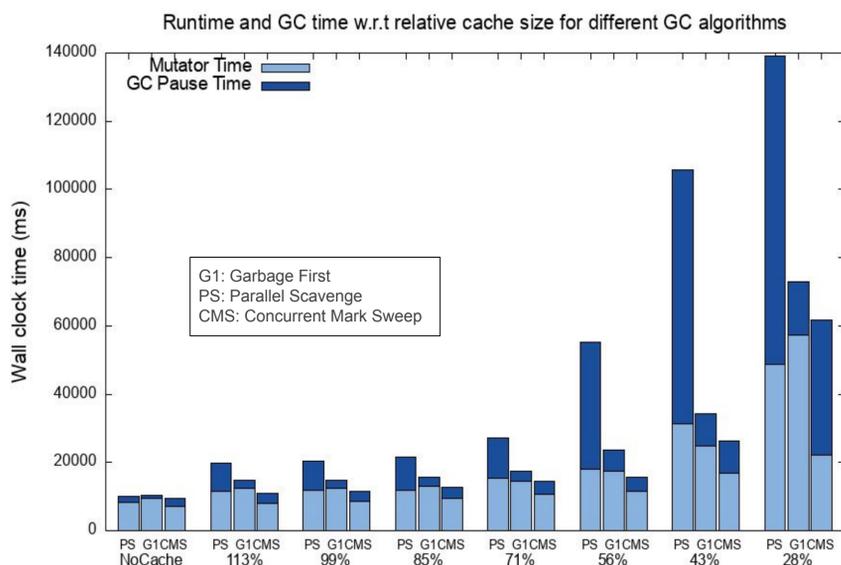
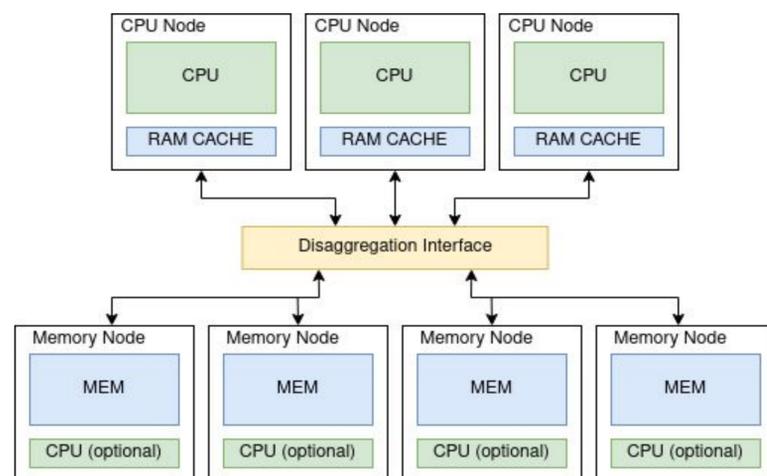


Auteurs

Adam Chader
Yohan Pipereau
Gaël Thomas
Mathieu Bacou

LA MÉMOIRE DÉSAGRÉGÉE ...

1. On **expose** le serveur monolithique **selon ses ressources**
2. Des **noeuds** dédiés à la **mémoire** (Memory Node), et d'autres au **calcul** (CPU Node)
3. Un serveur peut **faire varier** sa quantité de mémoire **à la volée**
4. Un **noeud de calcul** garde un peu de mémoire comme **cache**



... EST INCOMPATIBLE AVEC LES GC ...

Dépendance au cache

1. **Déporter la mémoire** à distance nuit aux **performances**
2. On utilise un **cache** pour mitiger ce coût
3. Les performances d'une application désagrégée dépendent beaucoup de sa **localité**

Les Garbage Collectors

1. Ils servent à **nettoyer la mémoire non utilisée** des applications
2. Pour cela, ils doivent parcourir **toute la mémoire**, ce qui **rend inutile un cache**
3. Pendant la collection, l'application doit se mettre en **pause**, ou se **synchroniser** pour ne pas créer de nouveaux objets

Les Garbage Collectors dégradent fortement les performances d'applications en mémoire désagrégée

LA SOLUTION : TÉLÉGC

Mettre le GC à l'endroit de la mémoire

1. On **déplace le GC** dans le noeud mémoire
2. Le GC ne passe plus par le cache, et lit directement sur la mémoire
3. Pas de pertes de performance à cause de la désagrégation

Domaines de cohérence de cache différents

1. L'**application**, localisée sur le noeud de calcul **agit sur le cache** et les écritures sont **périodiquement propagées sur la mémoire**
2. Le **GC**, sur le noeud mémoire **lit directement la mémoire (mais n'écrit pas)**
3. Le GC voit un **snapshot de la mémoire effective**

Intérêt du snapshot

1. On sait que la mémoire sur le noeud mémoire n'est pas modifiée **entre deux propagations du cache**
2. En **bloquant habilement cette propagation**, on peut **collecter** la mémoire en **s'assurant qu'elle ne sera pas modifiée**
3. C'est la **write-back barrier**

