir recevoir des appels et appeler d'où qu'on soit via notre FAI. Il n'y aura plus aucune restriction géographique étant donné que la seule IP qui discutera avec votre FAI sera celle de votre serveur dédié. De votre côté, vous pourrez alors configurer autant d'utilisateurs que vous voulez en leur donnant les droits d'accès à votre réseau OpenVPN & Asterisk précédemment mis en place. Vive le serveur dédié et le Libre ! Mais attention à la facture quand même.

Quelques commandes utiles :

Linux :

- debug : **/usr/sbin/asterisk -vvvc**
- CLI : **/usr/sbin/asterisk -r**
- Default : **/etc/init.d/asterisk start**

FreeBSD :

Attention

Ajoutez **asterisk_enable="YES"** dans **/etc/rc.conf**.

- debug : **/usr/local/sbin/asterisk -vvvc**
- CLI : **/usr/local/sbin/asterisk -r**
- Default : **/usr/local/etc/rc.d/asterisk start**

4 Testons le tout

Pour tester le tout, j'ai choisi d'utiliser le softphone Twinkle qui m'a semblé approprié et que j'utilise pour ma consommation personnelle. :) Pour installer Twinkle sur votre machine (pas sur le serveur dédié bien sûr !):

Sur une Debian (ou Ubuntu):

```
sudo apt-get install twinkle
```

Sous FreeBSD:

```
cd /usr/ports/net/twinkle && make install clean
```

Enregistrons alors notre utilisateur **pinpin**:

- registrar : **10.10.10.1** (la patte « OpenVPN » de notre Asterisk);
- utilisateur : **pinpin**;
- domaine : **asterisk**;
- mot de passe : **lecodeestlibre**.

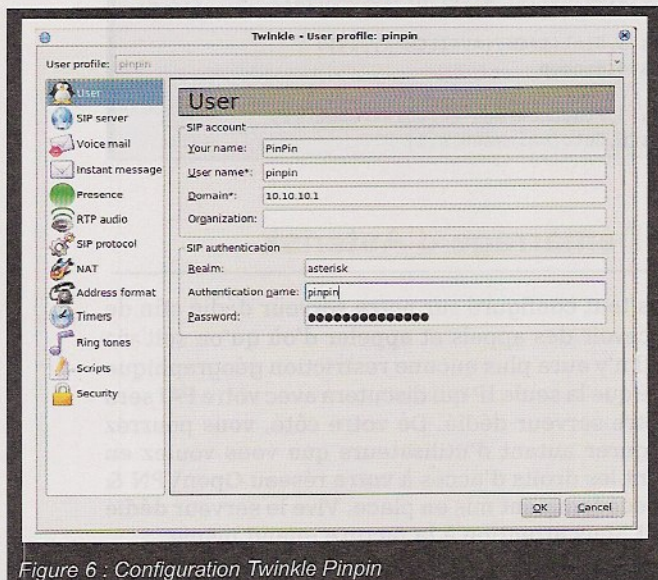


Figure 6 : Configuration Twinkle Pinpin

Voici les scénarios possibles, associés avec leur tcpdump vus dans wireshark, pour bien voir que non seulement c'est magique, mais c'est beau !

Note

Afin d'éclaircir la lisibilité des traces, j'ai modifié le fichier **/etc/hosts** pour avoir dans les champs source et destination des adresses résolues.

4.1 Pinpin -> Ghantoos

pinpin compose le numéro interne de **ghantoos** : [11] (voir le fichier **extension.conf**). Les messages SIP ainsi que le RTP (la voix) sont alors relayés par le serveur dédié :

Source	Destination	Protocol	Info
2 asterisk-openvpn	pinpin	SIP	Status: 100 Trying
3 asterisk-openvpn	ghantoos	SIP/SDP	Request: INVITE sip:ghantoos@10.10.10.20, with session description
4 ghantoos	asterisk-openvpn	SIP	Status: 100 Trying
5 ghantoos	asterisk-openvpn	SIP	Status: 180 Ringing
6 asterisk-openvpn	pinpin	SIP	Status: 180 Ringing
7 ghantoos	asterisk-openvpn	SIP/SDP	Status: 200 OK, with session description
8 asterisk-openvpn	ghantoos	SIP/SDP	Request: ACK sip:ghantoos@10.10.10.20
9 asterisk-openvpn	pinpin	SIP/SDP	Status: 200 OK, with session description
10 pinpin	asterisk-openvpn	SIP	Request: ACK sip:10.10.10.1
11 ghantoos	asterisk-openvpn	RTP	PT=ITU-T G.711 PCMU, SSRC=0x901DFA44, Seq=865, Time=207875093
12 asterisk-openvpn	pinpin	RTP	PT=ITU-T G.711 PCMU, SSRC=0x795F49, Seq=5099, Time=207875096
13 pinpin	asterisk-openvpn	RTP	PT=ITU-T G.711 PCMU, SSRC=0x85A00F5, Seq=12305, Time=208437605
14 asterisk-openvpn	ghantoos	RTP	PT=ITU-T G.711 PCMU, SSRC=0x2A04716, Seq=57735, Time=208437624
15 ghantoos	asterisk-openvpn	SIP	Request: BYE sip:pinpin@10.10.10.1
16 asterisk-openvpn	ghantoos	SIP	Status: 200 OK
17 asterisk-openvpn	pinpin	SIP	Request: BYE sip:pinpin@10.10.10.1
18 asterisk-openvpn	asterisk-openvpn	RTP	Status: 200 OK

Figure 7 : tcpdump Pinpin -> Ghantoos

4.2 Pinpin -> Mobile français

Pinpin compose le numéro d'un mobile en prenant garde de taper [9] : **906xxxxxxx**. Les messages SIP et le RTP sont alors relayés par notre serveur dédié au registrar et proxy RTP de notre FAI :

Source	Destination	Protocol	Info
2 asterisk-openvpn	pinpin	SIP	Status: 100 Trying
3 asterisk-public	FAI-registrar	SIP/SDP	Request: INVITE sip:05 @freemove.net, with session description
4 FAI-registrar	asterisk-public	SIP	Status: 100 Trying
5 FAI-registrar	asterisk-public	SIP/SDP	Status: 180 Ringing, with session description
6 asterisk-openvpn	pinpin	SIP	Status: 180 Ringing
7 FAI-registrar	asterisk-public	SIP/SDP	Status: 200 OK, with session description
8 asterisk-public	FAI-registrar	SIP	Request: ACK sip:
9 asterisk-openvpn	pinpin	SIP/SDP	Status: 200 OK, with session description
10 pinpin	asterisk-openvpn	SIP	Request: ACK sip:05 @10.10.10.1
11 pinpin	asterisk-openvpn	RTP	PT=ITU-T G.711 PCMU, SSRC=0x456A61E1, Seq=9304, Time=1058200778
12 asterisk-public	FAI-RTP-proxy	RTP	PT=ITU-T G.711 PCMA, SSRC=0x3A80662, Seq=8689, Time=1058200776
13 FAI-RTP-proxy	asterisk-public	RTP	PT=ITU-T G.711 PCMA, SSRC=0x480A037, Seq=52, Time=1058200776
14 asterisk-openvpn	pinpin	RTP	PT=ITU-T G.711 PCMA, SSRC=0x7650131F, Seq=52026, Time=99952
15 pinpin	asterisk-openvpn	SIP	Request: BYE sip:05 @10.10.10.1
16 asterisk-openvpn	pinpin	SIP	Status: 200 OK
17 asterisk-public	FAI-registrar	SIP	Request: BYE sip:
18 FAI-registrar	asterisk-public	SIP	Status: 200 OK

Figure 8 : tcpdump Pinpin -> mobile

4.3 Numéro publique -> Pinpin

Un numéro public appelle sur notre ligne de téléphone, puis appuie sur la touche [1] pour joindre **pinpin**.

Source	Destination	Protocol	Info
2 asterisk-public	FAI-registrar	SIP	Status: 100 Trying
3 asterisk-public	FAI-registrar	SIP/SDP	Status: 200 OK, with session description
4 FAI-registrar	asterisk-public	SIP	Request: ACK sip:05 @10.10.10.1
5 asterisk-openvpn	pinpin	SIP/SDP	Request: INVITE sip:pinpin@10.10.10.10, with session description
6 pinpin	asterisk-openvpn	SIP	Status: 100 Trying
7 pinpin	asterisk-openvpn	SIP	Status: 180 Ringing
8 pinpin	asterisk-openvpn	SIP/SDP	Status: 200 OK, with session description
9 asterisk-openvpn	pinpin	SIP	Request: ACK sip:pinpin@10.10.10.10
10 pinpin	asterisk-openvpn	RTP	PT=ITU-T G.711 PCMA, SSRC=0x277CE2D9, Seq=13608, Time=1206087255
11 asterisk-public	FAI-RTP-proxy	RTP	PT=ITU-T G.711 PCMA, SSRC=0x4458F443, Seq=4586, Time=200807288
12 FAI-RTP-proxy	asterisk-public	RTP	PT=ITU-T G.711 PCMA, SSRC=0x2801809, Seq=416, Time=2074135013
13 asterisk-openvpn	pinpin	RTP	PT=ITU-T G.711 PCMA, SSRC=0x555A2C00, Seq=30229, Time=2074135012
14 FAI-registrar	asterisk-public	SIP	Request: BYE sip:05 @10.10.10.1
15 asterisk-public	FAI-registrar	SIP	Status: 200 OK
16 asterisk-openvpn	pinpin	SIP	Request: BYE sip:pinpin@10.10.10.10
17 pinpin	asterisk-openvpn	SIP	Status: 200 OK

Figure 9 : Numéro public -> pinpin

4.4 Numéro public -> tous nos utilisateurs

Un numéro public appelle sur notre ligne de téléphone, puis appuie sur [4] pour faire sonner tous les utilisateurs actuellement enregistrés. Pinpin répond en premier.

