Equilibrage de charge multi-critère pour les serveurs DNS(SEC)

Stanislas Francfort, Stéphane Sénécal, Qinghui Xu, Daniel Migault

6 décembre 2010

Résumé

Cet article présente le problème de répartition de charge pour des requêtes DNSSEC. Ce problème a deux critères : équilibrer le nombre de requêtes tout en équilibrant le nombre de calcul de signatures cryptographiques. En tirant parti de la distribution statistique très particulière des requêtes DNS, ainsi que de la clusterisation étudiée dans [15], nous définissons une méthode efficace pour résoudre ce problème. Cette optimisation permet une économie d'échelle sur le nombre de serveurs à déployer en production ainsi qu'une utilisation énergétique largement moindre que celle utilisée actuellement.

Table des matières

1 Introduction

Les serveurs DNS reçoivent une énorme quantité de requêtes, et ce nombre de requêtes ne cesse de croître. Afin de répartir la charge entre différents serveurs, l'architecture est celle-ci : un serveur de répartition de charge reçoit toutes les requêtes, et les répartit selon une distribution uniforme entre d'autres serveurs qui traitent eux-mêmes les requêtes. Dans ce cas de figure, pour éviter qu'un serveur arrive en limite de capacité de charge alors que les autres ont encore la capacité à traiter des requêtes, il convient de répartir équitablement (cad. selon une répartition uniforme) les requêtes entre les différents serveurs.

Or, avec l'arrivée prochaine du DNSSEC, une autre problématique se pose. En effet, chaque requête est adressée avec un FQDN. Or pour chaque FQDN une requête de calcul de signature cryptographique aura lieu. Ce calcul est gourmand en ressource CPU et il convient d'en minimiser le nombre. L'administration des serveurs DNSSEC prévoit de mettre en cache le résultat des calcul de signatures cryptographiques pendant un certain temps (appelé TTL - Time To Leave). Ainsi, si deux requêtes DNSSEC ayant le même FQDN sont adressés à deux serveurs différents, il sera effectué un calcul cryptographique inutile alors que la deuxième requête aurait pu être envoyée au même serveur et un calcul aurait pu être évité (sous réserve que la deuxième requête arrive dans le temps imparti par le TTL).

Nous avons alors la possibilité de répartir uniformément les FQDN entre les serveurs de telle sorte que chaque FQDN soit attribué à un unique serveur. Ainsi, chauque serveur calculera autant de signature crytpographique que les autres. Mais dans ce cas, il n'est pas assuré que chaque serveur traitera autant de requêtes. La situation est même pire, car la répartition du nombre de requêtes par FQDN est telle qu'en procédant de la sorte on crée un grand déséquilibre entre le serveur qui reçoit le plus de requêtes et celui qui reçoit le moins de requêtes.

Nous nous intéresserons donc dans cet article à la répartition équitable aussi bien en terme de nombre de requêtes qu'en terme de nombre de signatures cryptographiques. Dans un premier temps, dans la section 2 nous allons présenter les données DNS, puis dans la section 3 nous allons étudier la distribution statistique particulière. En section 4 nous allons tirer partie de cette distribution pour modéliser le problème et le résoudre. Nous allons également discuter l'efficacité de ces nouvelles méthodes. La section 5 concluera cet article.

Les calculs statistiques ont été effectués avec le logiciel R interfacé avec MySQL, le solveur MIP utilisé est GLPK v4.38, le langage PERL v5 a été largement mis à contribution pour gérer les données. Tous les calculs ont été effectués sur un PC équipé d'un processeur AMD 64 bits à 3.1Ghz quadri-coeur avec 4Go de RAM. L'OS est un Linux 2.6.35.

2 Présentations des données

Afin de garantir que les données utilisées dans cette étude soient réalistes, nous avons travaillés sont des captures de requêtes DNS issues de serveur de production DNS de Orange datant de 2009. Le fichier principal étudié est une capture brute de 5 minutes de trafic, soit 800Mo de données.
Dans la suite de l'article, nous allons noter $i \in I$ les FQDN, et N_i le nombre de requêtes ayant pour objet le FQDN i.
Voici un exemple des 10 requêtes les plus demandées pendant ces 5 minutes de captures de flux réseau :

nom	nbr_occur
www.facebook.com	271586
wpad	256144
1.0.0.127.dnsbugtest.1.0.0.127.in-addr.arpa	206656
ad.fr.doubleclick.net	193632
www.google.com	187028
view.atdmt.com	131181
www.yahoo.com	130095
profile.ak.fbcdn.net	129737
www.google-analytics.com	124445
www.google.fr	116170

