Aurélien MONNET-PAQUET
Antoine THEBAUD
Groupe 2
Master1 INFO

CSE : TP Allocateur mémoire et mesures

1. Présentation des métriques évalués

Le but de cette expérience est de savoir comment se comporte les programmes vis à vis de la mémoire. Nous allons mesurer la fragmentation après une série d'allocations mémoire.

Les deux questions auxquelles ce rapport tente de répondre sont les suivantes :

Quelle est la fragmentation résultante de l'exécution de programmes simples utilisés tous les jours ?

La stratégie best fit vaut-elle le coup par rapport à une simple stratégie first fit ?

2. Description de l'expérience menée

L'allocateur mémoire qui est utilisé dans cette expérience est thread-safe.

Lors des différents test de fragmentation nous récupérons 3 informations :
- Le cumul de la mémoire allouée au programme (axe horizontal)
- La taille jusqu'au dernier bloc occupé (libre + occupé).
- La taille totale des blocs occupés uniquement.

Les données sont inscrites dans un fichier au fur et a mesure des allocations. A des fin d'analyse, ces données sont importées dans un fichier de type "Excel".

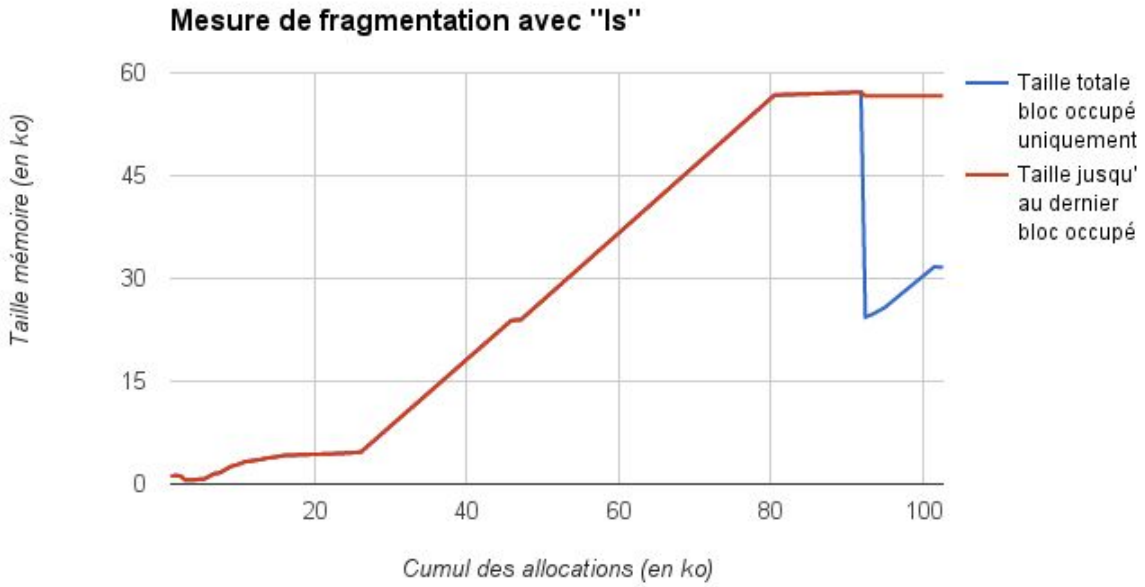
Pour la première partie la stratégie d'allocation utilisée est "first-fit". Le but de cette stratégie est de mettre a disposition le premier bloc mémoire assez grand pour le programme demandant de la mémoire.