Source	Destination	Protocol	Info
1 FAI-registrar	asterisk-public	SIP/SDP	Request: INVITE sip:s@10.10.10.1:5060;transport=udp, with session description
2 asterisk-public	FAI-registrar	SIP	Status: 100 Trying
3 asterisk-public	FAI-registrar	SIP/SDP	Status: 200 OK, with session description
4 FAI-registrar	asterisk-public	SIP	Request: ACK sip:s@10.10.10.1
5 asterisk-openvpn	pinpin	SIP/SDP	Request: INVITE sip:pinpin@10.10.10.10, with session description
6 asterisk-openvpn	ghantoos	SIP/SDP	Request: INVITE sip:ghantoos@10.10.10.20, with session description
7 ghantoos	asterisk-openvpn	SIP	Status: 100 Trying
8 pinpin	asterisk-openvpn	SIP	Status: 100 Trying
9 ghantoos	asterisk-openvpn	SIP	Status: 180 Ringing
10 pinpin	asterisk-openvpn	SIP	Status: 180 Ringing
11 pinpin	asterisk-openvpn	SIP/SDP	Status: 200 OK, with session description
12 asterisk-openvpn	pinpin	SIP	Request: ACK sip:pinpin@10.10.10.10
13 asterisk-openvpn	ghantoos	SIP	Request: CANCEL sip:ghantoos@10.10.10.20
14 ghantoos	asterisk-openvpn	SIP	Status: 200 OK
15 FAI-RTP-proxy	asterisk-public	RTP	PT=ITU-T G.711 PCMA, SSRC=0x6ECD9618, Seq=515, Time=2379108865
16 asterisk-public	FAI-RTP-proxy	RTP	PT=ITU-T G.711 PCMA, SSRC=0x1A328EE, Seq=9297, Time=1276201008
17 asterisk-openvpn	pinpin	RTP	PT=ITU-T G.711 PCMA, SSRC=0x392FC647, Seq=61759, Time=2379109024
18 FAI-RTP-proxy	asterisk-public	RTP	PT=ITU-T G.711 PCMA, SSRC=0x6ECD9618, Seq=517, Time=2379109185
19 pinpin	asterisk-openvpn	SIP	Request: BYE sip:06@10.10.10.1
20 asterisk-openvpn	pinpin	SIP	Status: 200 OK
21 asterisk-public	FAI-registrar	SIP	Request: BYE sip:
22 FAI-registrar	asterisk-public	SIP	Status: 200 OK

Le flux RTP (voix) a été redirigé par la présentation.

Figure 10 : tcpdump Numéro public → tout notre réseau

Voici ce que montre le CLI d'Asterisk pendant cette manœuvre :

```
ghantoos@monserveur-# asterisk -r
Asterisk 1.4.10, Copyright (C) 1999 - 2007 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=====
Connected to Asterisk 1.4.10 currently running on monserveur (pid = 15853)
Verbosity is at least 3
-- Executing [s@fromFAI:1] Goto("SIP/FAI-registrar-081ae468", "mainmenu|s|1") in new stack
-- Goto (mainmenu,s,1)
-- Executing [s@mainmenu:1] Answer("SIP/FAI-registrar-081ae468", "") in new stack
-- Executing [s@mainmenu:2] Wait("SIP/FAI-registrar-081ae468", "2") in new stack
-- Executing [s@mainmenu:3] SetMusicOnHold("SIP/FAI-registrar-081ae468", "default") in new stack
-- Executing [s@mainmenu:4] Set("SIP/FAI-registrar-081ae468", "TIMEOUT(digit)=5") in new stack
-- Digit timeout set to 5
-- Executing [s@mainmenu:5] Set("SIP/FAI-registrar-081ae468", "TIMEOUT(response)=5") in new stack
-- Response timeout set to 5
-- Executing [s@mainmenu:6] Background("SIP/FAI-registrar-081ae468", "mes-enregistrements0") in new stack
-- <SIP/FAI-registrar-081ae468> Playing 'mes-enregistrements0' (language 'en')
== CDR updated on SIP/FAI-registrar-081ae468
-- Executing [4@mainmenu:1] Dial("SIP/FAI-registrar-081ae468", "SIP/pinpin&SIP/ghantoos&SIP/stariles") in new stack
-- Called pinpin
-- Called ghantoos
-- SIP/pinpin-081b52b8 is ringing
-- SIP/ghantoos-081b9a58 is ringing
-- SIP/pinpin-081b52b8 answered SIP/FAI-registrar-081ae468
== Spawn extension (mainmenu, 4, 1) exited non-zero on 'SIP/FAI-registrar-081ae468'
monserveur*CLI>
```

On constate bien que les lignes du fichier **extension.conf**, plus précisément la partie **[menu_principal]**, s'exécutent tour à tour. :)

Ignace Mouzannar – ghantoos

Ingénieur VoIP chez Comverse/NetCentrex, constructeur d'équipements de Voix sur IP. Crooner officiel du jardin magique GCU-Squad.

5 Conclusion

L'époque où il fallait déboursier des milliers de francs pour mettre un place un réseau téléphonique privé/public est bel et bien révolue !

J'espère que cet article vous a été utile pour comprendre un peu plus comment fonctionnait la Voix sur IP, ainsi que la voie vers les diverses méthodes servant à monter une architecture saine et sécurisée afin d'en profiter où que vous soyez, moyennant une connexion au net, bien sûr. :)

Sur ce, à la prochaine,

Cheers !

Références

- [1] <http://openvpn.net/>
- [2] <http://openvpn.net/index.php/documentation/howto.html#vpntype>
- [3] http://www.asteriskguru.com/tutorials/extensions_conf.html
- [4] <http://kevinbusque.com/?p=17>
- [5] <http://www.voip-info.org/wiki-Asterisk+tips+IVR+menu>
- [6] <http://www.voip-info.org/wiki-Asterisk+cmd+Record>
- [7] <http://www.voip-info.org/wiki-Asterisk+cmd+Gotof>
- [8] <http://www.voip-info.org/wiki-Asterisk+cmd+ChanlsAvail>
- [9] <http://nerdvittles.com/index.php?p=158>

Entretien avec Odile Boutleux



Il y a beaucoup de demandes en ce moment, surtout pour des profils orientés open source...

GLMF : Pouvez-vous vous présenter rapidement pour nos lecteurs ?

J'ai 25 ans et je vis à Paris.

Cela m'a donné l'occasion de construire une bonne partie du parc informatique, Debian et Ubuntu essentiellement, et de gérer tout ce qui se présente dans la vie d'une entreprise au niveau technique, étant la seule personne du service informatique :)

GLMF : De quel type d'infrastructure êtes-vous responsable actuellement ?

« Aller parler à ses machines n'est pas le meilleur moyen d'avoir l'air normal :) »

Actuellement, et depuis peu, je fais partie, via une SSII, d'une équipe d'exploitation et support sur l'interconnexion internet d'une grosse infrastructure bancaire, en tant que « spécialiste » Xen, donc une grande partie système principalement Debian, et aussi pas mal de réseau sur matériels divers, comme Juniper, Cisco, Forti, F5...

Donc, bien évidemment, la mise en place et la maintenance des serveurs et des postes, mais aussi son amélioration au niveau projet en collaboration avec les développeurs, les relations avec les différents prestataires, internet, développement, matériel, et bien sûr le support des utilisateurs.

**GLMF : Comment en êtes-vous arrivé là ?
Pouvez-vous nous décrire rapidement votre parcours ?**

A la base, j'avais commencé par de la programmation, et j'ai eu la révélation en observant le travail des administrateurs système et réseau de mon IUT, ça a lancé ma vocation en quelque sorte.

En bagage universitaire, j'ai un master en systèmes répartis, surtout axé programmation. Mais, après la fin de mes études, j'ai plutôt cherché des postes en administration UNIX. J'ai donc commencé par un court passage en SSII, où j'ai été embauchée en temps qu'ingénieur système Solaris, que j'ai quitté pour un poste d'administrateur système et réseaux *open source*, dans une *start-up*, où je suis restée 18 mois.

Je suis partie récemment suite à la fermeture de la société et j'ai plutôt cherché un poste dans une structure plus importante, et surtout au sein d'une équipe. Afin de pouvoir acquérir de l'expérience un peu plus rapidement et surtout sur différents types de matériels, au niveau réseau notamment, que j'ai peu abordé dans mon poste précédent, j'ai privilégié la SSII.

GLMF : Pouvez-vous nous décrire l'une de vos journées types ?

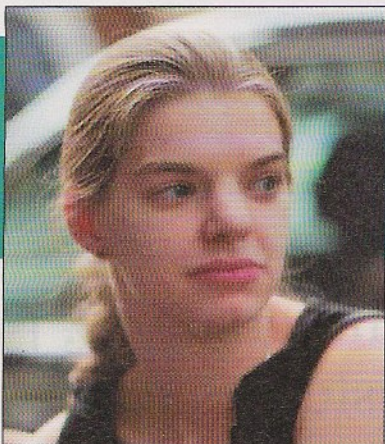
Dans ma précédente expérience, une journée type était difficilement définissable : étant responsable d'énormément de choses, ça consistant à gérer d'abord les urgences et problèmes sur la partie serveurs, ensuite le support utilisateurs, puis la maintenance normale du parc, puis, entre deux, avancer un peu sur les projets en cours...

Ses bookmarks

- www.google.com
- www.gcu.info

Sur sa table de nuit

- Time Management for System Administrators
Thomas A. Limoncelli
ISBN-13: 978-0596007836
- Perl Cookbook
Tom Christiansen, Nathan Torkington
ISBN-13: 978-0596003135



L'image de la femme sysadmin

est encore, de nos jours, quelque chose que beaucoup considèrent comme un mythe. C'est malheureux et revient à ne pas reconnaître que les compétences techniques et les qualités humaines d'un sysadmin n'ont rien à voir avec le sexe, l'âge ou les origines ethniques...

Entretien avec

Il s'agit plutôt d'une question de priorités ! En ce qui concerne les

horaires, j'essaie d'arriver tôt pour pouvoir faire avancer les projets en cours avant que les problèmes, pardon les utilisateurs, ne se présentent.

