HOTS : allocation de ressources pour conteneurs par une méthode hybride machine learning - optimisation.

Etienne Leclercq[1], Jonathan Rivalan[1]

Smile, France

{etienne.leclercq, jonathan.rivalan}@smile.fr

1 Introduction

Dans l'univers du *cloud computing*, la gestion des ressources est primordiale car elle met en jeu deux aspects opérationnels majeurs : la performance des applications et leur niveau de consommation énergétique. L'émergence de la virtualisation en conteneurs micro-services, et de l'élasticité de la consommation inhérente à cette technologie, a apporté de nouvelles contraintes à ce problème d'allocation de ressources [1].

Une grande partie des méthodes existantes dans le domaine de l'optimisation des ressources proposent soit une solution de placement des conteneurs *a priori*, sans adapter celle-ci à l'évolution des consommations de ces conteneurs [3] ; soit l'adaptation des ressources allouées aux conteneurs, mais pas leur placement [2].

Nous avons développé une méthode alliant apprentissage et optimisation, identifiant des profils de consommation pour guider le placement des conteneurs. Les profils et le placement de ces micro-services sont mis à jour en fonction de l'évolution des consommations grâce à une boucle utilisant notamment des propriétés de dualité.

Cette méthode a été implémentée à travers une librairie Python permettant l'évaluation de cette méthode. À travers ces expérimentations, nous analysons l'efficacité globale de la méthode via l'identification de critères métiers, mais également son utilisabilité dans un environnement réel de production ainsi que l'impact de l'évolution du clustering sur la solution de placement.

2 Méthodologie complète

Afin d'obtenir des solutions de placement adaptées à l'évolution de la consommation des conteneurs, nous élaborons une stratégie de placement en utilisant les profils de consommation de ces individus, grâce à une étape de clustering. Ce problème de clustering est modélisé par un PLNE, dont certaines variables de décision sont utilisées ensuite, en les considérant comme des données, dans la modélisation du problème de placement par un PLNE également.

Afin de répondre à la dynamicité de nos problèmes de clustering et de placement, nous incorporons dans ces modèles mathématiques l'information des solutions courantes (respectivement de clustering et de placement) à travers des contraintes (respectivement $mustLink_C$ et $mustLink_A$) permettant de fixer les variables de décision liant les conteneurs entre eux (respectivement dans un même cluster et dans un même nœud). L'évolution des valeurs des variables duales associées à ces contraintes permettent de remettre en cause les solutions existantes, en construisant un graphe de conflit pour chacun de nos problèmes (clustering et placement). Ces graphes de conflit permettent d'obtenir des listes de conteneurs qui sont susceptibles de changer respectivement de cluster et de nœud, en ayant une vision globale des conflits levés. Ces changements sont réalisés par des heuristiques de replacement, vérifiant l'amélioration effective de la solution en cas de changement. Ces mécanismes sont implémentés à travers une boucle, s'exécutant en continu à l'arrivée de nouvelles données. Cette boucle suit un fonctionnement à deux niveaux :

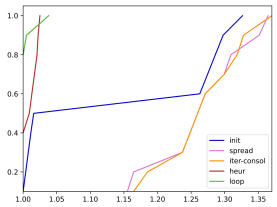
— la mise à jour du clustering en fonction des nouvelles données de consommation ;
— l'intégration des changements de clustering dans la modélisation du placement, permettant à son tour sa mise à jour.

La construction des graphes de conflits se fait via la résolution de programmes linéaires, dont la construction n'est réalisée qu'une seule fois (mise à jour des données du programme à chaque itération de la boucle) et dont la résolution successive d'une itération à une autre permet l'utilisation de la précédente résolution pour accélérer la résolution du problème actuel. Les heuristiques de modifications des solutions de clustering puis de placement à partir des listes de réassignation (respectivement L_C et L_A) ont une complexité respectivement en $O(|L_C|K)$ et en $O(|L_A|M)$. Ces étapes permettent la modification rapide des solutions.

3 Implémentation et évaluation

La méthode décrite précédemment a été implémentée à travers un module Python appelé *HOTS*, disponible publiquement sous licence open source [4], et suivant une architecture modulaire, laissant un grand choix de personnalisation à l'utilisateur (utilisation du solver souhaité, modification des problèmes d'optimisation, modification des heuristiques ...). Son implémentation a également ouvert à l'évaluation de cette méthode par comparaison à d'autres méthodes de l'état de l'art.

Cette évaluation a été réalisée en détaillant différents points : l'impact sur des critères métiers identifiés, visant l'évaluation de la performance intrinsèque de la méthode, ainsi que sur divers indicateurs visant l'analyse de l'impact du clustering sur le déroulement de la méthode et aux résultats obtenus. Par ailleurs, l'applicabilité de notre méthode a été étudiée en comparant les temps d'exécution avec la temporalité des jeux de données.



Method	*nodes*	*alerts*
initial	50	**0**
spread	50	**0**
iter-cons	46,8	3,21
heuristic	46,3	2,73
loop	**46**	0,20

FIG. 1 – Critères de stabilité (haut) et critères énergétiques (bas) sur jeux de données réels.

Avec cette évaluation, nous montrons une amélioration des critères de performance propres au métier, en particulier par l'économie énergétique représentée par l'utilisation d'un nombre réduit de serveurs, tout en maintenant une stabilité de charge sur ces derniers, ce qui réduit les risques d'incidents de production. Par exemple, la Figure 1 montre la supériorité de notre méthode *loop* sur deux critères de stabilité (amplitude maximum et variance moyenne de consommation totale des nœuds) et deux critères énergétiques (nombre de noeuds et nombre d'alertes déclenchées) : en moyenne notre méthode présente un pourcentage d'amélioration de 6,3% sur l'ensemble des critères. Nous montrons également que notre méthode originale de mise à jour du clustering reste cohérente du point de vue de critères intrinsèques au problème de clustering, en la comparant à d'autres méthodes de clustering de données dynamiques. De plus, les indicateurs propres au problème métier de placement, pour lequel notre modélisation du clustering a été pensée, montrent l'intérêt de cette méthode de mise à jour en ligne : avec notre méthode, les critères métiers sont améliorés et les solutions obtenues sont plus stables dans le temps.

Références

[1] Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, and Philippe Merle. Elasticity in cloud computing : state of the art and research challenges. *IEEE Transactions on Services Computing*, 11(2) :430–447, 2017.

[2] M. Fokaefs, C. Barna, R. Veleda, M. Litoiu, J. Wigglesworth, and R. Mateescu. Enabling devops for containerized data-intensive applications : An exploratory study. *Proceedings of the 26th Annual International Conference on Computer Science and Software Engineering*, pages 138–148, 2016.

[3] Anwar Hassen. *A survey of docker swarm scheduling strategies*, 2016.

[4] Smile. Hots : Hybrid optimization for time series, 2023. Available at https ://pypi.org/project/hots/.