Groupe de travail Réseau
Request for Comments : 1794
Catégorie : Information

T. Brisco, Rutgers University
avril 1995
Traduction Claude Brière de L'Isle

Prise en charge de l'équilibrage de charge par le DNS

Statut de ce mémoire
Le présent mémoire apporte des informations pour la communauté de l'Internet. Il ne spécifie aucune sorte de norme de l'Internet. La distribution du présent mémoire n'est soumise à aucune restriction.

1. Introduction

La présente RFC est destinée à rendre compte d'un débat au sein du groupe de travail DNS de l'IETF, sur les possibles solutions de remplacement pour fournir/simuler la prise en charge de l'équilibrage de charge dans le DNS, et pour fournir une solution finale souple pour que le DNS prenne en charge de nombreux types d'équilibrage de charge.

2. Historique

L'histoire de ce débat remonte probablement bien avant mon époque – aussi y aura t-il indubitablement quelques lacunes. On peut espérer qu'elles seront comblées par d'autres auteurs.

Initialement; "l'équilibrage de charge" était destiné à permettre aux agents du système des noms de domaines (DNS, *Domain Name System*) [1] de prendre en charge le concept de "grappes" (dérivé de l'utilisation du VMS) de machines – où toutes les machines étaient fonctionnellement similaires ou les mêmes, et il n'importait pas particulièrement de savoir quelle machine était sollicitée – tant que la charge du traitement était raisonnablement bien répartie à travers une série d'hôtes réels différents. Vers 1986 un certain nombre de schémas différents ont commencé à faire surface comme des brèches dans la distribution du serveur des noms de domaines de l'Internet de Berkeley (BIND, *Berkeley Internet Name Domain*). Probablement la plus largement distribuée d'entre elles était la modification "Shuffle Address" (SA) par Bryan Beecher, ou éventuellement le code "Round Robin" de Marshall Rose.

Les enregistrement SA ont cependant fait un classement round-robin des enregistrements de ressource d'adresse, et n'ont pas fait grand chose en ce qui concerne les charges particulières des machines cibles. Matt Madison (de TGV) a mis en œuvre quelques changements qui utilisaient les facilités de VMS pour revoir les charges du système, et retournaient les enregistrements de ressource (RR) A dans l'ordre du moins chargé au plus chargé.

Le problème avec les SA était que la charge n'était pas réellement un facteur, et les TGV s'appuyaient sur des facilités spécifiques de VMS pour ordonner les enregistrements. Les RR SA exigeaient des changements à la spécification du DNS (à la syntaxe des fichiers et au traitement des enregistrements). Cela était vu à la fois comme un inconvénient et pas comme une solution générale.

La plupart des acteurs de l'Internet attendaient une méthode approuvée par l'IETF pour simuler des "grappes".

Suite à quelques sessions du groupe de travail DNS de l'IETF (présidées par Rob Austein de Epilogue), il y a eu un accord général sur le fait qu'un certain nombre de critères devaient être satisfaits :

A) Rétro compatibilité avec la RFC existante sur le DNS.
B) Les informations changent fréquemment.
C) Plusieurs adresses devraient être envoyées.
D) Doit interagir correctement avec les autres RR.
E) Doit être capable de représenter de nombreux types de "charges".
F) Doit être rapide.

(A) assurerait que la base installée de BIND et des autres mises en œuvre du DNS continueraient de fonctionner et d'interopérer correctement.

(B) permettrait des temps de mise à jour très rapides – pour permettre de modéliser des données en temps réel. Cinq minutes étaient estimées un intervalle normal, bien que des changements aussi rapides que toutes les soixante secondes puissent être imaginés.

(C) couvrirait la possibilité que l'adresse d'un hôte soit annoncée comme optimale, bien que la machine ait une défaillance durant la période du TTL du RR. La seconde adresse dans l'ordre de préférence serait annoncée en second, la troisième ensuite, et ainsi de suite. Cela permettrait donnerait une perspective raisonnable de récupération durant les défaillances de machines.