On peut déjà voir dans cet exemple que la différence entre le FQDN le plus demandé et le 10^{eme} FQDN, il y a un rapport du simple au double... La répartition est en effet très inégale ainsi, en 5 minutes, on a capturé 17299154 requêtes réparties en 1211880 FQDN ce qui nous donne une moyenne de 14.27 requêtes par FQDN.
Cependant, ne nous y trompons pas, cette moyenne n'a aucun sens. En effet, le minimum de cette répartition est 1 requête, mais la médiane est aussi égale à 1, comme la moyenne. Il y a en fait, 837154 FQDN demandés une unique fois et 374726 FQDN demandés strictement plus que une fois.
Contrairement à cela, le maximum vaut 275586 et est atteint pour un unique

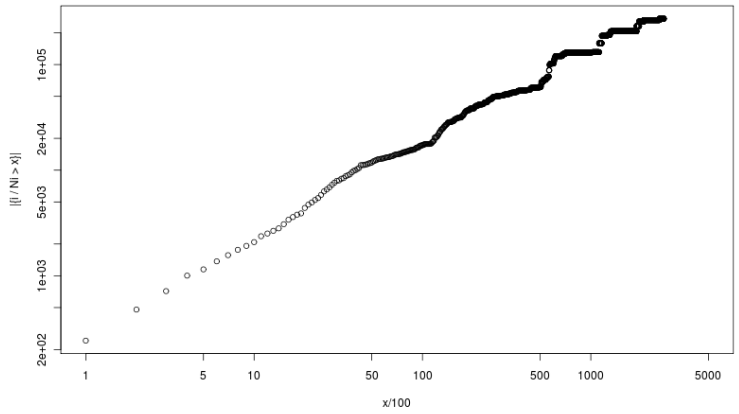

FIGURE 1 – Répartition linéaire des i tels que $N_i > x$

FQDN.

Nous allons étudier plus en détail cette répartition fortement inégale dans la section suivante puis en tirer parti afin de décomposer le problème.

3 Distribution statistique

La répartition du nombre de requêtes par FQDN est très inégalitaire. Il est très vraisemblable que cette répartition suive une loi de puissance. En effet, certains phénomènes où la notoriété engendre la notoriété conduise à l'apparition de lois de puissances. On peut voir dans [1] l'analyse de la distribution du nombre de requêtes par domaine sur les serveurs de AOL au cours d'une journée. L'analyse conduit à la conclusion que cette répartition se distribue effectivement selon une loi de puissance. Le cas de la popularité des FQDN dans les requêtes DNS est un cas similaire.
Cependant, nous n'avons pas besoin de prouver que nous avons affaire à une loi de puissance, seul le comportement global est important, et savoir que l'on se rapproche d'une loi de puissance, avec une grande inégalité entre les événements les plus et les moins fréquents, ainsi qu'une nombre écrasant d'événements de faible fréquence nous permettra de concevoir une méthode efficace pour résoudre notre problème.
La figure 1 représente le nombre de FQDN i tq $N_i > x$ en fonction de x. En d'aute terme, il s'agit de FQDN qui reçoivent un nombre de requêtes supérieur à un nombre donné.

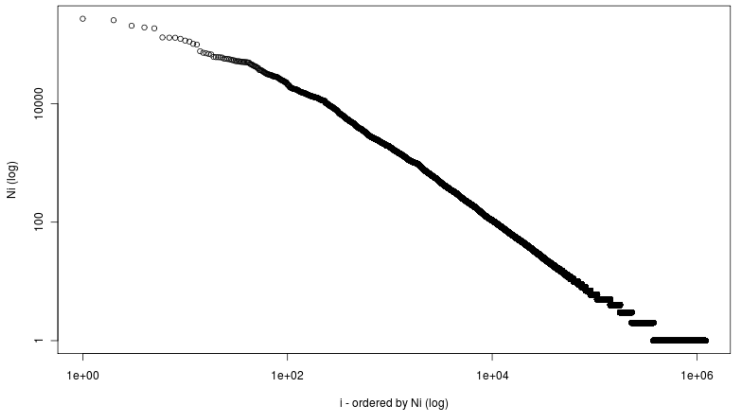

FIGURE 2 – Répartition des N_i (log-log)

La figure 2 représente la distribution des N_i ordonnés du plus fréquent au moins fréquent. Affiché dans un graphique log-log, cette distribution en linéaire dasn sa plus grande partie, ce qui est une caractérisituqe des distribution en loi de puissance.