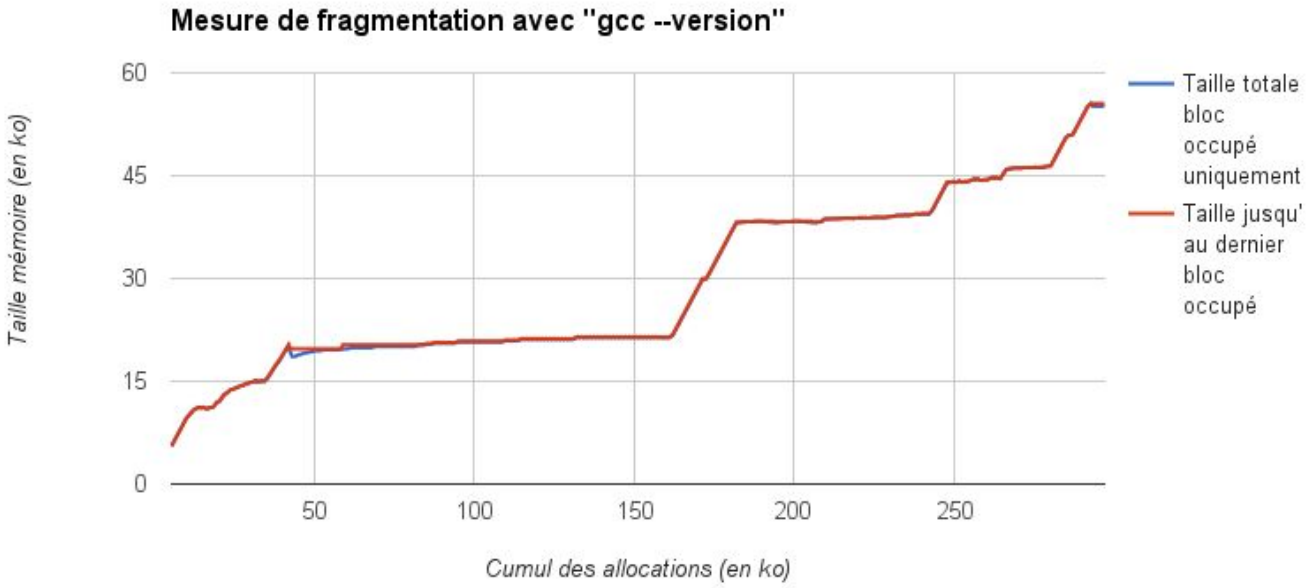
Un test a été réalisé pour chacune de ces commandes :
- "ls"
- "gcc --version"
- "make"

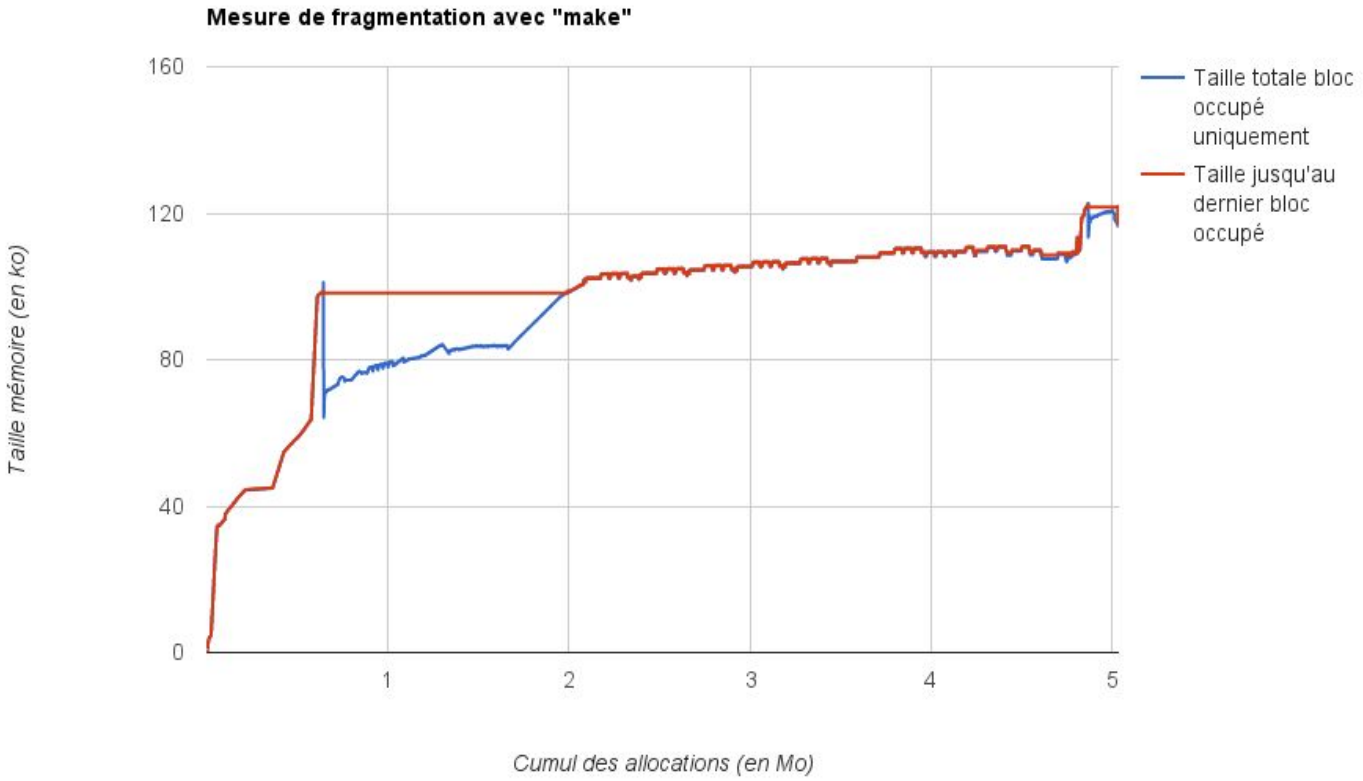
Dans une seconde partie nous testons un même programme avec deux stratégies d'allocation différentres : first fit et best fit. Ce programme consiste en une succession aléatoire d'allocations et de libérations d'espace mémoire.