(D) assurerait un traitement correct de toutes les informations auxiliaires – telles que les informations de MX, RP et TXT, ainsi que les informations de recherche inverse. Il était nécessaire de s'assurer que des processus tels que le traitement de la messagerie continueraient de fonctionner d'une façon sans surprise et prévisible.

(E) assurerait la souplesse souhaitée par tous. Divers membres du groupe de travail DNS souhaitaient qu'un large éventail de "charges" soit représenté. Certaines "charges" étaient très éclectiques – comme l'ordre des adresses par le RTT chez l'hôte, d'autres étaient pragmatiques – comme l'équilibrage de la charge de CPU à travers une série d'hôtes. Toutes représentaient des préoccupations valides dans leur propre contexte, et l'idée d'avoir des types de RR distincts pour chacun était impensable (principalement, cela aurait violé l'objectif A).

(F) nécessitait de s'assurer de quelques petites choses. Principalement que le temps pour calculer les informations pour ordonner les information d'adressage n'excédait pas le TTL des informations distribuées – c'est-à-dire, que les éléments avec un TTL de cinq minutes ne prenaient pas six minutes à calculer. De même, cela semble un objectif parfaitement clair dans la RFC du DNS que les clients ne devraient pas rester à attendre – c'est à dire que le traitement des demandes devrait se poursuivre sans considération de l'état de tout autre traitement à effectuer.

3. Solutions de remplacement possibles

Durant les diverses discussions au sein du groupe de travail DNS et avec le comité d'équilibrage de charge, il a été constaté qu'il n'existait aucune solution qui satisfasse correctement tous les souhaits. Un des succès majeurs du DNS est sa souplesse – et il a été estimé que cela devait être conservé dans tous les aspects. Il a été estimé que peut être non seulement les informations d'adresse auraient besoin d'être changées rapidement, mais que d'autre enregistrement pourraient aussi devoir changer rapidement (au moins cela ne pouvait pas être exclu – qui sait ce que la technologie nous cache pour l'avenir).

Le principal souci de beaucoup était la capacité d'interagir avec les anciennes mises en œuvre du DNS. Le DNS est maintenant largement mis en œuvre, et des changements à des portions critiques du protocole pourraient causer des désordres pour des années. Il est devenu rapidement évident à travers des conversations avec Jon Postel et Dave Crocker (Directeur de zone) que les modifications au protocole seraient vues d'un œil très critique.

4. Un modèle flexible

Après de nombreuses heures de discussions est apparue la suggestion de Rob Austein que les changements pourraient être mis en œuvre sans modification du protocole ; si le comportement de transfert de zone pouvait être changé de façon subtile, le processus de transfert de zone pourrait s'accommoder du changement des diverses informations de RR. Ce qui était nécessaire était un programme plus intelligent pour les transferts de zone. Conformément à cette idée, des changements ont été apportés à BIND qui permettait la spécification du programme des transferts de zone pour des zones particulières.

Il n'est spécifié nulle part qu'un serveur secondaire doive recevoir les mises à jour de son serveur primaire d'une manière spécifique – il doit seulement vérifier périodiquement, et obtenir de nouvelles copies de zone lorsque les changements ont été faits. On peut concevoir que l'agent de transfert de zone puisse obtenir les informations de la part de nombreuses sources (par exemple, un démon d'équilibrage de charge, un trieur round-robin) et qu'il présente les informations en retour au serveur de noms pour leur distribution.

Un certain nombre de questions se posent au sujet de ce concept, et toutes semblent avoir été réglées.