Actuellement, c'est beaucoup plus calme, en grande partie parce que je ne suis pas encore en mesure de pouvoir tout gérer. Et le fait d'être au sein d'une équipe permet de pouvoir se relayer pour gérer le support. Ainsi, les autres peuvent faire avancer les projets en cours plus rapidement.

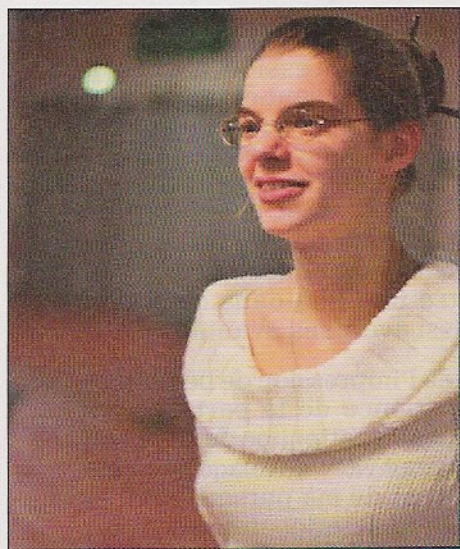
GLMF : Racontez-nous le pire cauchemar de votre carrière.

Je crois qu'un de mes pires cauchemars a été de gérer une coupure de courant, prévue, de 4 heures, et d'attraper un début de grippe la veille, avec aucun moyen de demander à quelqu'un d'autre pour me faire remplacer, étant toute seule à gérer l'infrastructure...

Donc, j'ai dû me traîner au travail à 7h le matin pour faire tous les préparatifs préalables à la coupure, somnoler sur mon bureau le temps que ça a duré, et ensuite régler tous les problèmes liés à la coupure après que le courant ait été rétabli, services non redémarrés, synchros à refaire, et autres...

Moi qui espérais pouvoir retourner dormir pas trop tard, je n'ai pu partir qu'en début de soirée. Ça a été une de mes pires journées.

« Et une femme sysadmin, c'est parfois considéré comme un mythe ou un fantasme, ce qui est amusant, mais un peu déprimant aussi ! »



A 25 ans, Odile Boutleux, alias Zborg, est spécialiste Xen dans une importante infrastructure bancaire.

Un parcours qui laisse sans doute rêveur beaucoup d'apprentis sysadmin, mais qui n'a rien d'exceptionnel pour quelqu'un qui sait marier créativité, passion et rigueur.

GLMF : Racontez-nous le meilleur moment, la plus grosse joie ou le plus gros soulagement que vous ayez eu.

Les meilleurs moments de ma carrière, ça a été les fois où une mise en place de serveur s'est passée comme prévu :)

GLMF : Le sysadmin, le vrai, est souvent caricaturé sous la forme de quelque chose entre le techno-fétichiste et le BOFH. Mythe ou réalité ?

Je dirais que ça dépend des moments. C'est très tentant parfois de faire le BOFH. Fort heureusement, je n'ai pas eu souvent à en arriver là !

Par contre, j'avoue être un peu techno-fétichiste, ce qui donne une image moyenne au niveau social dans l'entreprise. Aller parler à ses machines n'est pas le meilleur moyen d'avoir l'air normal :)

Entretien avec Odile Boutleux

GLMF : Quel est l'état du marché de l'emploi dans le secteur selon vous ? Existe-t-il vraiment une carence en sysadmins en ce moment ?

D'après ce que j'ai pu voir quand je cherchais un emploi récemment, il y a beaucoup de demandes en ce moment, surtout pour des profils orientés open source. D'après les chiffres qu'on m'a donnés, et ce que j'ai pu constater, il y a une vraie carence.

GLMF : Qu'est-ce qui différencie un bon sysadmin d'un mauvais ?

À mon avis, une bonne curiosité technique et une certaine débrouillardise sont importantes pour être un bon sysadmin, ainsi qu'un peu de rigueur et un peu de fainéantise :) Et quand on est en contact avec des utilisateurs, avoir un abord disponible et savoir écouter est important aussi à mon sens. Ça fait un peu tomber le mythe du BOFH...

GLMF : Quel type de relations s'installent-elles entre un sysadmin et ses utilisateurs ?

Les problèmes de communication sont-ils fréquents ? Êtes-vous vu comme un inquisiteur ou comme un personnage plus paternaliste réglant les petits problèmes de chacun ?

C'est important à mon avis que les utilisateurs puissent avoir confiance en un sysadmin. Ça pose parfois des problèmes, c'est parfois énervant aussi, car il arrive souvent que la vision qu'a un utilisateur d'un problème n'ait rien à voir avec l'approche qu'en a l'admin.

Il m'est souvent arrivé de me voir reprocher d'être trop technique dans une explication d'un problème, et inversement de devoir extorquer plus de détails sur des problèmes aussi vagues que « mon écran marche mal »...

J'ai eu la chance de pouvoir « éduquer » les utilisateurs à mon précédent poste, et je préfère qu'on vienne me voir dix fois pour un même

« En bagage universitaire, j'ai un master en systèmes répartis, surtout axé programmation. »

problème stupide plutôt que de passer à côté d'un problème sérieux parce qu'on n'a pas osé venir me voir. C'est quand même arrivé quelques fois, parce que parfois on n'osait pas venir me déranger pour quelque chose qui paraissait mineur et qui finalement ne l'était pas.

GLMF : Et la question que tout le monde se pose sans oser la dire : le fait d'être une femme dans le milieu de l'administration système/réseau est-il encore malheureusement un problème d'une manière ou d'une autre actuellement ?

La fameuse question :) Honnêtement, je ne pense pas que ce soit vraiment un problème. Le regard est différent, et il faut un peu de travail pour bien montrer qu'on est sysadmin et pas juste fille.

J'ai eu quelquefois l'impression d'avoir à prouver ma valeur, mais c'est peut-être dû à mon caractère plus qu'au fait d'être de sexe féminin, car je n'ai jamais eu de remarques négatives. Malheureusement, les femmes sont encore peu nombreuses dans ce milieu, et le stéréotype est masculin. Du coup, une fille incompetente peut vite être érigée en généralité, même si j'en ai rarement vu. Être une fille dans ce milieu d'hommes peut aussi avoir ses avantages.

Socialement, j'ai eu l'impression de bénéficier d'un peu plus de patience de la part de mes interlocuteurs, et je fais très attention de ne pas en abuser, car cela donnerait vite une image négative de mes compétences et de ma personnalité. Je n'ai jamais eu le cas d'un interlocuteur essayant de m'expliquer mon travail, heureusement ! Mais, c'est un numéro d'équilibriste un peu délicat. Et une femme sysadmin, c'est parfois considéré comme un mythe ou un fantôme, ce qui est amusant, mais un peu déprimant aussi...

Propos recueillis par : Denis Bodor

Émile Heitor



Le bon sysadmin ne travaille pas, il crée une œuvre

GLMF : Pouvez-vous vous présenter rapidement pour nos lecteurs ?

Bonjour, je suis Émile Heitor, mais plutôt iMil dans la communauté. Pour être franc, je ne reconnais plus vraiment mon prénom « officiel »... J'ai 34 ans, je suis marié et j'habite dans le 17ème arrondissement de Paris.

Deux ans plus tard, j'entrais chez un petit ISP de l'époque comme administrateur système et réseau junior. C'est là que je fis plus ample connaissance avec les systèmes BSD et que je créais GCU-Squad.

En 2000, après une évolution honorable dans cette première expérience professionnelle, j'ai eu la chance de rentrer chez l'un des

Ses bookmarks

- <http://www.feyrer.de/NetBSD/blog.html>
- <http://netbsd.gw.com/>
- <http://undeadly.org/>

Sur sa table de nuit

- The Magic Garden:
The internals of UNIX
System V Release 4
Berny Goodheart & James Cox
ISBN-13: 978-0130981387
- Design and Implementation
of the 4.4BSD Operating
system
Marshall Kirk McKusick,
Keith Bostic, Michael J.
Karels, John S. Quarterman
ISBN-13: 978-0201549799
- The Art of UNIX
programming
Eric S. Raymond
ISBN-13: 978-0131429017
- UNIX Network
programming vol.1 et 2
W. Richard Stevens, Bill
Fenner, Andrew M. Rudoff
ISBN-13: 978-0131411555
ISBN-13: 978-0130810816
- Systèmes d'exploitation
Andrew Tannenbaum
ISBN-13: 978-2744070020
- Réseaux
Andrew Tannenbaum
ISBN-13: 978-2744070655

« La migration a eu lieu de 6h à 7h du matin. À 8h, le million d'utilisateurs italiens s'authentifiait »

GLMF : De quel type d'infrastructure êtes-vous responsable actuellement ?

Je m'occupe des infrastructures de déploiement d'une solution industrielle de Voix sur IP. Le parc dont je suis directement responsable se limite à une centaine de machines (réelles ou virtuelles), mais le parc installé doit se situer autour de 2000 machines à travers le monde.

Ces serveurs, réels ou virtuels, font office de *clusters* de *softswitches* SIP/H323, de *gatekeepers*, de *SIP registrars* ou encore de bases d'abonnés.

GLMF : Comment en êtes-vous arrivé là ? Pouvez-vous nous décrire rapidement votre parcours ?

Ouuuh, vaste programme. Je vais tâcher d'être concis. J'ai découvert l'existence du projet GNU en 1994 sur Amiga. L'année suivante fût l'année de mon premier flirt avec GNU/Linux sur un 486DX4/100 et, par là même, mon premier accès à Internet à travers un modem 14400.

opérateurs en vue du moment, ISDnet (racheté puis tué par Cable & Wireless). Là, je me familiarisais avec les réseaux de grande envergure, plusieurs centaines de machines dispatchées dans plusieurs *datacenters*, là encore essentiellement sous FreeBSD.

Près de quatre ans dans cette société m'ont permis d'apprendre énormément sur les problématiques de haute disponibilité. Le courage de mon manager de l'époque et sa parfaite compréhension du monde libre m'ont autorisé à publier une bonne partie des travaux que j'ai réalisés là-bas, tant de la documentation, que des patchs pour des logiciels libres connus ou encore des softs complets que j'ai placés sous GPL.

