Cours de Big Data

5A – Polytech Paris-Sud – 2023-2024

TD-2 : Analyse de performances et passage à l'échelle

Stéphane Vialle & Gianluca Quercini

Exercice 1 : Analyse des mesures de performances d'une application distribuée

La figure 1 montre des allures de temps d'exécutions d'une application distribuée, sur 3 clusters différents (cluster a (Ca), cluster b (Cb) et cluster c (Cc)), et pour 3 tailles successives de problèmes (N1, N2 et N3). Les échelles en abscisses et ordonnées sont logarithmiques.

Les 3 clusters ont des caractéristiques différentes en termes de puissance de calcul des nœuds et de capacité de communication des réseaux d'interconnexion. L'application utilise le même algorithme pour les trois tailles de données (avec N1 < N2 < N3).

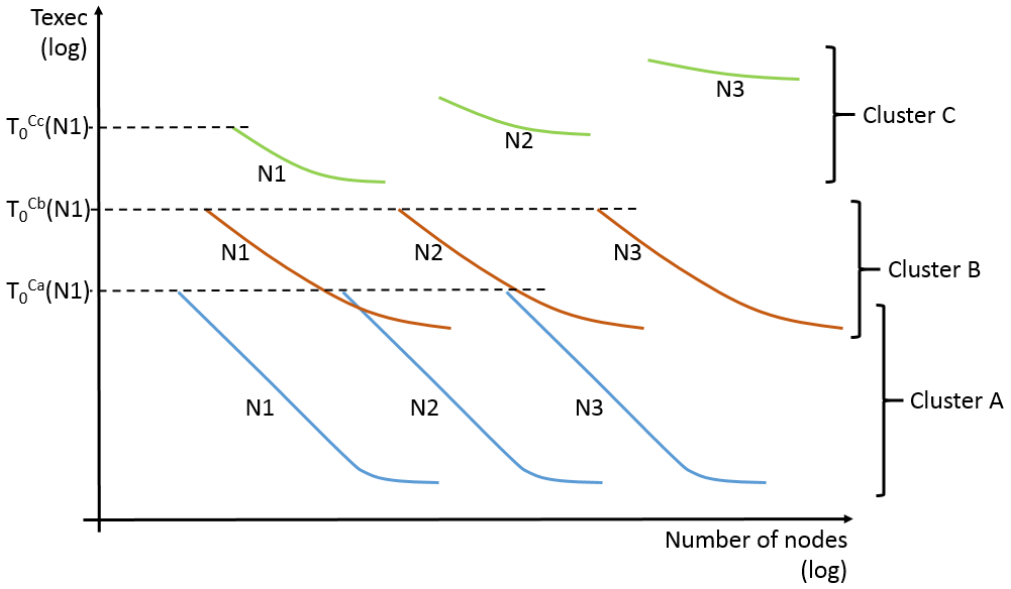


Figure 1 : temps d'exécution sur 3 clusters différents, et pour 3 tailles de problème

Question : Que pouvez-vous conclure de la capacité de passage à l'échelle de cette implantation distribuée ? et des caractéristiques relatives des 3 clusters ?

Exercice 2 : Analyse de la qualité d'un passage à l'échelle

Un code client distribué sur cluster de PC est indiqué, par le client, avec une complexité en $O(n^2)$. Ce client souhaite grossir le problème traité, et vous demande d'exécuter sur vos clusters des calculs de taille $2.n_0$, puis $4.n_{0,}$

Des expérimentations menées sur vos clusters ont fournis les superpositions de courbes de temps de la figure 2 pour 3 tailles de problèmes. On peut y observer :

- Les temps d'exécution à taille de problème constant (lignes en pointillés)
- Un temps d'exécution constant pour des problèmes de taille croissante (ligne pleine).