Les quantiles de cette distribution sont également parlants pour indiquer la répartition très inégalitaire des N_i entre les différents FQDN i :

quantiles				
0%	25%	50%	75%	100%
1	1	1	2	271586

Un autre indicateur nous indiquant que nous sommes probablement en présence d'une loi de type loi de puissance est la valeur du coefficient de Gini. Cette indicateur est calculé de la manière suivante... Pour plus d'information, se reporter par exemple à [16] ou [8]. Ce coefficient correspond au rapport de proportion entre l'aire de la surface entre la courbe de répartition des N_i et la droite joignant le minimum au maximum. Plus le coefficient de Gini est élevé, plus la distribution est inégale. Le coefficient de Gini de la distribution des N_i, avec une valeur de $1-1.05*10^{-4}$ indique une répartition très fortement inégalitaire.

Nous sommes donc en présence d'une distribution largement non gaussienne. On peut en conclure plusieurs choses. Tout d'abord, qu'il faudra se méfier lors de l'utilisation des indicateurs statistiques tels que la moyenne ou l'écart-type (variance) calculés de manière empirique, qui peuvent perdre tout sens dans un tel contexte. De plus, il conviendra également d'utiliser

avec prudence les théorèmes tels que le Théorème de la Limite Centrale ou la Loi des Grands Nombres. Ceux-ci ne s'appliquent pas dans un contexte non gaussien sans un minimum de prudence.

Il convient cependant d'être prudent en notant que cette ressemblance graphique ne permet pas de conclure qu'on est effetivement en présence d'une loi de puissance. Comme expliqué dans [9] et [6] il est possible de confondre avec une loi exponentielle ou log-normale, et nous devrions faire une étude statistique plus rigoureuse pour pouvoir affirmer sans ambiguïté que nous sommes en présence d'une loi de puissance.

Les caractéristiques de cette distribution sont telles que si la répartition des FQDN est uniforme entre les serveurs, alors ils auront à gérer un nombre de requêtes qui pôurra être très inégale. Cette propriété est dûe au fait que les FQDN les plus demandés dominent, et poussent la répartition au déséquilibre. La simulation d'une telle répartition nous donne un déséquilibre moyen de 805329 requêtes, ce qui est énorme comparé au nombre de requêtes total à répartir.
Notons que les FQDN qui ne sont demandés qu'une unique fois peuvent être, eux, répartis uniformément. Ainsi, le déséquilibre créé sera seulement de 1009, ce qui est négligeable.
A mi-chemin entre ces deux solutions, nous pouvons choisir de ne répartir qu'une partie des FQDN (ceus ayant les N_i les plus faibles), l'autre sera répartie selon une autre procédure, plus performante (ceux ayant les plus grand N_i. Cette procédure sera décrite en détail dans les sections suivanteS.
Il s'agit d'un programme linéaire en nombres entiers (PLNE ou MIP).
La table 1 et la figure 3 regroupent les déséquilibre résiduels en fonction du nombre de FQDN qui seront répartis selon le MIP. Nous y voyons l'intérêt de gérer un maximum de FQDN par MIP et de ne répartir uniformément qu'un minimum de FQDN.

4 Optimisation bi-critère

4.1 Description du problème

Le problème qui se pose est donc le suivant : répartir aussi équitablement que possible les requêtes sur un groupe de serveurs. Cette répartition est effectuée par un serveur de répartition de charge (load-balancer).
Etant donné un seuil s fixé, nous allons décrire comment le problème peut se modéliser. Nous rappelons le but à atteindre : étant donné un certain nombre de FQDN i recevant chacun N_i requêtes, il s'agit de répartir la charge de façon à ce que celle-ci soit équilibrée. Nous allons donc tacher d'obtenir une répartition telle que chaque serveur reçoit approximativement autant de re-

Nombre de FQDN gérés par MIP	Déséquilibre $S_{max} - S_{min}$
1	805329
100	233180
200	141988
300	111583
400	88818
500	77275
700	63588
1000	50768
2000	27468
4000	15579
8000	10414
16000	6202
32000	3799
48583	2625
374726	1009

TABLE 1 – Déséquilibre en fonction du nombre de FQDN répartis uniformément.