À l'issue de cette formidable expérience, j'ai fait un rapide passage par la case management dans une *startup*, environnement dont je garde un souvenir tel que j'éviterai même d'en parler plus longuement.

Finalement, de retour dans le monde opérateur depuis début 2006, je suis aujourd'hui *technical manager* chez NetCentrex, constructeur d'équipements de Voix sur IP à destination d'opérateurs, équipements

Entretien avec Émile Heitor

que probablement certains d'entre les lecteurs traversent sans le savoir :) Pour finir sur mon parcours, j'ajouterai que j'ai toujours fait en sorte de ne travailler qu'avec des technologies libres, incluant mon poste de travail, et qu'il m'est fréquemment arrivé de refuser des postes alléchants pour cette raison : « Applique à toi-même », etc.

GLMF : Pouvez-vous nous décrire l'une de vos journées type?

Oh bah, comme tout le monde, YouTube, bashfr, icanhascheezburger... non, je plaisante. Vraiment. À mon poste actuel, je suis amené à faire beaucoup de *consulting*, essentiellement sur les aspects concernant le déploiement, l'exploitabilité, la *scalability*, soit de manière générale un terme qui rapporte 100 points au business loto : la *X-ability*. D'autre part, mon équipe est souvent sollicitée sur des problématiques d'exploitation, *hotfix*, chaos à l'autre bout de la planète ; il n'est pas rare que nous revêtions nos costumes de pompiers afin d'éteindre un feu, le plus souvent en relation avec une panne matérielle. Enfin, nous sommes responsables d'une plateforme dite « de référence », une sorte

« Je hais l'esprit startup »

de maquette ou « *template* » de la version courante des différents produits, construite avec l'aide précieuse des équipes R&D.



Voici l'image type du sysadmin : stressé, débordé mais toujours sur le coup.

Pour quiconque ne connaît pas ce métier et ses racines, le sysadmin passe parfois pour le BOFH : Bastard Operator From Hell, le personnage créé par S. Travaglia, la caricature du méchant administrateur réseau, démoniaque et sadique.

GLMF : Racontez-nous le pire cauchemar de votre carrière.

Le seul vrai cauchemar n'est pas technique. Justement. Comme je le racontais plus haut, je me suis essayé au management, le vrai, celui qui dégouline de caramel et où l'on doit aller se prostituer en vantant les mérites d'un produit dans lequel on ne croit pas soi-même. Cette expérience n'a servi qu'à confirmer deux axiomes :

- 1) Je hais l'« esprit startup ».
- 2) S'il arrive un jour, le temps où je m'éloignerai de la technique n'est pas venu.

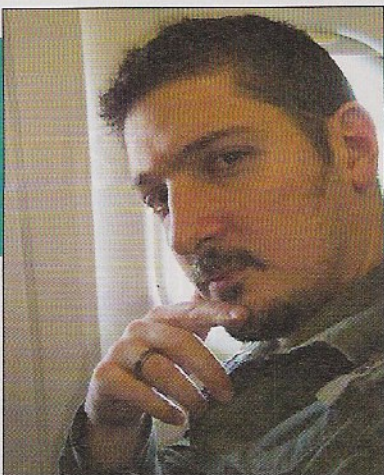
GMLF : Racontez-nous le meilleur moment, la plus grosse joie ou le plus gros soulagement que vous ayez eu.

C'était en 2002. La société Cable & Wireless nous demandait d'unifier les services des sociétés européennes qu'elle avait englouties, et de basculer l'authentification de ces services sur notre système hybride basé sur des serveurs OpenLDAP couplés à des passerelles LDAP/SQL dont j'étais l'auteur. C'était là leur baptême du feu et autant vous dire que mon *stress-o-mètre* était dans le rouge. La migration a eu lieu de 6h à 7h du matin. À 8h, le million d'utilisateurs italiens s'authentifiait sur nos bases. Jubilation !

GLMF : Le sysadmin, le vrai, est souvent caricaturé sous la forme de quelque chose entre le techno-fétichiste et le BOFH. Mythe ou réalité ?

Il n'y a pas de fumée sans feu, dit-on... Je pense que le sysadmin, le vrai, est un passionné, et que, en tant que tel, il ne comprend pas la paresse intellectuelle de l'utilisateur lambda ou *luser*. Il vit son monde (et je ne dis pas « dans son monde ») : son parc, son architecture, il l'a souvent construite « *from scratch* », et il en est fier.

Ce qu'il voit, ce ne sont pas forcément des services, mais une œuvre, un tableau, alors oui, il est parfois rude envers ceux qui sont incapables d'entrevoir son art, ce qui lui vaut l'étiquette de BOFH. Mais ce qu'il aimerait



Emile Heitor, plus connu sous le nickname d'iMil, est l'un des personnages clefs du monde du logiciel libre en France. Créateur du groupe GCU-Squad, il possède une vision très nette du milieu de l'OpenSource, vision qui est considérée par certains comme radicale.

Entretien avec

par-dessus tout, ce qui le rassurerait sur la nature humaine, c'est que l'utilisateur ne soit pas un élément passif, un

spectateur sans cerveau qui lui demande ce qu'il doit faire quand il a fini la page d'un livre.

GLMF : Quel est l'état du marché de l'emploi dans le secteur selon vous ? Existe-t-il vraiment une carence en sysadmins en ce moment ?

J'espérais voir venir cette question : le marché de l'emploi, du débutant à l'expert est tout simplement incroyablement propice ! J'ai récemment rencontré plusieurs cabinets de recrutement, plus par curiosité que par réelle volonté de changer d'entreprise, et ce que j'y ai entendu laisse sans voix.

« Non seulement les places dans l'univers libre pullulent, mais la demande est telle qu'on en arrive à faire la fine bouche ! »

Je cite le dernier recruteur que j'ai rencontré : « pour 3 postes dans le secteur, j'arrive péniblement à trouver un candidat qui le plus souvent accepte sans réelle conviction ».

Non seulement les places dans l'univers libre pullulent, mais la demande est telle qu'on en arrive à faire la fine bouche ! Vous l'aurez compris, la réponse à la seconde question est sans hésitation : oui !

Nous manquons cruellement de sysadmins. Un dernier exemple : GCU possède un espace privé dans lequel nous postons des offres d'emplois à destination des membres du groupe. Depuis environ un an, **aucun** des postes proposés, pourtant très intéressants, allant du codeur PHP au manager d'équipe en passant par l'administration système de haut niveau n'a trouvé de postulant.

GLMF : Qu'est-ce qui différencie un bon sysadmin d'un mauvais ?

Un seul mot : la passion.

GLMF : Quel type de relations s'installent-elles entre un sysadmin et ses utilisateurs ? Les problèmes de communication sont-ils fréquents ? Êtes-vous vu comme un inquisiteur ou comme un personnage plus paternaliste réglant les petits problèmes de chacun ?

Comme je l'évoquais plus haut, l'incompréhension, si elle n'est pas inévitable, est à mon sens liée à un facteur majeur : le bon sysadmin ne travaille pas, il crée une œuvre. Il ne se lève pas le matin plein de fatalisme en se disant : « Vivement ce soir que je puisse siffler un pack de bières devant le match France - Brésil » (désolé pour les amateurs, ce sont les seules équipes qui me viennent à l'esprit). En outre, en tant que bon sysadmin, il est perpétuellement obsédé par le prochain *upgrade* crucial qui rendra son architecture plus rapide/plus sûre/plus belle/plus « maintenable »/..., ce qui le rend irritable.

En effet, quoi de plus exaspérant, alors que vous êtes en train de *designer* une plateforme à l'épreuve des balles, qu'un utilisateur faisant irruption dans votre bureau et vous interrompant de la sorte : « petite question... » (je déteste cette entrée en matière) « ...c'est normal que XXXX marche pas ? ».

Alors oui, les problèmes de communication sont fréquents, de même qu'il doit exister des problèmes de communication entre un musicien et son auditoire. Il existe cependant un remède à ce problème de communication : la pédagogie. J'ai appris avec les années que certains utilisateurs ont le potentiel d'écouter, voire de comprendre. De fait, j'ai converti un certain nombre d'entre eux à l'utilisation de systèmes libres, et leur regard sur mon travail a radicalement changé. Seule ombre au tableau, cela prend du temps et de la patience. Beaucoup de patience.

Pour ce qui est de la troisième question, je suis de GCU monsieur l'inquisition, c'est mon métier.

Merci beaucoup pour cette interview, et longue vie à GLMF !

Propos recueillis par : Denis Bodor

MultiTail, un super-tail pour garder

Qui ne connaît pas tail, la commande permettant d'afficher la fin d'un fichier et de voir s'afficher en temps réel les éléments s'y ajoutant ? tail s'avère très pratique, mais possède des limites. Pour pallier cela, Folkert van Heusden a écrit MultiTail, un « tail on steroids », comme le décrit si bien son créateur.

Avant d'entrer dans le vif du sujet, voyons un peu ce qui existe de base dans un système Unix digne de ce nom. Les fichiers journaux comme tout fichier texte peuvent être très volumineux. Dans ce genre de situations, l'utilisation de commandes comme **cat**, **more** ou **less** s'avère très pénible. *Dumper* l'ensemble du fichier à l'écran est une perte de temps.

