Génération de seuils optimaux dans une file hystérésis : application à un modèle de cloud

Thomas Tournaire, Hind Castel-Taleb, Emmanuel Hyon, Toussaint Hoche

Génération de seuils optimaux dans une file hystérésis : application à un modèle de cloud

T. Tournaire, [2,3] H. Castel-Taleb, [2] E. Hyon [1] et T. Hoche [4 †]

[1] *Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 Paris UMR 7606, 4 place Jussieu 75005 Paris France*
[2] *SAMOVAR, Télécom SudParis, CNRS, 9 rue Charles Fourier-91011 Evry Cedex, France*
[3] *Nokia Bell Labs, Nozay, France*
[4] *DAVID, Université Versailles Saint-Quentin en Yvelines, 45 av. des Etats-Unis, Versailles, France*

Nous proposons dans cette étude des méthodes d'optimisation efficaces pour la recherche de valeurs de seuils dans une file d'attente multi-serveur hystérésis. Il s'agit de minimiser un coût global prenant en compte les performances et l'utilisation des ressources et se calculant comme une fonction de coût sur la distribution stationnaire. Ce coût global étant une fonction non convexe, la recherche de la valeur optimale est complexe. Nous proposons deux types de méthodes d'optimisation : l'une est basée sur des heuristiques, couplées à de l'agrégation de chaînes de Markov pour réduire les temps d'exécution et l'autre est une méta-heuristique plus précisément le recuit simulé.

Mots-clefs : files d'attente hystérésis, évaluation des performances, consommation d'énergie, optimisation

1 Introduction

Nous considérons un modèle de files d'attente de type hystérésis, où les serveurs sont activés et désactivés selon l'occupation de la file. Ce modèle permet de représenter la variabilité des ressources en fonction de la demande [SPS+15, LG99] dans un cloud. Les requêtes arrivent, selon un processus de Poisson de taux λ, dans un système de file d'attente de capacité B et sont servies par un ensemble de serveurs (représentant dans notre étude des machines virtuelles ou VMs), dont le nombre total est K. Le temps de service d'un serveur est exponentiel de taux μ. On définit K niveaux de fonctionnement correspondant au nombre de serveurs (ou VMs) actifs : le nombre de serveurs actifs à un niveau $1 \leq k \leq K$, est de k. Le modèle est caractérisé par un vecteur de seuils d'activation $F = [F_1, F_2, ..., F_{K-1}]$, où F_k pour $1 \leq k < K$ fait passer du niveau k au niveau $k+1$, et un vecteur de seuils de désactivation $R = [R_1, R_2, ..., R_{K-1}]$, où R_k pour $1 \leq k < K$ fait passer du niveau k au niveau $k-1$. On supposera également que $F_1 < F_2 < ... < F_{K-1} < B$, que $R_1 < R_2 < ... < R_{K-1}$ et que $R_k < F_k, \forall k, 1 \leq k < K$. L'évolution du système est décrite par une chaîne de Markov à temps continu $\{X(t)\}_{t \geq 0}$ (voir Figure 1), dont chaque état est représenté par le couple (m,k) où m est le nombre de clients dans le système et k le nombre de serveurs activés.

Pour un jeu de seuils donné, nous définissons le coût global moyen à minimiser qui prend en compte les performances et l'utilisation des ressources. Il est calculé à partir de la distribution stationnaire π de la chaîne de Markov $\{X(t)\}_{t \geq 0}$:

$$\overline{C}^{\pi}_{[F,R]} = \sum_{k=1}^{K} \sum_{m=R_{k-1}+1}^{F_k} \pi(m,k) \cdot \overline{C}(m,k) \tag{1}$$

où $\overline{C}(m,k) = C_H \cdot m + C_S \cdot k + C_A \cdot \lambda \cdot \mathbb{1}_{m=F_k,\, k<K} + C_D \cdot \mu \cdot min\{m,k\} \cdot \mathbb{1}_{m=R_{k-1}+1,\, 2 \leq k \leq K} + C_R \cdot \lambda \cdot \mathbb{1}_{m=B,\, k=K}$, sachant que certains coûts sont liés aux performances : C_H est le coût par unité de temps du maintien d'un client dans la file et C_R est le coût de perte d'un client, tandis que d'autres sont liés à l'utilisation des ressources et à leur consommation énergétique : C_S le coût d'utilisation d'un serveur par unité de temps, C_A le coût d'activation d'un serveur , C_D le coût de désactivation.