Premièrement, le protocole du DNS ne garantit pas d'ordre. Bien que le protocole DNS ne garantisse pas d'ordre, il est clair que l'ordre est prévisible – que les informations lues deux fois dans le même ordre seront présentées deux fois dans le même ordre aux clients. Les clients, bien sûr, peuvent réordonner ces informations, mais c'est réputé une "affaire locale"

car c'est configurable par les administrateurs des systèmes distants (par exemple, des listes de tri, etc.). L'agent de transfert de zone devrait avoir à tenir compte de tout "désordre" qui pourrait survenir localement, mais la réorganisation à distance (par exemple, des listes de tri côté client) des RR est impossible à prédire. Comme le non ordre local est cohérent, les agents de transfert de zone peuvent facilement en tenir compte.

Deuxièmement mais peut-être plus subtil, se pose le problème des transferts de zone qui ne sont pas utilisés par les serveurs de noms primaires, et seulement par les serveurs de noms secondaires. Pour préciser cela, l'idée de sous zones "rapides" ou "volatiles" doit être approfondie. Dans un environnement volatile (où l'ordre des adresses ou d'autres RR change rapidement), le taux de rafraîchissement d'une zone doit être réglé très bas, et le TTL des RR traités doit de même être très court. Il ne sert à rien de traiter une information avec un TTL d'une heure, lorsque les conditions pour ranger les RR changent toutes les minutes. Il doit y avoir une relation relativement étroite entre les taux de rafraîchissement et les TTL des informations. Bien sûr, avec des taux de rafraîchissement très bas, les transferts de zone entre les serveurs primaires et secondaires devraient avoir lieu fréquemment. Étant donné que les serveurs de noms primaires et secondaires devraient être topologiquement et géographiquement bien séparés, déplacer ces quantités de données fréquemment est vu comme prohibitif. Aussi, plus le temps de propagation entre serveurs primaire et secondaire est long, plus est large la fenêtre dans laquelle les circonstances peuvent changer – et donc invalider les informations du serveur secondaire. On pense généralement que passer des informations volatiles à un serveur secondaire est parfaitement inutile – si les serveurs secondaires veulent des informations exactes, il devraient alors les calculer eux-mêmes et non les obtenir via les transferts de zone. Cela évite le problème des serveurs secondaires qui perdent le contact avec les primaires (mais l'accès aux cibles du domaine volatile est toujours disponible), mais les informations du serveur secondaire se périment.

Ce qui est par essence nécessaire est un serveur secondaire (sans primaire) qui puisse calculer l'ordre nécessaire des données de RR pour lui-même (ce qui évite aussi le problème des versions différentes de serveurs de domaines qui ordonnent de façon prévisionnelle les informations de RR selon des prévisions différentes). Pour une zone volatile, il n'y a pas d'agent DNS primaire, mais plutôt une série d'agents secondaires autonomes. Chaque agent secondaire autonome est, bien sûr, capable de calculer lui-même l'ordre ou le contenu des RR volatiles.

5. Mise en œuvre

Avec l'aide de Masataka Ohta (Institut de technologie de Tokyo), j'ai mis en œuvre les modifications de BIND pour permettre la spécification du programme de transfert de zone (d'agent de transfert de zone) pour des domaines particuliers :

transfert <nom-de-domaine> <nom-de-programme>

Actuellement je définis un sous domaine séparé qui a quelques hôtes rattachés toutes ces informations sont volatiles. La zone a un taux de rafraîchissement de 300, et un TTL minimum indiqué de 300. Le fichier de configuration est indiqué comme "volatile.hosts". Toutes les 300 secondes, un programme "doAxfer" effectue le transfert de zone. Le programme "doAxfer" lit le fichier "volatile.hosts.template" et le fichier "volatile.hosts.list". Les adresses spécifiées dans volatile.hosts.list subissent une rotation d'un nombre de fois aléatoire, puis sont substituées (dans l'ordre) dans volatile.hosts.template pour générer le fichier volatile.hosts. Le programme "doAxfer" sort alors avec une valeur de 1 - pour indiquer au serveur de noms que le transfert de zone a réussi, que le fichier devrait être mémorisé et les informations distribuées. Il en résulte qu'un hôte a plusieurs adresses, et les adresses sont rendues aléatoires toutes les cinq minutes (300 secondes).