3. Résultats obtenus

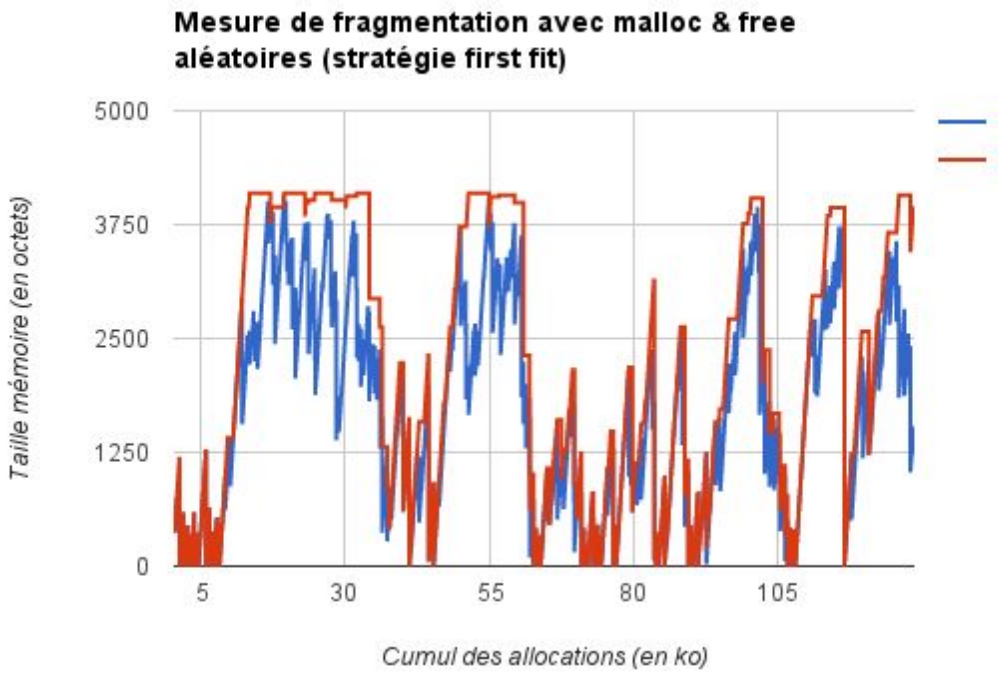
1) Mesure de fragmentation sur des programmes courants.