†Ce travail est soutenu par le projet de recherche BOBIBEE du programme PGMO de la fondation Hadamard

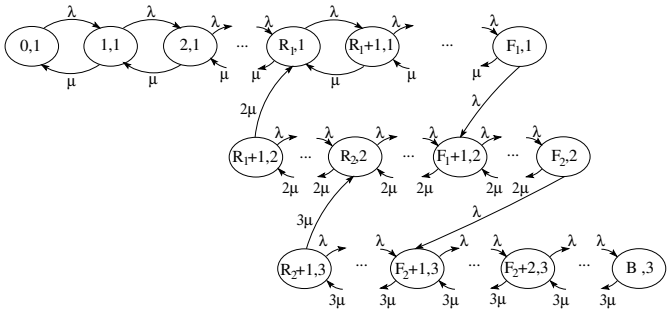

FIGURE 1: Schéma du modèle avec K=3 serveurs et B clients

L'objectif de ce travail est de proposer plusieurs heuristiques de recherche locale qui déterminent les vecteurs de seuils optimaux donnant le coût global minimum ; de comparer leur efficacité et de les comparer avec une méta-heuristique de type recuit simulé. De plus, nous utilisons des techniques d'agrégation, afin d'améliorer le temps d'exécution des heuristiques, en ne recalculant qu'une partie des coûts.

2 Description de la méthode de décomposition et calcul de la mesure globale

Lorsque le nombre de niveaux K devient grand, la résolution de la chaîne de Markov devient complexe : s'il existe bien une forme close, son calcul souffre d'instabilité numérique. C'est pourquoi, pour résoudre la chaîne de Markov nous appliquons une décomposition basée sur la méthode Stochastic Complement Analysis (SCA) [LG99] dans laquelle on partage la chaîne de Markov globale à K niveaux en K sous chaînes indépendantes (appelées micro-chaînes). On définit également une chaîne de Markov agrégée (appelée macro-chaîne) à K états, où chaque état représente un niveau de fonctionnement.

2.1 Description des micros-chaînes et de la macro chaîne

Chacune des K micro-chaînes garde les mêmes transitions que la chaîne de Markov exacte, sauf les transitions entre les niveaux qui sont coupées. Par contre, on rajoute ces transitions au niveau des micro-chaînes. Par exemple, pour la micro-chaîne de niveau 2 dans la figure 1, on garde les taux de transitions λ et 2μ entre les états et on rajoute la transition λ entre l'état $(F_2,2)$ et $(R_2,2)$ ainsi que la transition 2μ entre l'état $(R_1+1,2)$ et $(F_1+1,2)$. De cette manière chaque micro-chaîne est indépendante des autres. Soit π_k les probabilités stationnaires des micro-chaînes de niveau k pour chaque niveau $1 \leq k \leq K$, chaque distribution de probabilité stationnaire π_k peut-être résolue indépendamment. Quant à la macro-chaîne, elle est décrite par une chaîne à K états où chaque état k représente le niveau $1 \leq k \leq K$ et il s'agit d'un processus de naissance et de mort. Soit Π la distribution stationnaire de la macro-chaîne, son calcul est classique. Finalement, on obtient la distribution π de la chaîne initiale [LG99] par la formule :

$$\forall\ (m,k) \in \mathcal{E},\ \pi(m,k)\ =\ \Pi(k)\cdot\pi_k(m)\,. \tag{2}$$

2.2 Calcul du coût global sur la chaîne agrégée

La méthode d'agrégation décrite en 2.1, permet d'exprimer le coût global $\overline{C}^{\pi}_{[F,R]}$ de l'équation (1) en fonction des coûts par niveaux, calculés sur les micro-chaînes.

Theorem 1. *Soit $\overline{C}(k)$ le coût moyen du niveau k, le coût moyen global $\overline{C}^{\pi}_{[F,R]}$ du système peut s'écrire sous la forme :*

$$\overline{C}^{\pi}_{[F,R]}\ =\ \sum_{k=1}^{K}\ \Pi(k)\cdot\overline{C}(k)\ \ où\ \ \overline{C}(k)\ =\ \sum_{m=R_{k-1}+1}^{F_k}\ \pi_k(m)\cdot\overline{C}(m,k) \tag{3}$$

La preuve est directe : dans l'équation 1 on remplace la distribution stationnaire donnée par l'équation 2. Cette expression agrégée du coût par niveau, couplée à une propriété sur la modification des seuils, peut être mise à profit pour limiter la quantité de calculs du coût moyen.