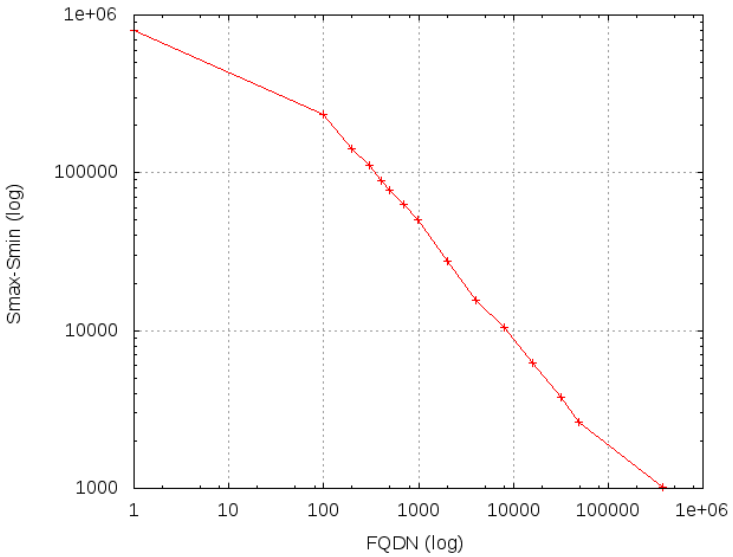

FIGURE 3 – Déséquilibre résiduel, courbe log-log

quêtes que les autres, et également que chaque serveur gère autant de FQDN que les autres, c'est à dire autant de signatures à vérifier.
Notons que ce problème est différent de celui du Bin-Packing et du Knapsack. Notre problème se réduirait à un problème de Bin-Packing si nous souhaitions une répartition ou la charge entre les serveurs soit strictement égale. Au contraire, nous cherchons un "bon" équilibre de charge entre les différents serveurs.Pour plus d'information sur le problème du Bin-Packing et sa résolution, voir par exemple [5].
Nous allons donc chercher à minimiser le déséquilibre, plutôt que d'introduire une contrainte imposant un déséquilibre nul.

Une donnée importante, et qui va diriger notre optimisation est la répartition très particulière du nombre de requête N_i par FQDN i (décrite en détail section 2).

4.2 Choix d'un seuil

Le choix d'un seuil s tel que les FQDN recevant plus que s requêtes seront repartis selon une optimisation, et les FQDN recevant moins que s requêtes seront répartis selon une distribution uniforme est cruciale. La table 1 et la figure 3 nous montre que plus s est petit plus la répartition sera équitable, mais plus le MIP sera difficile à résoudre. A contrario, plus s sera grand, plus le problème d'optimisation sera facile de résoudre , mais plus la répartition sera inéquitable entre les différents serveurs.

Dans la suite, nous allons étudier de quelle façon choisir un s satisfaisant et comparer les différents résultats. Ces résultats vont dépendrent des capacités de calcul à notre disposition.

Voici les différents choix possibles pour le seuil s :
- Fixer s à la médiane de la distribution : x tq $x > 1$. Ce choix nous conduit à optimiser un problème de taille 374726, ce qui risque d'être hors de portée de calcul.
- Plus raisonablement, fixer s à la moyenne de la distribution : x tq $x > 14,27$. Le problème d'optimisation à résoudre sera alors de taille 48583, ce qui est plus raisonable.
- Dans l'article [15] il est décrit une méthode de clustering dont le résultat met en exergue 4 clusters dont 3 d'entre eux comportent les FQDN les plus populiares. Plus précisément, sur 30 secondes de données DNS soit 167793 requêtes on obtient, avec l'algorithme "adaptative k-means" (resp. "k-means") à un clustering de 7, 57, 20 et 167.709 requêtes. Soit 84 éléments dans les 3 clusters les plus demandés ; (resp. 19, 95, 29 et 167.650, soit 143 éléments dans les 3 clusters les plus

demandés). Dans ce cas, on peut évaluer s à 200.

Nous allons par la suite envisager ces différentes valeurs de s et discuter leurs avantages et inconvénients.