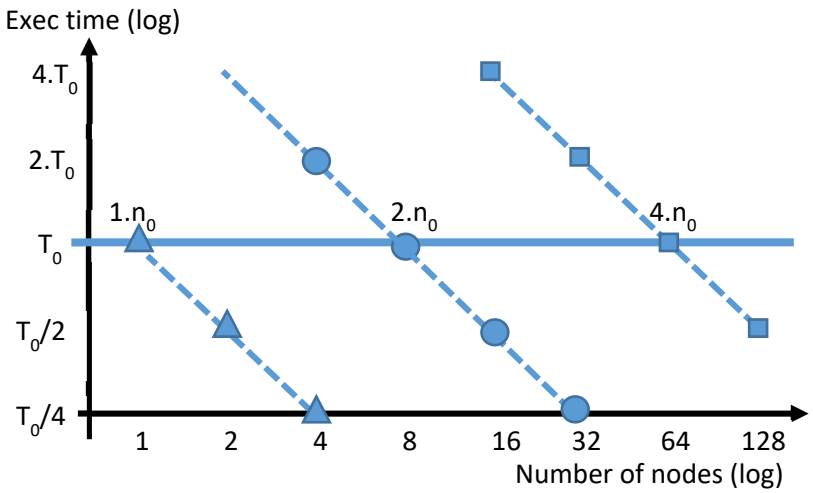


Figure 2 : abaque de temps d'exécution du code client

Question 1 : Les *speedup* sont-ils excellents, moyens, médiocres ? Justifiez votre réponse**.**

Question 2. Le *size-up* est-il celui attendu ? Pourquoi ?

Exercice 3 : Dimensionnement d'un nombre de ressources lors d'un passage à l'échelle

On considère un algorithme d'analyse de données mono-nœud et optimisé, mais limité à la mémoire et à la puissance de calcul d'une seule machine. Afin de traiter des données plus volumineuses on distribue cette application. Cependant l'adaptation de l'algorithme optimisé mono-nœud est profonde et entraîne un surcoût significatif de temps d'exécution sur 2 nœuds (T(N0,2)), avant d'entamer une décroissance régulière, comme illustré sur la figure 3.

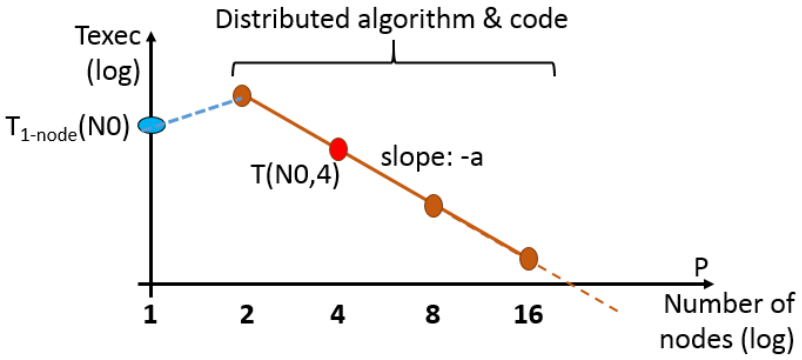


Figure 3 : Passage à une version distribuée avec surcoût initial sur 2 nœuds

Après mesure de T(N0,2) on note : $T(N0,2) = k.T_{1\text{-node}}(N0)$

Question 1 : Modélisez le temps d'exécution T(N0,P) de la figure 3, pour $P \geq 2$.

Question 2 : On suppose que les calculs dominants des deux versions de l'algorithme sont en $O(N^{3/2})$ et que le passage à l'échelle du code est parfait (hormis le surcoût d'un facteur k lors du passage à la version distribuée).

On suppose que les calculs dominants des deux versions de l'algorithme sont en $O(N^{3/2})$. On s'intéresse maintenant à un problème ayant deux fois plus de données : $N1 = 2 \times N0$. On souhaite conserver le temps de traitement obtenu avec N0 données sur $P_{u\ (usage)} = 4$ nœuds ($T(N1,P_x) = T(N0,P_u)$). Calculez le nombre de nœuds nécessaires (P_x) avec les deux hypothèses suivantes sur k = k(N) :

- $k(N) = k_0 = Cte$
- $k(N) = k_0 \times (N/N0)$

Faites des applications numériques avec : a = 1 (pente idéale, décroissance parfaite du temps d'exécution).

Question 3 : Tracez les deux nouvelles courbes de temps d'exécution pour $P \geq 2$ et pour N = N1, correspondant aux deux hypothèses sur k(N).