C'est pour cela qu'a été créé **tail**. Cette commande est très simple à utiliser. Il suffit de lui donner en argument un fichier pour qu'elle en affiche automatiquement les 10 dernières lignes :

```
% sudo tail -f /var/log/messages
[sudo] password for denis:
Apr 27 05:12:17 u840 -- MARK --
Apr 27 05:32:17 u840 -- MARK --
Apr 27 05:52:17 u840 -- MARK --
Apr 27 06:12:18 u840 -- MARK --
Apr 27 06:32:18 u840 -- MARK --
Apr 27 06:52:18 u840 -- MARK --
Apr 27 07:12:18 u840 -- MARK --
Apr 27 07:32:18 u840 -- MARK --
Apr 27 07:52:18 u840 -- MARK --
Apr 27 08:02:31 u840 syslogd 1.5
```

L'option **-f** permet d'afficher le contenu s'ajoutant au fur et à mesure dans le fichier. Il est également possible d'utiliser l'entrée standard avec, par exemple, **dmesg | tail**. Le nombre de lignes affichées peut être modifié via l'option **-n** suivie d'une valeur numérique. Deux autres options sont intéressantes pour les journaux. **--retry** permet de répéter la tentative d'ouverture du fichier spécifié après **-f** et **--max-unchanged-stats** permet de réouvrir le fichier après 5 itérations si sa taille n'a pas changé. Ces deux options permettent de traiter la rotation des journaux. En effet, à un moment donné, un système correctement configuré fera « tourner » les fichiers, renommant par exemple **messages** en **messages.0**, compressant **messages.0** en **messages.1.gz**, etc. C'est **logrotate** qui s'occupera de ce travail de manière récurrente.

tail est présent dans la plupart des systèmes, mais possède ses limites. Il n'est possible d'afficher le contenu que d'un seul fichier à la fois. **tail** ne possède pas de colorisation syntaxique et il faut jouer du **grep** et du **sed** pour mettre en exergue certaines parties.

1

MultiTail

Je ne vais pas lister ici les caractéristiques complètes de MultiTail, mais uniquement les plus intéressantes. Vous pourrez trouver la liste sur le site officiel. Sachez simplement que cet utilitaire permet de :

- coloriser les journaux ;
- afficher plusieurs journaux en même temps ;
- déclencher des actions en fonction du contenu des journaux ;
- fusionner plusieurs journaux dans une seule sortie.

Pour montrer la puissance de MultiTail face aux commandes classiques, voici un exemple clairement destiné à vous en mettre plein les yeux :

```
multitail -s 2 -P l -f /var/log/exim4/incoming/mainlog \
-e "ks36831 MailScanner" -ke "ks36831 MailScanner\[.\*\]:" \
-f /var/log/mail.log -f /var/log/exim4/mainlog \
-ke "^.* - " -f /var/log/lighttpd/lef-access.log
```

[illegible]

un œil ouvert sur vos journaux

Passons les options en revue :

■ **-s 2** permet de diviser l'écran en deux colonnes. Par défaut, lorsqu'on précise plusieurs fichiers à surveiller, l'écran est divisé horizontalement. Ce paramètre permet, en plus, de diviser verticalement. Nous précisons ici quatre fichiers, l'écran sera alors divisé verticalement et horizontalement.

■ **-P l** est l'équivalent de **-p l**, mais s'applique à toutes les fenêtres qui suivent. Ce paramètre permet de régler le retour à la ligne. Bien souvent, les lignes dans les fichiers journaux sont plus longues que la largeur de l'écran. De telles lignes se poursuivent donc sur la ligne suivante sauf si l'on précise **l** pour afficher le contenu des lignes en débutant par la gauche jusqu'au bord de la fenêtre. On peut également préciser **o** suivi d'une valeur numérique précisant une position sur la ligne.

■ **-e** est une option magique. Elle permet en effet de spécifier une expression régulière qui sélectionnera les lignes à afficher. Ici, nous nous intéressons aux occurrences de **ks36831 MailScanner**.

■ **-ke** utilise également une expression régulière. L'objectif est, cette fois, d'éliminer (*strip*) une partie du contenu des lignes. Comme nous sélectionnons uniquement les lignes

se rapportant à MailScanner dans le fichier **/var/log/mail.log**, inutile de saturer l'écran d'occurrences du nom d'hôte et du démon anti-SPAM. Idem pour le fichier journal de Lighttpd, **/var/log/lighttpd/lef-access.log**. Seules les requêtes nous intéressent.

Au final, on peut clairement affirmer que MultiTail remplace, de loin, le simple **tail** et permet, sans être un Nagios, une bonne solution pour garder un œil ouvert sur son serveur.



Denis Bodor

Rédacteur en chef de GLMF.
Utilisateur GNU/Linux depuis 1994.
Randonneur du jardin magique.

Lien

- Site officiel de MultiTail :
<http://www.vanheusden.com/multitail/>

Actuellement en kiosque

MISC N°37

**Votre serveur dédié
est en danger !**

**Apprenez à le protéger
des attaques les plus
efficaces :**

Les dénis de service

100% SÉCURITÉ INFORMATIQUE

37 MAI/JUIN 2008

MISC

Multi-System & Internet Security Cookbook

100 % SÉCURITÉ INFORMATIQUE

DOSSIER

DÉNI DE SERVICE

VOS SERVEURS EN LIGNE DE MIRE

DoS, DNS et botnets... un mélange explosif

DoS contre une plate-forme de voix sur IP

Évaluez votre résistance aux DoS

PROGRAMMATION

Développement d'une backdoor pour attaque ciblée (p. 77)

SYSTÈME

Les technologies par courants porteurs (CPL) sont-elles vraiment sécurisées ? (p. 54)

SCIENCE

Interceptions des communications vocales : techniques analogiques. (p. 66)

FICHE TECHNIQUE

À la découverte de ModSecurity 2

Apprenez à protéger vos applications WEB avec le pare-feu Apache (p. 60)

L 19018-37-F, 8,00 € - RD

Maîtriser le système de gestion

Il existe plusieurs méthodes pour garder un œil ouvert sur un serveur en production. La plus simple et celle utilisable par défaut et sans installation annexe reste la consultation des journaux d'activité. Les systèmes Unix disposent nativement d'un démon qui se charge de stocker et d'archiver les journaux : Syslog.

Les systèmes UNIX sont très bavards pour peu qu'on les y pousse. Ainsi, toutes les informations sur l'activité d'un système peuvent être connues, analysées, affichées et stockées. De cette manière, en cas d'attaque de la part d'un utilisateur malveillant ou d'un incident technique, on peut efficacement trouver la cause et la source du problème. Bien entendu, si l'attaque porte sur la machine elle-même, la première chose que tente de faire un pirate lorsqu'il va « *rooter* » le serveur est d'effacer ses traces. Il va donc essayer de retirer toute information le concernant ou concernant ses manipulations dans les journaux. L'étape suivante sera de dissimuler sa présence en temps réel en modifiant les binaires du système (**netstat**, **ps**, **who**, etc.). Enfin, il installera une *backdoor* lui permettant de revenir sur le serveur quand il lui plaira, même si l'administrateur corrige la faille exploitée, pour pénétrer dans le système.

Pour pallier ce problème, les journaux système peuvent être déportés sur une autre machine. Autre solution, si vous

ne disposez pas d'une autre machine, vous pouvez, sur le serveur lui-même, selon sa configuration matérielle, installer une machine virtuelle minimale (avec KVM par exemple). Celle-ci comprendra un serveur Syslog et uniquement un serveur Syslog (et un SSH bien sûr). Résultat : une machine virtuelle bien plus difficile à pénétrer que le système hôte. Bien entendu, l'attaquant pourra perturber le fonctionnement de ce système invité (*guest*), le stopper à souhait et même effacer l'image disque associée. Dans ces trois cas, il révélera sa présence et compromettra définitivement ses chances de revenir et d'utiliser le système. Ce sera déjà ça de gagner et, si vous avez mis en place une politique efficace de sauvegarde et de reprise sur incident, vous réglerez le problème rapidement.

Comprendre et exploiter pleinement Syslog vous permettra de conserver une trace sûre de toute activité licite ou non sur vos machines. Maîtriser la configuration des journaux système est toujours intéressant pour ne pas dire indispensable.

1

Syslog, système classique

Syslog est le système traditionnellement utilisé dans les Unix. Cette solution reposant sur les démons **syslogd** et **klogd**, bien que fonctionnelle, sera avantageusement remplacée par une nouvelle génération de gestion des journaux d'activité. Il est cependant important de garder à l'esprit la façon dont cela fonctionne, étant donné que le nouveau système repose énormément sur les principes déjà en œuvre et que, surtout, Syslog est installé par défaut avec la plupart des distributions.

Le premier démon, **syslogd**, est chargé de la gestion des requêtes qui lui sont envoyées par toutes les applications du système. Ce démon est une version évoluée de l'utilitaire Berkeley du même nom. Le second, **klogd**, est chargé de l'aspect noyau. Il gèrera tous les messages en provenance du programme le plus important du système : le *kernel* Linux.

syslogd utilise deux types d'informations pour qualifier un événement à prendre en charge. Dans un premier temps, celui-ci est désigné par sa priorité, son importance. Par ordre croissant :

- **none** : aucune information sur le niveau d'importance ;
- **debug** : simples informations de débogage ;
- **info** : messages d'informations simples ;

- **notice** : messages significatifs, mais informationnels ;
- **warning** : messages d'avertissement ;
- **error** : messages d'erreur ;
- **crit** : une situation critique se présente ;
- **alert** : messages d'alerte, une intervention humaine est demandée ;
- **emerg** : le système devient instable/inutilisable.

En plus de cela, **syslogd** organise les messages en fonction de leur type. On parle alors de « facilités » (*facilities*). Voici les facilités disponibles :

- **authpriv** (anciennement **auth**) pour tout ce qui traite de la sécurité et de l'authentification des services et des utilisateurs ;
- **cron** pour ce qui est en rapport avec l'ordonnancement des tâches (**cron** et **at**) ;
- **daemon** pour les messages provenant des démons en activité ;
- **kern** pour les messages du kernel (transitant par **klogd**) ;

des logs/journaux

- **local0** à **local7** pour les informations envoyées par des programmes qui permettent de choisir une facilité spécifique ;
- **lpr** pour ce qui est relatif au système d'impression ;
- **mail** pour ce qui concerne la messagerie électronique ;
- **news** pour les messages en rapport avec les services Usenet ;
- **syslog** pour les informations relatives au fonctionnement de **syslogd** lui-même ;
- **user** pour les informations génériques envoyées par les programmes « utilisateurs » ;
- **uucp** pour ce qui concerne le système UUCP ;
- **ftp** pour les messages concernant les services FTP ;
- **mark** est une facilité interne permettant d'ajouter des horodatages.