4.3 Modélisations

4 programmes linéaires en nombres entiers (PLNE ou MIP) appelés F1 à F4 ont été définis. Ces 4 modèles sont similaires, en ce sens que leur relaxation continue admet la même solution optimale. cependant, comme souvent lorsqu'il s'agit de MIP, il sera possible de mettre en évidence des performances très différentes, certains modèles étant beaucoup plus efficaces que d'autres. Ces performances seront comparées dans la section suivante.
Soit les variables et les contraintes :

$$I \quad \text{l'ensemble des FQDN} \tag{1}$$

$$J \quad \text{l'ensemble des serveurs} \tag{2}$$

$$x_{i,j} = \begin{cases} 1 \text{ si le FQDN } i \text{ est affecté au serveur } j \\ 0 \text{ sinon} \end{cases} \tag{3}$$

$$k \in \mathbb{R} \quad \text{un paramètre de pénalisation} \tag{4}$$

$$N_{i \in I} \quad \text{le nombre de requêtes pour le FQDN } i \tag{5}$$

$$S_{j \in J} = \sum_{i \in I} N_i * x_{i,j} \quad \text{le nombre de requêtes que gère le serveur } J \tag{6}$$

$$T_{j \in J} = \sum_{i \in I} x_{i,j} \quad \text{le nombre de FQDN que gère le serveur } J \tag{7}$$

Au jeu de variables et de contraintes 1 à 7 communes à toutes les modélisations, nous allons ajouter un jeu de contraintes spécifiques à chaque modélisation.

4.3.1 Modèle F1

$$\sum_{j \in J} x_{i,j} \geq 1 \qquad \forall i \in I \tag{8}$$

$$u_j \geq 0 \quad \forall j \in J, \text{ variables d'écart} \tag{9}$$

$$l_j \geq 0 \qquad \forall j \in J \tag{10}$$

$$S_j - u_j + l_j = \frac{1}{\|I\|} \sum_{i \in I} (N_i) \qquad \forall j \in J \tag{11}$$

$$\text{Fonction objectif : minimiser } \sum_{j \in J} u_j + l_j \tag{12}$$

4.3.2 Modèle F2

$$\sum_{j \in J} x_{i,j} \geq 1 \qquad \forall i \in I \tag{13}$$

$$S_{j1} + k * T_2 j \leq M \quad \forall j1, j2 \in J, \text{ variable d'écart M} \tag{14}$$

$$\text{Fonction objectif : minimiser } M \tag{15}$$

4.3.3 Modèle F3

$$\sum_{j \in J} x_{i,j} \geq 1 \qquad \forall i \in I \tag{16}$$

$$S_{min} \leq S_j \leq S_{max} \quad \forall j \in J, \text{ variables d'écart } S_{min} \text{ et } S_{max} \tag{17}$$

$$T_{min} \leq T_j \leq T_{max} \quad \forall j \in J, \text{ variables d'écart } T_{min} \text{ et } T_{max} \tag{18}$$

$$\text{Fonction objectif : minimiser } S_{max} - S_{min} + k * (T_{max} - T_{min}) \tag{19}$$

4.3.4 Modèle F4

$$\sum_{j \in J} x_{i,j} \geq 1 \qquad \forall i \in I \tag{20}$$

$$S_{j1} + k * T_2 j \geq m \quad \forall j1, j2 \in J, \text{ variable d'écart m} \tag{21}$$

$$\text{Fonction objectif : maximiser } m \tag{22}$$

4.3.5 Choix du paramètre k

Le paramètre de pénalisation k nous permet de comparer le coût respectif, en terme de ressource informatiques, d'une requête DNS C_{req} et d'un calcul de signature cryptographique C_{sig}. Nous définissons donc $k = \frac{C_{sig}}{C_{req}}$.
Dans l'article [15], dans le paragraphe "2.2 DNSSEC architecture With DNSSEC", nous pouvons voir que si les réponses aux requêtes de vérifications de signatures sont stockées dans le cache, alors les serveurs peuvent traiter 3.33 fois plus de requêtes.
Ainsi, une possibilité pour le choix du paramètre de pénalisation est de fixer $k = 3.33$. C'est le choix que nous avons fait tout au long de cette étude.

4.4 Résolution du problème

Une fois les modèles définies, nous avons cherché les solutions optimales grâce au solveur open-source GLPK.

4.4.1 Comparaison des modèles

Nous avons comparé les modèless F1 à F4 sur des instances de test de petite taille. Les résultats sont résumés dans la figure 4.