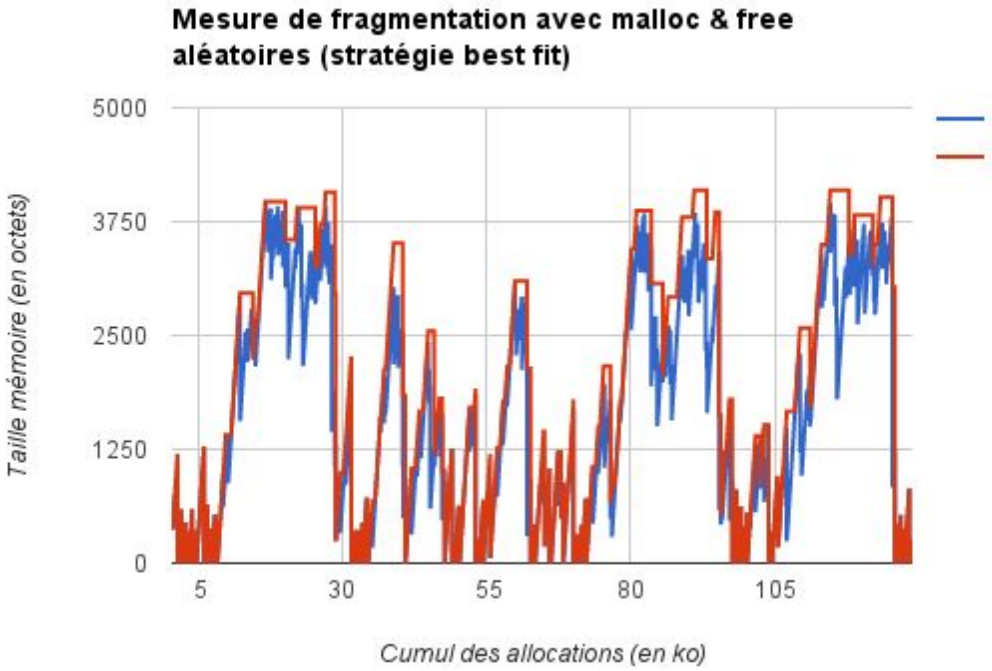






2) Mesure de l'influence de la stratégie best fit par rapport à first fit





4. Conclusion

1) Première partie

On constate que, pour des programmes courant comme ls, gcc et make, il y a finalement assez peu de fragmentation puisque les deux courbes sur les graphiques se chevauchent la plupart du temps. On relèvera quand même qu'il y a environ 50% de fragmentation pour la commande ls lorsqu'elle arrive à son terme (beaucoup d'espace libéré mais le dernier bloc occupé n'étant pas libéré, l'écart se creuse fortement entre les deux courbes). Et concernant "make" on peut aussi voir qu'à un certain stade de son exécution une grande partie de l'espace mémoire est libéré et il en résulte environ 33% de fragmentation. Néanmoins cette fragmentation se réduit très vite puisque plusieurs allocations son faites ensuite et au final les deux courbes se chevauchent à nouveau.

D'autre part on peut déduire de ses observations que la stratégie d'allocation choisie n'aura au final pas beaucoup d'influence pour ces 3 programmes. Cela s'explique par le fait que peu de free sont effectués, donc l'appel à la fonction d'allocation renvoie souvent le bloc libre correspondant à l'espace entre le dernier bloc alloué et la fin de la mémoire (càd le dernier bloc libre qu'il est possible d'avoir), et ce quel que soit la stratégie d'allocation choisie. La taille jusqu'au dernier bloc occupé et la taille réellement occupée sont donc souvent identiques, d'où le chevauchement des deux courbes. (Note : cette hypothèse a été vérifiée avec d'autres observations et d'autres courbes, non présentées ici).

2) Deuxième partie

Les deux courbes du graphique se chevauchent beaucoup plus souvent sur le graphique illustrant la stratégie best fit que sur celui illustrant la stratégie first fit. Là où la fragmentation dépasse plusieurs fois les 50% pour first fit, ce scénario ne se produit que quelques fois avec best fit, et dans des proportions moindres.

On constate donc que best fit est plus efficace, càd qu'il y a moins de fragmentation avec la cette stratégie qu'avec la stratégie first fit.

En conclusion, le temps CPU supplémentaire nécessaire à best fit pour trouver la meilleure zone à allouée résulte en une économie d'espace mémoire utilisé. La question est de donc savoir lequel de ses deux aspects (temps de calcul et espace mémoire) l'on veut optimiser.