En fonction du niveau d'importance des messages et de la facilité concernée, on peut *dispatcher* les informations dans différents fichiers journaux ou « *piper* » les messages vers une console (**/dev/tty***). Les niveaux et les facilités sont standardisés. On les retrouve ainsi dans la RFC 3164, « *The BSD syslog Protocol* ».

La configuration du démon **syslogd** se fait via le fichier **/etc/syslog.conf**. Sa syntaxe est relativement rigide

et basique. Nous verrons plus loin que des systèmes de journalisation de l'activité système plus modernes proposent une configuration plus intuitive et souple. Voici un exemple de configuration pour **syslogd** :

```
auth,authpriv.*      /var/log/auth.log
*.*;auth,authpriv.none -/var/log/syslog
mail.err             |/dev/xconsole
*.emerg              @hote.domain.fr
```

Les définitions s'entendent ligne par ligne. Nous avons sur la gauche les critères de sélection sous la forme d'une chaîne composée du nom de la facilité et du niveau d'importance. C'est le sélecteur de messages. Sur la droite, nous avons la destination des messages ainsi sélectionnés. **syslogs** propose plusieurs destinations possibles :

- un fichier directement désigné ;
- un fichier qu'il ne sera pas nécessaire de synchroniser après écriture (signe -) ;
- un tube nommé (|) ;
- une redirection vers le syslogd d'un autre système (@).

Il est également possible de jongler avec les critères de sélection en utilisant des négations (!) ou encore en précisant un niveau d'importance unique (=). Tout ceci est d'une rigidité affligeante.

2

La nouvelle génération

Bien que la quasi-totalité des distributions utilisent le classique système Syslog, il existe des solutions alternatives efficaces, souples et puissantes. L'une d'elles est Syslog-ng, « ng » signifiant « *new generation* ». Les différences résident principalement au niveau de la syntaxe de configuration et de sa souplesse.

Les notions de « facilité » et de « niveau d'importance » sont toujours présentes et parfaitement compatibles. En revanche, les auteurs de Syslog-ng ont compris que cela n'était pas suffisant pour trier et sélectionner convenablement les messages dans les divers fichiers journaux. Ainsi, Syslog-ng intègre une notion supplémentaire, la notion de « filtre ». On peut filtrer les messages en fonction de la facilité et de l'importance, mais également via, par exemple, le nom du programme générant le message, l'hôte d'où il provient et, comble du bonheur, une expression régulière définie par l'utilisateur.

Pour comprendre l'architecture de la configuration, il faut envisager cette dernière sous la forme de quatre éléments distincts : les sources, les destinations, les filtres et les journaux.

2.1 Les sources de messages

Les messages dans un système UNIX, comme GNU/Linux, peuvent provenir de plusieurs sources. Nous avons tout d'abord le système de gestion des journaux lui-même. Cette

source est définie sous la désignation **internal**. Nous avons ensuite les messages système envoyés par les programmes qui utilisent la fonction **syslog()**. Ces messages sont récupérés par Syslog-ng via **/dev/log**. Enfin, nous avons les messages en provenance du noyau Linux qui seront récupérés dans **/proc/kmsg**.

Pour qu'une source soit prise en compte par Syslog-ng, elle doit être définie dans le fichier de configuration (généralement **/etc/syslog-ng/syslog-ng.conf**). Voici un exemple de définition :

```
source s_all {
    internal();
    unix-stream("/dev/log");
    file("/proc/kmsg" log_prefix("kernel: "));
};
```

Nous déclarons une source appelée **s_all** qui regroupe des informations en provenance des trois points cités plus haut (interne, messages de programme et du noyau). Cet exemple est tiré du fichier de configuration par défaut, tel qu'installé avec le paquet **syslog-ng** sur une distribution Debian Etch. La déclaration porte sur une source unique combinant plusieurs « émetteurs » de messages. Ce n'est pas nécessairement la méthode qui vous conviendra le mieux. Nous aurions tout aussi bien pu définir trois sources différentes :


```
source s_interne {
    internal();
}
source s_programmes {
    unix-stream("/dev/log");
}
source s_noyau {
    file("/proc/kmsg" log_prefix("kernel: "));
}
```

La définition des sources constitue la base de la configuration. À présent que nous savons prendre en compte l'origine des messages pour les faire traiter par Syslog-ng, nous pouvons nous pencher sur le reste de la configuration.

2.2 Les destinations

Les messages arrivent dans le système de traitement. Nous devons maintenant décider des diverses destinations possibles après tri et filtrage. Comme avec le système Syslog classique, la destination la plus fréquente est, sans le moindre doute, constituée de fichiers placés dans **/var/log**.

Voici à quoi ressemble la définition d'une telle destination dans le fichier de configuration :

```
destination df_mail {
    file("/var/log/mail.log");
};
```

Comprenez bien que nous ne faisons que définir une destination possible. Il s'agit d'une sorte de réceptacle pour les futurs messages. À ce stade, nous n'avons pas encore décidé quels messages transiteront vers cette destination. Cette première sentence définit un simple fichier comme destination. La directive (ou *driver*) **file** prend en paramètre le nom du fichier à utiliser. Nous définissons ainsi une destination possible appelée **df_mail** qui aura pour but de placer les messages dans **/var/log/mail.log**.

Vous vous en doutez, les fichiers ne constituent pas la seule option à notre disposition. Nous pouvons, par exemple, également définir une destination qui enverra les messages dans un tube ou vers un périphérique :

```
destination dp_xconsole {
    pipe("/dev/xconsole");
};
```

/dev/xconsole est le pseudo-périphérique utilisé pour la commande **xconsole**. Cette destination ne fait donc qu'envoyer ce qu'elle reçoit vers le pseudo-fichier dans **/dev**. Si vous vous posez la question, oui, effectivement, **/dev/lp** fonctionnera également. Voilà de quoi donner une touche très eighties à votre salle de serveur. *Krkrkrkr ;)*

Plus sérieusement, il peut être intéressant de retourner des messages sur une console non utilisée pour les logins (**getty**) avec :

```
destination dp_tty7 {
    pipe("/dev/tty7");
};
```

Convention de nommage

Avez-vous remarqué que dans tous les exemples précédents, j'ai pris soin d'appeler mes sources **s_quelque_chose** et mes destinations **dp_quelque_chose** ou **df_quelque_chose** ? **s_** pour « source » bien sûr, **dp_** pour « destination pipe » et **df_** pour « destination file ».

Cette manière de nommer les éléments de configuration n'est aucunement obligatoire, mais vous permettra de garder un fichier de configuration cohérent et clair. C'est une bonne pratique héritée d'habitudes de développeurs qui vous permettra de grandement vous simplifier votre vie d'administrateur.

Si vous souhaitez renvoyer les messages sur un terminal afin qu'un utilisateur responsable en prenne connaissance, vous pouvez être tenté d'utiliser l'entrée correspondante dans **/dev**. Avec Syslog-ng, on préférera la directive **usertty** qui prendra en argument le nom d'utilisateur ou un caractère joker. À la charge du système de retrouver le ou les terminaux de cet utilisateur :

```
destination du_denis {
    usertty("denis");
};
```

Ainsi, qu'il s'agisse d'Xterm, de session en console ou encore de sessions ouvertes avec GNU Screen via SSH, l'utilisateur **denis** recevra directement les messages que vous lui destinez.

Finissons avec une dernière définition de destination pour compléter au mieux la série :

```
destination df_facility_dot_err {
    file("/var/log/$FACILITY.err");
};
```

Ici, nous utilisons une caractéristique intéressante de Syslog-ng. Il s'agit du fait de pouvoir utiliser des variables pour nommer les fichiers de destination. Nous utilisons directement le nom de la facilité pour nommer le fichier dans **/var/log**. Ceci nous permet de créer une destination générique. Celle-ci pourra, par exemple, être utilisée pour stocker les messages d'erreur du système de mail ou d'authentification. Il devient inutile de définir une destination par facilité, cette simple configuration suffit à régler le problème.

2.3 Les filtres

Nous avons désigné les sources et les destinations, mais rien n'est en place pour l'instant. Il nous manque une étape avant de pouvoir composer les règles qui, finalement, placeront les messages de la source vers la destination. Cette étape ou élément prend la forme de filtres qu'il nous faut définir. La façon la plus simple de comprendre reste une série d'exemples. Tout comme avec les deux précédents éléments, on définira tout un jeu de filtres réutilisables. Voici quelques exemples :

```
filter f_mail {
    facility(mail);
};
```

Nous définissons un filtre nommé **f_mail** destiné à ne laisser passer que les messages concernant la facilité **mail**. La directive **facility** prend, tout simplement, en argument le nom d'une facilité. Nous pouvons procéder de même avec les niveaux d'importance des messages :

```
filter f_at_least_warn {
    level(warn..emerg);
};
```

Ce filtre est légèrement plus complexe. Il s'agit de conserver les messages ayant un niveau d'importance au minimum égal à **warn** et au maximum égal à **emerg**. Contrairement au système

Syslog classique, le comportement par défaut d'un filtre portant sur le niveau d'importance des messages est de sélectionner uniquement les messages du niveau spécifié (et non celui spécifié et supérieur). Si l'on souhaite traiter, comme ici, un niveau et plus, il faut penser à utiliser cette syntaxe (deux points).

```
filter f_messages {
    level(info,notice,warn) and not
    facility(auth,authpriv); };
```

Afin de trier au mieux les messages, il est possible de combiner plusieurs conditions. En utilisant des opérateurs logiques (**and**, **or**, **not**), on peut affiner une sélection. Le filtre présenté ici sélectionne les messages de niveau **info**, **notice** et **warn** qui ne concernent pas les facilités **auth** et **authpriv**. Comme son nom l'indique, c'est un filtre que nous destinons, finalement, au tri des messages à destination de **/dev/xconsole** :

```
filter f_xconsole {
    facility(daemon,mail)
    or level(debug,info,notice,warn)
    or (facility(news)
        and level(crit,err,notice));
};
```

Ce filtre démontre clairement la souplesse de Syslog-ng. Nous sommes très loin de la rigidité de Syslog. Il existe d'autres directives permettant de filtrer en fonction d'autres éléments. Citons par exemple **match** permettant d'utiliser des expressions régulières, **netmask** se basant sur l'adresse source du message

et vérifiant la correspondance avec un sous-réseau donné ou encore **program** pointant la source du message :

```
filter f_ssh_login_attempt {
    program("sshd.*")
    and match("(Failed|Accepted)");
};
```

Ce filtre utilise la directive **program** pour cibler le nom du démon émettant le message et combine cette condition avec **match** et une expression régulière. Nous cherchons donc à sélectionner les messages, en provenance du démon SSH en fonction, qui comprennent la chaîne **Failed** ou **Accepted**.