Deux bogues continuent de nous gêner dans cette entreprise. BIND considère actuellement tout TTL en dessous de 300 secondes comme "irrationnel", et lui substitue la valeur de 300. Cela dégrade considérablement les fonctionnalités des zones volatiles. Dans le plus rapide de tous les cas - un TTL de 0 – l'information sera utilisée une seule fois, puis éliminée. On peut présumer que la nouvelle information de RR pourrait être calculée toutes les 5 secondes, et les RR traités avec un TTL de 0. Il faut considérer que cette limitation de la vitesse d'une zone va être la capacité d'une machine à calculer assez vite les nouvelles informations.

L'autre bogue qui affecte aussi le programme est que, comme avec les TTL, BIND considère tout taux de rafraîchissement de zone en dessous de 15 minutes comme également irrationnel. Visiblement, des taux de rafraîchissement de zone de 15 minutes sont inacceptables pour ce type d'applications.

Comme solution de rechange, le code actuel règle ces mêmes valeurs incorporées à 60 secondes. Soixante secondes est encore assez long pour éviter toutes les bogues résiduelles associées aux petites valeurs de temporisation, mais aussi assez court pour permettre d'utiliser des sous zones éphémères.

Cette version de BIND est actuellement livrée à l'Université Rutgers, et fonctionne à la fois en zones "éphémères" et normales.

6. Performances

Bien que les performances des zones éphémères ne soient pas exactement stellaires, elles ne sont pas beaucoup plus que les charges normales de CPU induites par BIND. Les essais ont été effectués sur un Sun Sparc-2 utilisé comme une station de travail normale, mais aucun résolveur n'utilisait le serveur de noms – le serveur de noms était essentiellement inoccupé. Pour une configuration sans sous zones éphémères, BIND a accumulé 11 secondes de CPU en 24 heures. Pour une configuration avec une zone éphémère, six enregistrements d'adresse, rafraîchis toutes les 300 secondes (5 minutes), BIND a accumulé 1 minute 4 secondes de temps de CPU. Pour la même configuration précédente, mais rafraîchie toutes les soixante secondes, BIND avait accumulé 5 minutes et 38 secondes de temps de CPU.

Comme ce n'est pas très surprenant, la charge de CPU sur la machine serveuse était linéaire à la fréquence du temps de rafraîchissement. La configuration de rafraîchissement à soixante secondes utilisait approximativement cinq fois autant de temps de CPU que le fait la configuration de rafraîchissement à 300 secondes. On peut facilement extrapoler que l'utilisation globale de CPU serait strictement proportionnelle au nombre de zones et à la fréquence de la période de rafraîchissement. Tout ceci se fonde sur un dogme gravé dans le marbre qui a toujours dit qu'une mise à jour de zone était nécessaire, un programme plus intelligent devrait réaliser quant la réorganisation des RR n'est pas nécessaire et éviter de tels rechargements périodiques des zones.

7. Remerciements

La plupart des idées de ce document sont le résultat de conversations avec de très nombreuses personnes et des propositions provenant – entre autres, de Robert Austein, Stuart Vance, Masataka Ohta, Marshall Rose, et des membres du groupe de travail DNS de l'IETF.

8. Référence

[1] P. Mockapetris, USC/Information Sciences Institute, "Noms de domaines – Mise en œuvre et spécification", STD 13, RFC 1035, novembre 1987.

9. Considérations pour la sécurité

Les questions de sécurité ne sont pas abordées dans le présent mémoire.

10. Adresse de l'auteur

Thomas P. Brisco
Associate Director for Network Operations
Rutgers University
Computing Services, Telecommunications Division
Hill Center for the Mathematical Sciences
Busch Campus
Piscataway, New Jersey 08855-0879
USA

téléphone : +1-908-445-2351
mél : brisco@rutgers.edu