Exercice 4 : Analyse de performance sur machines dual-socket

On a réalisé des benchmarks en utilisant plus ou moins de cœurs logiques d'une machine à 2x10 cœurs physiques (2x20 cœurs logiques) dénommée « John3 », puis sur une autre machine à 2x16 cœurs physiques (2x32 cœurs logiques) dénommée « John4 ».

L'application est multithreadée, et réalise une « co-simulation » : une importante simulation agrégeant de nombreux composants de simulations. A chaque pas de temps, chaque composant réalise des calculs pour calculer « sa progression » et échange des résultats avec d'autres composants en fin de pas de temps.

Question 1 : Analyser le comportement de l'application sur John3 en fonction du nombre de cœurs alloués (figure 4). Que pensez-vous de la répartition des threads et de l'allocation des cœurs logiques jusqu'à 20 cœurs et au-delà ?

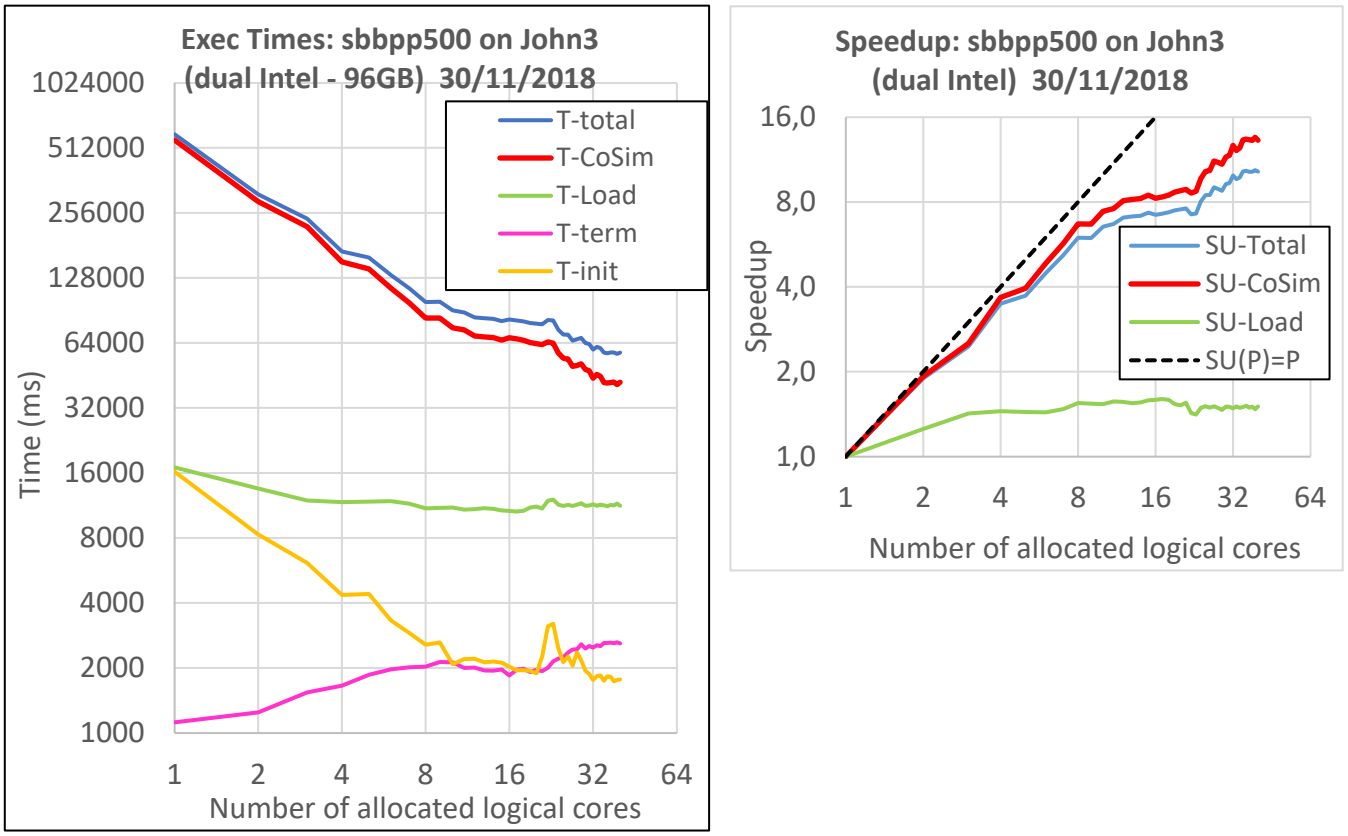
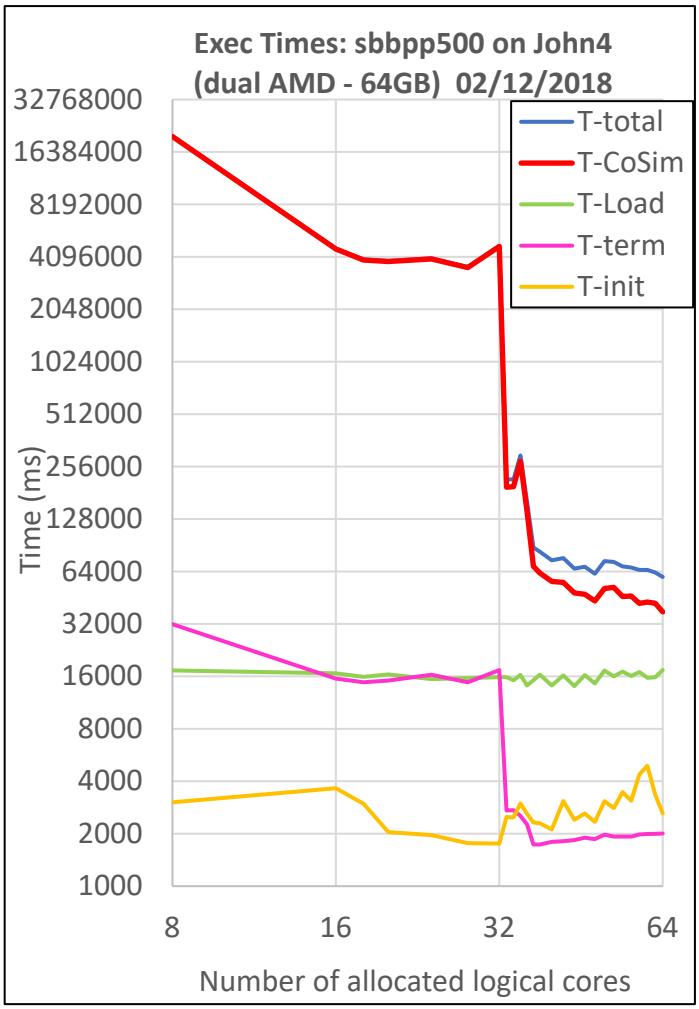