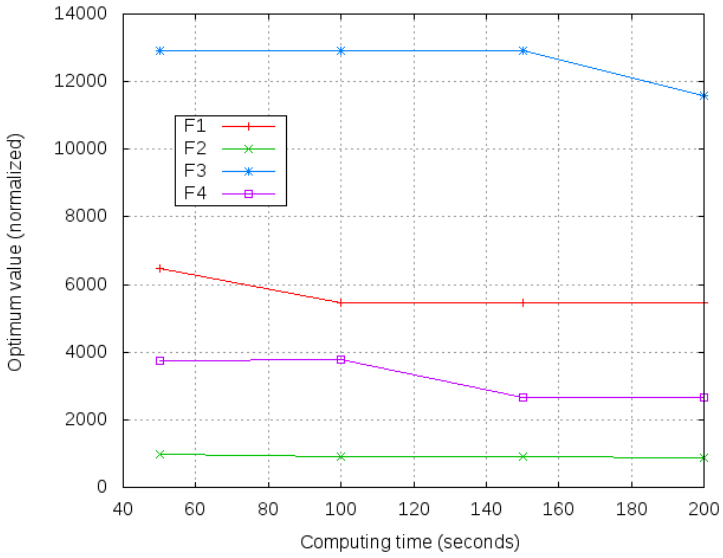

FIGURE 4 – La modélisation F2 (en vert) se détache nettement des 3 autres

A la suite des tests, on a pu conclure, comme le montre la figure 4, que la modélisation F2 est largement plus performante que les 3 autres modèles. Nous l'avons donc naturellement retenue pour les tests sur des plus grandes instances. Notons que ce résultat est conforme à l'étude menée dans [7].

4.4.2 Résolution du problème sur des plus grandes tailles avec le modèle F2

Dans la suite de l'étude, nous avons fixé le nombre de serveurs à $\|J\| = 10$. Dans la figure 5, nous pouvons voir que GLPK instancié avec le modèle F2 sur une taille de données de 200 FQDN converge vite vers une borne. Ainsi, la vitesse de convergence de l'algorithme est rapide, et le gain résiduel est faible après la 200^{eme} seconde.

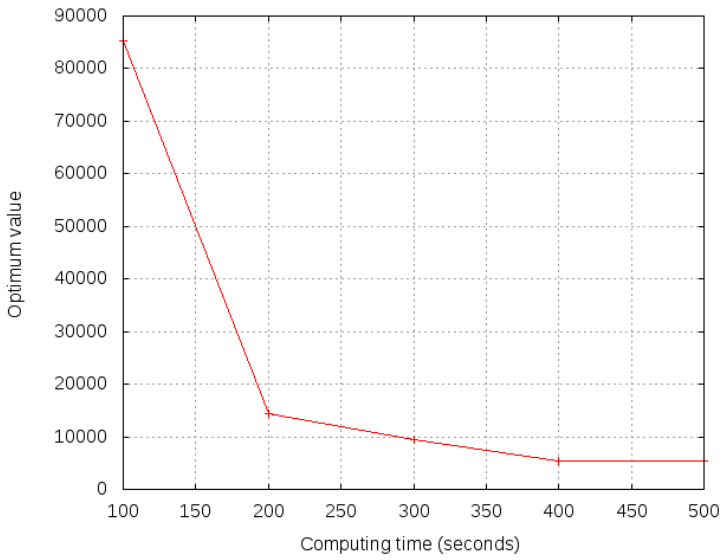

FIGURE 5 – Vitesse de convergence de F2

Les figures 6 et 7 illustrent comment le déséquilibre croit en fonction du nombre de FQDN que doit traiter le MIP F2. Le déséquilibre est mesuré comme la différence entre le nombre de requêtes minimum et le nombre de requêtes maximum que reçoivent les serveurs ($max(S_{j1}) - min(S_{j2})$ pour j_1 et j_2 dans J).
Dans la figure 6, GLPK a été arrêté au bout de 200 secondes. Il apparait également une forte pente entre 200 et 300 FQDN correspondant à une taille de MIP où GLPK n'a pas eu le temps d'effectuer des branchements efficaces. La solution trouvée sur des instances de taille supèrieures à 300 FQDN est tellement mauvaise que nous avons dû afficher la valeur du déséquilibre logarithmiquement.
Le nombre maximum de FQDN que le MIP peut traiter en 200 secondes est 400. Au-delà, aucune solution réalisable entière n'est trouvée, seule la solution fractionnaire est donnée par le solveur.
Le déséquilibre trouvé pour 200 FQDN est 14397, et dans ce cas, la différence entre le nombre de FQDN que gère le serveur le plus chargé et le serveur le moins chargé est 4.

