Compas'2023 : Parallélisme/ Architecture/ Système
LISTIC/USMB - Annecy France, 4-7 juillet 2023

*télé*GC : Un garbage collector (GC) délocalisé grâce à la mémoire désagrégée

Adam Chader, Yohan Pipereau, Mathieu Bacou, Gaël Thomas

Télécom SudParis,
Institut Polytechnique de Paris
prenom.nom@telecom-sudparis.eu

Résumé
La désagrégation mémoire [7, 2, 1, 4] permet d'exécuter des applications gourmandes en mémoire, telles que des frameworks de big data [5, 6, 3] ou des bases de données in-memory, tout en utilisant la mémoire déjà disponible en grande quantité dans le rack. Contrairement à un système distribué, il n'est pas nécessaire de modifier le code ni de supporter le coût de sérialisation des transferts de données.
L'idée de la désagrégation est de créer un système informatique avec des nœuds spécifiques aux ressources : des machines pour le calcul et d'autres pour la mémoire. L'ensemble du système étant géré pour garder l'abstraction d'une machine monolithique. Les accès mémoire sur de tels systèmes deviennent ainsi distants, et pour atténuer le coût aux performances, il est fait un usage intensif de la mise en cache sur les nœuds de calcul.
Malheureusement, beaucoup d'applications ont une localité médiocre et fonctionnent donc mal avec la désagrégation. Particulièrement les Gargage Collectors (GC), qui sont une fonctionnalité clé de certains langages. En effet, les GCs doivent parcourir l'intégralité de la mémoire pour déterminer quels objets doivent être collectés, ils passent donc très mal à l'echelle pour de très grands tas, et polluent le cache.
Nous proposons *télé*GC, un runtime Java (JVM) modifié offrant une transparence totale avec du code Java non modifié, permettant une collecte non obtrusive d'objets sur de grands tas désagrégés. Ceci est rendu possible en déchargeant la collection sur les nœuds de mémoire, où se trouvent les données, plutôt que sur les nœuds de calcul. Cela permet de limiter la communication et le mouvement des données entre les nœuds lors de la collecte.
De plus, en utilisant la différence de domaine de cohérence de cache entre les nœuds de calcul et de mémoire, nous pouvons effectuer une collecte *Snapshot-at-the-begining* (SATB) permettant la collecte concurrente, sans coût supplémentaire. L'implémentation de notre GC est complètement déportée de la JVM et indépendante du langage, ce qui signifie que *télé*GCpourrait être utilisé pour collecter la mémoire d'autres langages que Java.

Mots-clés : Ramasse-miettes, Mémoire désagrégée

Bibliographie

1. Amaro (E.), Branner-Augmon (C.), Luo (Z.), Ousterhout (A.), Aguilera (M. K.), Panda (A.), Ratnasamy (S.) et Shenker (S.). – Can far memory improve job throughput ? – In *Proceedings*

of the Fifteenth European Conference on Computer Systems, *EuroSys '20*, EuroSys '20, New York, NY, USA, 2020. Association for Computing Machinery.

2. Gu (J.), Lee (Y.), Zhang (Y.), Chowdhury (M.) et Shin (K. G.). – Efficient memory disaggregation with infiniswap. – In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pp. 649–667, Boston, MA, mars 2017. USENIX Association.
3. Malewicz (G.), Austern (M. H.), Bik (A. J.), Dehnert (J. C.), Horn (I.), Leiser (N.) et Czajkowski (G.). – Pregel : A system for large-scale graph processing. – In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, *SIGMOD '10*, SIGMOD '10, p. 135–146, New York, NY, USA, 2010. Association for Computing Machinery.
4. Wang (C.), Ma (H.), Liu (S.), Li (Y.), Ruan (Z.), Nguyen (K.), Bond (M. D.), Netravali (R.), Kim (M.) et Xu (G. H.). – *Semeru : A Memory-Disaggregated Managed Runtime*. – USA, USENIX Association, 2020.
5. Zaharia (M.), Chowdhury (M.), Das (T.), Dave (A.), Ma (J.), McCauley (M.), Franklin (M. J.), Shenker (S.) et Stoica (I.). – Resilient distributed datasets : A fault-tolerant abstraction for in-memory cluster computing. – In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, *NSDI'12*, NSDI'12, p. 2, USA, 2012. USENIX Association.
6. Zaharia (M.), Chowdhury (M.), Franklin (M. J.), Shenker (S.) et Stoica (I.). – Spark : Cluster computing with working sets. – In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*, Boston, MA, juin 2010. USENIX Association.
7. Zhang (J.), Ding (Z.), Chen (Y.), Jia (X.), Yu (B.), Qi (Z.) et Guan (H.). – Giantvm : A type-ii hypervisor implementing many-to-one virtualization. – In *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, *VEE '20*, VEE '20, p. 30–44, New York, NY, USA, 2020. Association for Computing Machinery.