Figure 4 : temps d'exécution et Speedup sur John3

Question 2 : On relance la campagne de benchmarks sur John4, et on obtient les courbes de la figure 5. Comment analysez-vous le phénomène ? Comment expliquez-vous les courbes ?



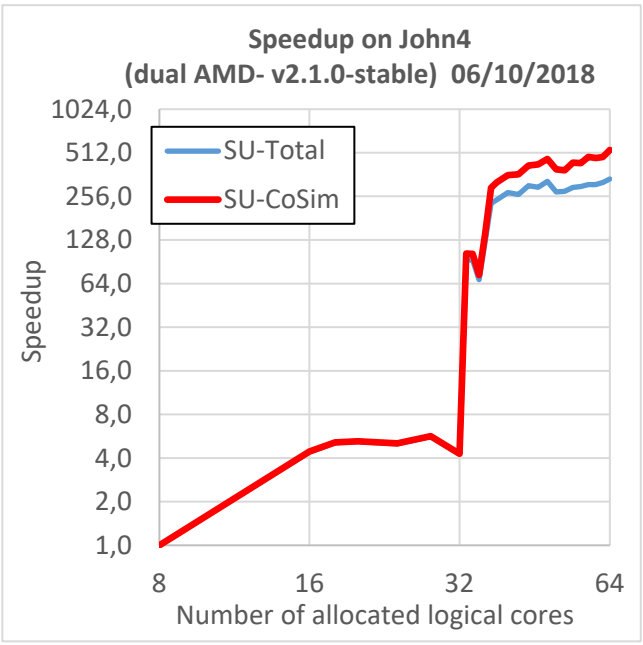


Figure 5 : temps d'exécution et Speedup sur John4