Dans la figure 7, GLPK a été arrêté au bout de 3600 secondes. Il apparait sur ce graphique un pic à 200 FQDN. Ce résultat est contre-intuitif et il nous semble qu'il correspond à la difficulté pour le solveur de gérer uniquement les plus gros FQDN. En effet, quand le nombre de FQDN augmente, le nombre de FQDN ayant des petits N_i augmente aussi, or ceux-ci sont moins problématiques à positionner et peuvent correspondre à des variables plus aisés à positionner. Après 300 secondes, la courbe redevient croissante, ce qui est

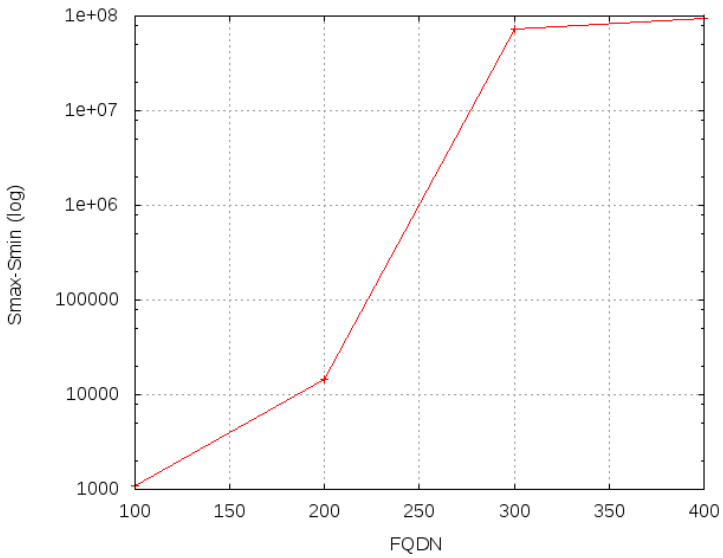

FIGURE 6 – Déséquilibre en fonction de la taille du problème

plus convenable.
Le nombre maximum de FQDN que le MIP peut traiter en 3600 secondes est 700. Au-delà, aucune solution réalisable entière n'est trouvée, seule la solution fractionnaire est donnée par le solveur.
Le déséquilibre trouvé pour 700 FQDN est 8865, et dans ce cas, la différence entre le nombre de FQDN que gère le serveur le plus chargé et le serveur le moins chargé est 19.

4.4.3 Evaluation

A l'issu de ces tests, nous pouvons évaluer les gains que l'utilisation des techniques décrites dans cet article nous ferait gagner. Nous allons retenir 2 scénarios où nous optimisons une partie de la répartition avec un MIP, et les comparer aux 2 scénarios où soit les FQDN, soit les requêtes sont en totalité répartis uniformément. Notons N_{sig} le nombre total de calculs de signatures effectué par l'ensemble des serveurs.

Scénario 1 : toutes les requêtes sont réparties uniformément entre les serveurs, comme dans le cas classique du DNS.
$Smin - Smax = 0$ (environ) et $Nsig = 2493964$.

Scénario 2 : tous les FQDN sont répartis uniformément entre les serveurs.
$Smin - Smax = 805329$ et $Nsig = \|I\| = 1211880$.