Corollary 1. *Soit les vecteur $[F,R]$ fixés. On suppose que les micro-chaînes ainsi que les coûts par niveau sont calculés. La modification d'un seuil F_k ou R_k dans la chaîne de Markov n'a d'impact que sur les niveaux k et $k+1$. Aussi, le calcul du coût moyen du nouveau système ne nécessite uniquement le recalcul de π_k, π_{k+1} , $\overline{C}(k)$, $\overline{C}(k+1)$ et Π.*

La preuve de ce corollaire peut s'illustrer facilement, en observant la chaîne de la figure 1 : si on modifie F_2, cela aura un impact que sur les distributions π_2 et π_3 et donc on recalcule les coûts des niveaux $\overline{C}(2)$ et $\overline{C}(3)$, mais pas ceux des autres niveaux lors du calcul du coût global.

3 Algorithmes pour la recherche de seuils optimaux

3.1 Heuristiques

Pour des paramètres fixés $(\lambda,\mu,C_a,C_d,C_h,C_s,C_r,K,B)$ nous cherchons la politique optimale définie par les seuils qui minimisent le coût global moyen. Nous proposons différentes heuristiques de recherche locale qui sont des algorithmes itératifs testant différents jeux de seuils. Ce sont des méthodes de descente qui s'arrêtent dès qu'il n'y a plus d'améliorations dans un voisinage. On introduit la fonction *solve*$[F,R]$ qui, pour un vecteur de seuil donné, génère sa chaîne de Markov associée, calcule sa distribution stationnaire avec la méthode SCA et calcule le coût moyen associé $\overline{C}^{\pi}_{[F,R]}$ (1).

L'**Algorithme 3** est une heuristique classique de recherche par voisinage où le voisinage est défini par les jeux de seuils ayant eu une modification de $\pm$ 1 sur un des seuils :

1. **Initialiser** aléatoirement un vecteur de seuils $[F,R]_0$
2. Appliquer **solve**$[F,R]$ pour le vecteur de seuils courant
3. **Parcourir** le voisinage du vecteur de seuils courant
    4. Pour tout élément $[F,R]_i$ appartenant au voisinage : **solve**$[F,R]_i$
    5. **Stocker** le coût et le vecteur de seuil associé si amélioration pour chaque voisin parcouru
6. **Refaire** les étapes 3 à 5, jusqu'à ce qu'il n'y ait plus d'améliorations du coût dans un voisinage
7. **Retourner** C^* et $[F,R]^*$

Les deux autres heuristiques proposées (**Algorithme 1**, **Algorithme 2**) se distinguent par le choix du voisinage, par la manière dont on parcourt le voisinage mais aussi par l'initialisation du jeu de seuils.

3.1.1 Algorithme avec agrégation

Le corollaire 1 va induire un gain algorithmique important lors de l'exploration d'un voisinage. A partir d'un vecteur de seuils initial, on calcule le coût moyen à l'aide du théorème 1 et on stocke les distributions stationnaires des micro-chaînes ainsi que les coûts moyen de chaque niveau. Un voisin se caractérise par la modification d'un seuil d'un niveau k du vecteur initial. L'évaluation du coût global moyen d'un voisin sera effectuée plus rapidement car il ne faut recalculer que les distributions stationnaires et les coûts moyens des niveaux k et $k+1$, puis la distribution stationnaire de la macro-chaîne.

3.2 Approche d'une méta-heuristique : le Recuit Simulé

L'analyse des simulations nous a permis de confirmer la non convexité de la fonction de coût et ainsi de montrer que les heuristiques pouvaient rester bloquées dans un minimum local. Pour empêcher cela nous introduisons une méta-heuristique qui a pour but d'empêcher cette stagnation dans un minimum local. Parmi l'ensemble des méta-heuristiques disponibles nous choisissons l'une des plus répandues : le recuit simulé [Teg13]. Cette méthode autorise à sortir d'un minimum local en acceptant avec une certaine probabilité, qui décroit au cours du temps, une solution avec un coût moins bon. Cette méta heuristique nécessite d'adapter les paramètres de l'algorithme afin de converger raisonnablement vite.