2.4 Les journaux

Nous avons nos trois briques de base : les sources, les destinations et les filtres. Il ne nous reste plus qu'à combiner tout cela pour créer nos règles Syslog-ng. Exemple :

```
log {
    source(s_all);
    filter(f_mail);
    destination(df_mail);
};
```

La directive **log** nous permet de connecter des sources avec des destinations via les filtres. Il est possible, même si ce n'est pas la majorité des cas, de spécifier dans la portée d'un **log** plusieurs sources, plusieurs filtres et plusieurs destinations.

On utilisera autant de portées **log** que nous souhaitons faire de connexions.

Attention, l'ordre des occurrences de **log** a son importance. Les messages seront traités dans l'ordre. Un message transitera par défaut au travers de toutes les occurrences de **log** sauf si deux couples source/filtre sont identiques. Par exemple, ceci ne fonctionnera pas :

```
log {
    source(s_all);
    filter(f_ssh_login_attempt);
    destination(df_denis);
};
log {
    source(s_all);
    filter(f_ssh_login_attempt);
    destination(df_denis);
};
```

Il vous faudra utiliser :

```
log {
    source(s_all);
    filter(f_ssh_login_attempt);
    destination(df_denis);
    destination(df_denis);
};
```

Et éventuellement, plus loin :

```
log {
    source(s_all);
    filter(f_auth);
    destination(df_denis);
};
```

Vous pourrez utiliser la directive **flags** afin de spécifier la manière dont le traitement se déroulera. Ainsi, si vous souhaitez qu'un **log** interrompe le traitement du message, il faudra ajouter **flags(final)** ; dans sa portée. Si un message correspond, toutes les occurrences suivantes de **log** seront ignorées.

3 Exporter les journaux

Comme dit plus haut, disposer d'un système de gestion des journaux efficace et performant est une chose importante. Cependant, en cas d'incident, ce système peut, lui-même, avoir subi des dommages. Il faut dédier une machine (physique ou virtuelle) qui accumulera les messages système d'un ou plusieurs hôtes à surveiller. Syslog-ng permet de travailler en réseau très simplement. Envoyer les messages à un hôte distant revient simplement à définir une nouvelle destination et à l'utiliser :

```
destination dudp_syslog_server {
    udp("192.168.0.42" port(514));
};
```

udp est la directive permettant d'envoyer les messages en UDP à l'hôte spécifié en paramètre. Il est également possible

d'utiliser un protocole avec connexion, auquel cas on utilisera la directive **tcp**. C'est à vous de choisir le protocole qui vous conviendra le mieux en fonction votre architecture réseau (présence d'un VPN ou non par exemple).

Côté hôte recevant l'information, il suffira de définir une nouvelle source ou de l'inclure dans celle déjà définie :

```
source s_webserver {
    udp(ip(192.168.0.42) port(514));
};
```

La directive **udp** est la même, mais les paramètres diffèrent. Par défaut, aucun n'est indispensable. Ici, on précise l'IP de l'interface à écouter et le port à utiliser, car la machine dispose de plusieurs interfaces.

Partant de là, deux solutions s'offrent à vous :

- Vous pouvez filtrer les messages et n'en envoyer qu'une partie sur le serveur Syslog-ng distant.
- Vous pouvez envoyer tous les messages du système et configurer le filtrage sur le serveur Syslog-ng distant.

L'élément à prendre en considération est, bien entendu, la consommation de la bande passante des serveurs. Faire transiter tous les messages d'une machine dédiée à une autre ne devrait pas être un problème. Idem pour des machines sur un LAN. Il en va tout autrement si vous renvoyez, par exemple, une copie des messages du serveur dédié vers une machine cliente connectée via ADSL à un VPN.

4

Enregistrer dans MySQL

L'archivage et le stockage des journaux système, même parfaitement triés et organisés, posent un problème de gestion à la fois d'espace disque (d'où l'intérêt de la rotation des journaux), mais aussi de stockage de l'information. Les fichiers journaux sont de simples fichiers textes et l'on a beau maîtriser **grep**, **awk** et **sed** comme un *dino*, la souplesse obtenue n'est pas celle d'une base de données dans laquelle on peut piocher à souhait.

L'utilisation de MySQL conjointement avec Syslog-ng est d'une simplicité déconcertante. Nous avons vu précédemment qu'il était possible de créer une destination pouvant être un tube, un fichier ou un hôte distant. Afin d'enregistrer les messages dans une base MySQL, nous allons utiliser comme destination un programme :

```
destination d_mysql {
  program("mysql -uroot -pmon_passe_mysql syslog"
  template("INSERT INTO logs
  (host, facility, priority, level,
  tag, date, time, program, msg)
  VALUES
  ('$HOST', '$FACILITY', '$PRIORITY',
  '$LEVEL', '$TAG', '$YEAR-$MONTH-$DAY',
  '$HOURL:$MIN:$SEC', '$PROGRAM', '$MSG');\n")
  template-escape(yes)
  );
};
```

Nous créons la destination **d_mysql** utilisant la directive **program** avec, en paramètre, la ligne de commande complète. Le programme lancé (**mysql**) recevra sur son entrée standard (STDIN) le message qui aura été dirigé vers la destination **d_mysql**.

Nous complétons ensuite les arguments de la directive **program** avec le **template** qui formatera les messages envoyés sur le **STDIN** du programme. On voit ici clairement comment la requête d'insertion SQL est composée. Notez que **template** n'est pas limité à l'utilisation de **program**, mais peut également être ajouté aux directives **pipe** et **file**. La requête SQL est complétée par des valeurs directement issues des variables utilisées par Syslog-ng : **\$HOST**, **\$FACILITY**, **\$PRIORITY**, etc. La liste complète des variables utilisables est donnée dans la documentation officielle de Syslog-ng (nous les utilisons presque toutes ici).

Nous ajoutons également le paramètre **template-escape(yes)** qui permet d'activer l'échappement des caractères simples et doubles *quotes* pour les variables dans la sortie. Ceci est particulièrement intéressant pour les requêtes SQL et permet d'éviter d'éventuelles injections SQL de la part d'un

attaquant. Le principe est le même que pour le développement d'applications PHP.

Notre destination est maintenant prête, mais il nous faut à présent créer la base et la table MySQL :

```
CREATE DATABASE syslog;
USE syslog;
CREATE TABLE logs (
  host varchar(32) default NULL,
  facility varchar(10) default NULL,
  priority varchar(10) default NULL,
  level varchar(10) default NULL,
  tag varchar(10) default NULL,
  date date default NULL,
  time time default NULL,
  program varchar(15) default NULL,
  msg text,
  seq int(10) unsigned NOT NULL auto_increment,
  PRIMARY KEY (seq)
) TYPE=MyISAM;
```

La création de notre destination est terminée. Il ne nous reste plus qu'à l'utiliser dans un **log** :

```
log {
  source(s_all);
  filter(f_ssh_login_attempt);
  destination(d_mysql);
};
```

En cas de problème, jetez un coup d'œil aux journaux de Syslog-ng. Si vous y voyez un message de type :

```
syslog-ng[8560]: I/O error occurred while writing;
fd='6',
error='Broken pipe (32)'
```

Il s'agira sans doute d'un problème de connexion à la base de données. Vérifiez login, mot de passe et permissions sur la base désignée.

Il ne s'agit là, bien sûr, que d'une base de travail. L'utilisation des variables dans la définition de la destination couplée aux capacités de filtrage vous permettent de stocker la totalité des informations utiles dans des tables MySQL (**template("INSERT INTO \$FACILITY (host...)**), voire différentes bases MySQL.

Une fois toutes les informations stockées ainsi dans la base, les requêtes d'interrogation ou encore l'écriture de scripts PHP analysant et graphant les informations est un jeu d'enfant. Dans le même ordre d'idée et toujours en relation avec la notion de « base de données », vous pouvez imaginer interfacer de la même manière Syslog-ng et RRDtools.

5

Conclusion

Nous venons de voir que le système de gestion des journaux a grandement évolué avec l'apparition de Syslog-ng. Cet article ne présente cependant qu'une partie de ce qu'il est possible de faire. La syntaxe du fichier de configuration et les fonctionnalités de Syslog-ng vous permettent de faire tout ce qu'il vous plaît. Je le dis et le répète, les journaux sont un élément important de la sécurité d'un serveur. Utiliser et correctement configurer son système Syslog ou Syslog-ng ne vous assurera pas contre une attaque, mais représente un élément parfois déterminant dans l'analyse d'incidents.



Denis Bodor

Rédacteur en chef de GLMF.
Utilisateur GNU/Linux depuis 1994.
Randonneur du jardin magique.

Boostez votre collection!

LES 4 FAÇONS DE COMMANDER !

Au **03 88 58 02 09** C.B. et/ou bon de commande administratif.