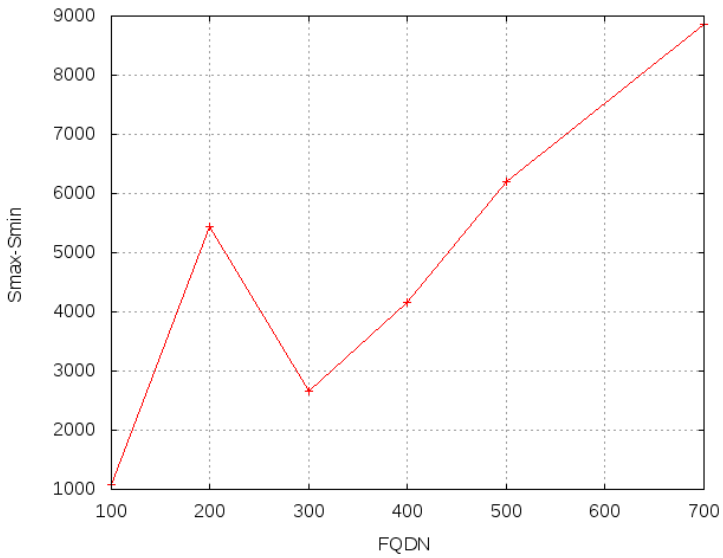

FIGURE 7 – Déséquilibre en fonction de la taille du problème

Scénario 3 : répartition des 200 FQDN les plus fréquents par MIP, répartition uniforme des suivants.
$Smin - Smax = 14397 + 141988 = 156386$ soit un déséquilibre 5 fois plus faibles que pour le scénario 2, et $Nsig = \|I\| = 1211880$ soit 2 fois moins de requêtes en signatures que le scénario 1. Ce scénario requiert 200 secondes de calcul.

Scénario 3 : répartition des 700 FQDN les plus fréquents par MIP, répartition uniforme des suivants.
$Smin - Smax = 8865 + 63588 = 72453$ soit un déséquilibre 11 fois plus faible que pour le scénario 2, et $Nsig = \|I\| = 1211880$ soit 2 fois moins de requêtes en signatures que le scénario 1. Ce scénario requiert 3600 secondes de calcul.

5 Conclusion

Nous avons décris dans cet article comment utiliser la répartition inégalitaire des requêtes DNS à notre avantage pour séparer le problème en deux sous problèmes dont chacun sera résolu de manière efficace. Nous sommes en mesure d'améliorer la solution de répartition de charge issu du DNS afin de la rendre performante sur une architecture DNSSEC. L'amélioration est significative en terme de consommation électrique car un grand nombre de calcul cryptographiques gourmands en ressources CPU peuvent être évités.

Grâce à cette la solution décrite dans cet article ,nous nous pouvons également limiter le nombre de serveurs à déployer car comme la charge sera mieux répartie, les serveurs atteindrons plus leurs limites de capacités.
Avec peu de ressources (200 ou 3600 secondes selon l'amélioration souhaitée), nous obtenons une amélioration notable.

Afin de déployer une solution pérenne, il conviendrait d'étudier plus avant comment la notoriété des requêtes DNS varie dans le temps. Ceci permettrait de déterminer la fréquence à laquelle faire tourner le solveur de MIP sur un jeu de données afin de le recalibrer.
Une autre suite intéressante à ce travail serait d'améliorer les modèles et étudier les possibilités d'algorithmes combinatoires tirant parti de la répartition semblable à une loi de puissance afin d'optimiser plus efficacement et ainsi diminuer les ressources informatiques nécessaires à l'administration de serveurs DNSSEC.

Références

[1] L.A. Adamic and B.A. Huberman. The Nature of Markets in the World Wide Web. *Quaterly Journal of Electronic Commerce*, 1 :512, 2000.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), March 2005.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard), March 2005. Updated by RFC 4470.

[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard), March 2005. Updated by RFC 4470.

[5] O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of Bundle and Classical Column Generation. *Math. Program.*, 113(2) :299–344, 2008.

[6] A. Clauset, C.R. Shalizi, and M.E.J. Newman. Power-Law Distribution in Empirical Data. *SIAM reviews*, june 2007.

[7] B. Decocq. *Résolution d'un problème de placement de processus sur calculateurs et de fichiers sur disques à EDF.* PhD thesis, CNAM/EDF, 1996.

[8] C. Gini. Variability and Mutability, 1912.

[9] M.L. Goldstein, S.A. Morris, and G.G. Yen. Problems with Fitting to the Power-Law Distribution. *The European Physical Journal B - Condensed Matter and Complex Systems*, 41(2) :255–258, 2004.

[10] J.K. Karlof. *Integer programming : theory and practice*. Operations research series. Taylor & Francis/CRC Press, 2006.

[11] D. Migault. Performance Measurements With DNS/DNSSEC. IEPG/IETF79, November 2010.

[12] D. Migault, C. Girard, and M. Laurent. A Performance view on DNSSEC migration, November 2010.

[13] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592.

[14] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658, 4033, 4034, 4035, 4343.

[15] Q. Xu, D. Migault, S. Sénécal, and S. Francfort. K-means and adaptative k-means algorithms for clustering dns traffic. Submitted to Valuetools Conference on Performance Evaluation Methodology and Tools, 2011.

[16] S. Yizhaki. Calculating Jacknife Variance Estimators for Parameters of the Gini Method. *Journal of Business & Economic Statistics*, 9(2), April 1991.