4 Résultats Numériques

Nous présentons les temps d'exécution et l'efficacité des différents algorithmes pour différentes tailles de systèmes. Les heuristiques sont lancées sur un grand nombre d'instances choisies arbitrairement. La précision des calculs est fixée à 10^{-6} ce qui compense l'erreur due aux résolutions approchées des chaînes.

Les résultats pour un système à faible échelle (avec très peu de VMs et de clients) sont donnés **Table 1**. Les heuristiques ont été comparées à une recherche exhaustive (colonne "% Optimal") pour mesurer la proportion où le minimum global est atteint. Pour des systèmes à large échelle, **Table 2**, le temps d'exécution de la recherche exhaustive explose, aussi la colonne "% Minimum" mesure la proportion où un algorithme obtient la plus petite valeur. En terme d'optimalité et de temps, on constate que ces heuristiques sont très

-	Temps Moyen	Temps Maximal	% Optimal
Recherche Exhaustive	4,425 sec	8.839 sec	100 %
Algorithme 1	0,029 sec	0,516 sec	95,76 %
Algorithme 1 Agg	0,02 sec	0,507 sec	95,76 %
Algorithme 2	0.005 sec	0.057 sec	93,27 %
Algorithme 2 Agg	0,004 sec	0,038 sec	93,27 %
Algorithme 3	0.04 sec	0.194 sec	85,03 %
Algortithme 3 Agg	0,032 sec	0,171 sec	85,03 %
Recuit Simulé	2,45 sec	8,67 sec	95,40 %

TABLE 1: Comparaison des temps et de l'optimalité pour 46656 instances sur le cas K=3 et B=20

-	Temps Moyen	Temps Maximal	% Minimum
Algorithme 1	250 sec	1100 sec	65,27 %
Algorithme 1 Agg	52 sec	288 sec	65,27 %
Algorithme 2	0.12 sec	0.43 sec	72,22 %
Algorithme 2 Agg	0,033 sec	0,103 sec	72,22 %
Algorithme 3	96 sec	758 sec	15,27 %
Algorithme 3 Agg	22 sec	171 sec	15,27 %
Recuit Simulé	47 sec	327 sec	7 %

TABLE 2: Comparaison des temps et de l'optimalité pour 144 instances sur le cas K=16 et B=200

efficaces pour un système à faible échelle mais que l'efficacité de ces algorithmes se dégrade plus le système est large, notamment parce que le nombre de jeux de seuils parcourus par les heuristiques par rapport au nombre de jeux de seuils total diminue et donc qu'on explore moins de candidats. On remarque également le gain de temps significatif obtenu grâce à l'agrégation du coût intégré dans les algorithmes. Quant au Recuit Simulé, on note que, même sur un système à faible échelle et après optimisation des paramètres de l'algorithme, cette méthode souffre d'un surcoût temporel afin d'atteindre l'optimum global.

5 Conclusion

Nous avons montré dans cette étude que l'approche par agrégation permettait des gains substantiels en temps de calcul pour une même efficacité et que les approches par recuit simulé ne sont pas intéressantes en pratique. Pour la suite, nous prévoyons de définir des modèles avec des coûts basés sur des consommations énergétiques réelles [BLOR18] et sur des exemples concrets de SLA. Nous prévoyons aussi d'étudier des modèles avec taux d'arrivées non statiques à partir de mesures réelles des requêtes dans des datacenters.

Références

[BLOR18] A. Benoit, L. Lefèvre, A.-C. Orgerie, and I. Rais. Reducing the energy consumption of large scale computing systems through combined shutdown policies with multiple constraints. *Int. J. High Perform. Comput. Appl.*, 32(1) :176–188, 2018.

[LG99] J. C. S. Lui and L. Golubchik. Stochastic complement analysis of multi-server threshold queues with hysteresis. *Performance Evaluation*, 35 :19–48, 1999.

[SPS+15] S.Y. Shorgin, A. V. Pechinkin, K. E. Samouylov, Y. V. Gaidamaka, I. A. Gudkova, and E. S. Sopin. Threshold-based queuing system for performance analysis of cloud computing system with dynamic scaling. *AIP Conference Proceedings*, 1648(1), 2015.

[Teg13] J. Teghem. *Recherche opérationnelle - Tome 1*, volume 1. Ellipse, 2013.