BON DE COMMANDE POWER PACKS À REMPLIR ET À RETOURNER À (OU PHOTOCOPIER)
Diamond éditions - GNU/Linux Magazine - BP 20142 - 67603 SÉLESTAT Cedex

1 Voici mes coordonnées postales :																																																	
Nom :																																																	
Prénom :																																																	
Adresse :																																																	
Code Postal :																																																	
Ville :																																																	
2 Je joins mon règlement :																																																	
Je règle par chèque bancaire ou postal à l'ordre de Diamond Editions																																																	
Paiement par carte bancaire : <table border="1"> <tr> <td>N° Carte :</td> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> </tr> <tr> <td>Expire le :</td> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td>Cryptogramme Visuel :</td> <td></td><td></td><td></td> <td>Voir image ci-dessous</td> </tr> </table>															N° Carte :																		Expire le :												Cryptogramme Visuel :				Voir image ci-dessous
N° Carte :																																																	
Expire le :												Cryptogramme Visuel :				Voir image ci-dessous																																	
Date et signature obligatoire :												200																																					

Abonnez-vous ! Économisez

*

Plus de

20%

* Sur le prix de vente unitaire France Métropolitaine



11 Numéros de
GNU/Linux Magazine

11

Numéros de
GNU/Linux Magazine
pour
le prix de 9*

* Gain pour un abonnement France Métropolitaine, par rapport au prix unitaire France Métropolitaine

à
55€
(Offre France Métro)

Soit

Votre GNU/Linux Magazine

à
5,00€
(Tarif au numéro dans le cadre
d'un abonnement France Métro)

Vous lisez d'autres magazines des Éditions Diamond ?
Des offres de couplage sont disponibles dans ce magazine.

Les 3 bonnes raisons de vous abonner !

- ▶ Ne manquez plus aucun numéro.
- ▶ Recevez GNU/Linux Magazine chaque mois chez vous ou dans votre entreprise.
- ▶ Économisez 16,50 €/an ! (soit plus de 2 magazines offerts !)

Pour les tarifs hors France Métropolitaine, consultez notre site : www.ed-diamond.com.

Offres d'abonnement

(Nos tarifs s'entendent TTC et en euros)

Offres d'abonnement		F	D	T	E1	E2	EUC	A	RM
(Nos tarifs s'entendent TTC et en euros)		France Métro	DOM	TOM	Europe 1	Europe 2	Etats-unis Canada	Afrique	Reste du Monde
1	Abonnement Linux Magazine	55 €	59 €	67 €	69 €	66 €	70 €	68 €	77 €
2	Linux Magazine + Hors-série	83 €	89 €	101 €	104 €	100 €	105 €	103 €	116 €
3	Linux Magazine + MISC	84 €	90 €	102 €	105 €	101 €	107 €	104 €	117 €
4	Linux Magazine + Linux Pratique	78 €	85€	96 €	99 €	95 €	101 €	98 €	111 €
5	Linux Magazine + Hors-série + Linux Pratique	110 €	119 €	134 €	138€	133 €	140 €	137 €	154 €
6	Linux Magazine + Hors-série + MISC	116 €	124 €	140 €	144 €	139 €	146 €	143 €	160 €
7	Linux Magazine + Hors-série + MISC + Linux Pratique	143 €	154 €	173 €	178 €	172 €	181 €	177 €	198 €
8	Linux Pratique Essentiel + Linux Pratique	57 €	62 €	69 €	71 €	69 €	73 €	71 €	79 €

• Europe 1 : Allemagne, Belgique, Danemark, Italie, Luxembourg, Norvège, Pays-Bas, Portugal, Suède
• Europe 2 : Autriche, Espagne, Finlande, Grande Bretagne, Grèce, Islande, Suisse, Irlande

• Zone Reste du Monde : Autre Amérique, Asie, Océanie
• Zone Afrique : Europe de l'Est, Proche et Moyen-Orient

Toutes les offres d'abonnement : en exemple les tarifs ci-dessous correspondant à la zone France Métro (F)
(Vous pouvez également vous abonner sur : www.ed-diamond.com)

Linux Magazine
(11 n°)

offre 1

par ABO : **55€**

Economie : 16,50 €

en kiosque : **71,50€**

Linux Magazine
(11 n°)

en kiosque : **110,50€**

offre 2

par ABO : **83€**

Economie : 27,50 €

Linux Magazine
(11 n°)

en kiosque : **119,50€**

offre 3

par ABO : **84€**

Economie : 35,50 €

Linux Magazine
(11 n°)

en kiosque : **107,20€**

offre 4

par ABO : **78€**

Economie : 29,20 €

Linux Magazine
(11 n°)

en kiosque : **146,20€**

offre 5

par ABO : **110€**

Economie : 35,20 €

Linux Magazine
(11 n°)

en kiosque : **158,50€**

offre 6

par ABO : **116€**

Economie : 42,50 €

Linux Magazine
(11 n°)

en kiosque : **194,20€**

offre 7

par ABO : **143€**

Economie : 51,20 €

Linux Pratique
(6 n°)

en kiosque : **74,70€**

offre 8

par ABO : **57€**

Economie : 17,70 €

Bon d'abonnement à découper et à renvoyer à l'adresse ci-dessous :

Je fais mon choix de la 1ère offre :

Je sélectionne le N° (1 à 8) de l'offre choisie :	
Je sélectionne ma zone géographique (F à RM) :	
J'indique la somme due :	€

Exemple : je souhaite m'abonner à l'offre Linux Magazine + Hors-série + MISC (offre 6) et je vis en Belgique (E1), ma référence est donc 6E1 et le montant de l'abonnement est de 144 euros.

Je choisis de régler par :

- ☐ Chèque bancaire ou postal à l'ordre de Diamond Editions
- ☐ Carte bancaire n°
- Expire le :
- Cryptogramme visuel :

Date et signature obligatoires



Je fais mon choix de la 2ème offre :

Je sélectionne le N° (1 à 8) de l'offre choisie :	
Je sélectionne ma zone géographique (F à RM) :	
J'indique la somme due :	€
Total	€

Diamond Editions
Service des Abonnements
B.P. 20142 - 67603 Sélestat Cedex

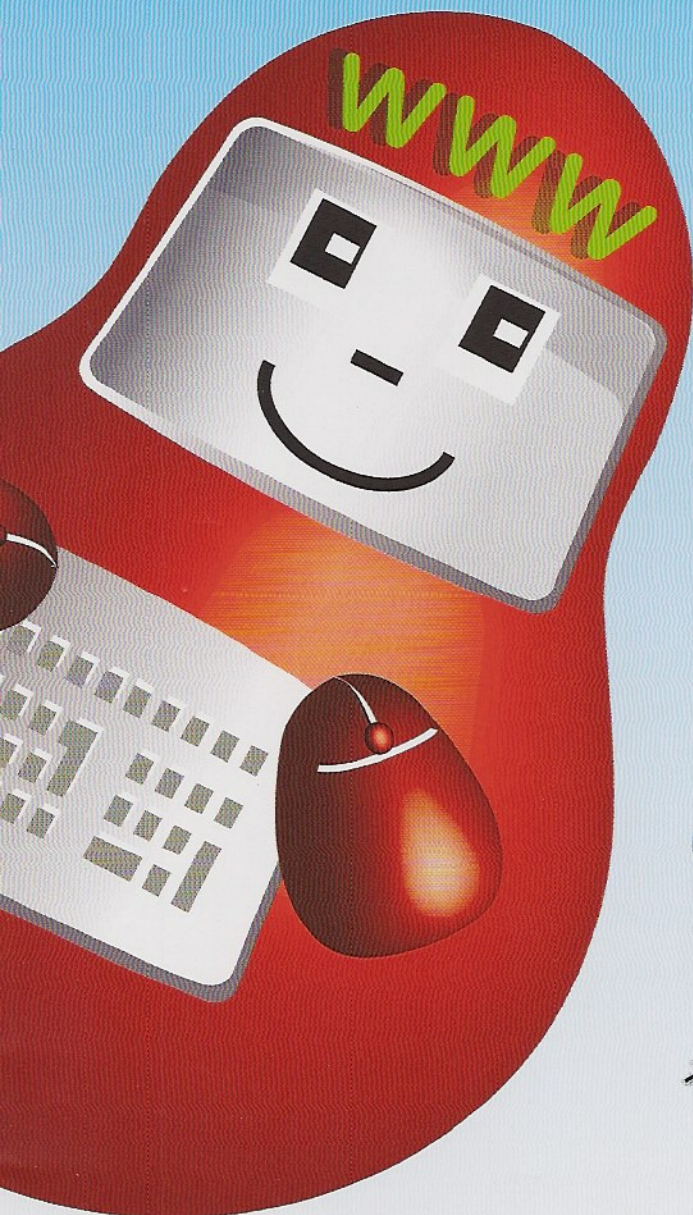
GNU/LINUX MAGAZINE HORS-SÉRIE



du... au

Votre cryptogramme visuel

Dans la famille NFrance, je demande le serveur dédié !



L'offre «**serveur dédié 100 % infogéré**» est le cœur de compétence de NFrance Conseil.

Des solutions d'hébergement intégralement infogérées pour votre activité en ligne.

Le suivi et l'accompagnement personnalisé pour une qualité de service optimale.

La garantie de fonctionnement par contrat.

➔ Ce sont les engagements et le savoir-faire de NFrance.

NFrance c'est aussi :



➔ **Domaine**
achat, transfert,
gestion de vos
noms de domaine



➔ **Hébergement**
professionnel
ou de loisir, mutualisé,
dédié virtuel ou dédié

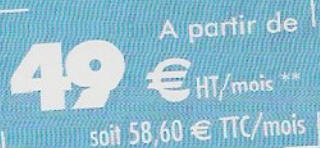


➔ **Architectures**
complexes,
redondantes,
haute charge

Intégrateur ou entreprise à la recherche de sécurité pour vos applications ASP, webservices, sauvegardes, VPN sécurisé, n'hésitez pas à nous contacter au **05 34 455 500**.

Il y a forcément une solution d'hébergement spécialement étudiée pour vous...

D'ÉMISSION
-55%
DE CO₂*



<http://eco.amen.fr>

- AMD Athlon X2 3400+ ou BE-2350
- De 2x1,8GHz à 2x2,1GHz
- Interface Plesk 8.2 – 10 domaines
- OS : Fedora Core 7, Ubuntu 6, Debian 4, Windows Server 2003
- De 1 à 3 Go de RAM
- Disque dur de 80 Go à 160 Go
- Trafic mensuel : 1 To
- 1 à 2 adresses IP
- Inclus : Reboot, Restore et Recovery
- En option : Amen DataBackup 10 Go (Inclus sur ECO 3000)



 entrepreneurs,
faites le choix de
l'économie numérique
www.econumerique.pmi.gouv.fr