Ordonnancement sur une plate-forme hétérogène

Résumé: Ce TD propose une étude expérimentale d'un problème d'ordonnancement sur une plate-forme hétérogène.

Nous nous intéressons ici à l'étude de l'ordonnancement d'un grand nombre de tâches indépendantes sur une plate-forme hétérogène telle que celle qui est représentée figure 1. La question que l'on se pose est donc la suivante : «Étant données n tâches indépendantes et identiques (en terme de quantité de calcul et de communications nécessaires à leur traitement) détenues par un maître, comment les répartir au mieux sur les p autres machines ?». L'évaluation des performances des algorithmes est effectuée à l'aide du simulateur SimGrid. Toute la documentation concernant son utilisation est fournie dans `/home/alegrand/doc/simgrid2/` ou bien sur http://www.ens-lyon.fr/~alegrand/simgrid2.html.

Un prototype d'ordonnanceur extrêmement stupide est fourni dans `/home/alegrand/MIM2/` et il convient donc de l'améliorer par tous les moyens possibles et imaginables. Dans sa version actuelle, ce prototype implémente un maître disposant d'un certain nombre de tâches et plusieurs esclaves. Les esclaves envoient une requête au maître (de taille 100 ko et nécessitant un calcul de valeur 100kFlop sur le maître) quand ils estiment avoir besoin de travail à effectuer. Actuellement, le maître sert les requêtes dans l'ordre où il les reçoit. Le nombre de tâches à traiter ainsi que les communications et les calculs qu'elles impliquent sont spécifiés à part, dans un fichier de déploiement (`deployment_1.txt`).

Pour vous faire gagner un peu de temps, une seconde version du maître et des esclaves est également disponible (`deployment_2.txt`). Dans cette version, chaque esclave est capable d'évaluer ses performances (vitesse de calcul et bande passante vers le maître) et les envoie avec la requête. Le maître maintient une liste de requêtes en attente et ne commence à les satisfaire que quand il en dispose de d requêtes (d étant indiqué dans le fichier de déploiement).

La plate-forme servant à évaluer les résultats est représentée figure 1. Elle vous permettra dans dans un premier temps vos heuristiques pour différents nombres de tâches. Vous pourrez également, si vous en avez le temps, regarder le comportement de vos heuristiques pour d'autres types de tâches (avec d'autres valeurs pour les quantités de calcul et de communication), voire pour des tâches dont les caractéristiques sont uniformément distribuées dans un intervalle.

▷ **Question 1** *En négligeant le mécanisme de requêtes introduit dans la simulation précédente, formalisez le problème d'ordonnancement que l'on cherche à résoudre. Essayer de décider de la NP-complétude de ce problème dans différents cas étant plutôt difficile, passer à la question suivante.* □

▷ **Question 2** *En négligeant le mécanisme de requêtes, donner une forme générale approximative du temps nécessaire au traitement de n tâches. Donner une borne inférieure sur ce temps de traitement.* □

▷ **Question 3** *Dans la version initiale de l'ordonnanceur qui vous est fournit, le maître maintient une liste de requêtes en attente et ne commence à les satisfaire que quand il en dispose de suffisamment de requêtes.*

Essayez différentes stratégies de satisfaction des requêtes : on sert d'abord la machine la plus rapide, la machine la plus lente, la machine avec lequel le maître communique le plus rapidement, celle avec laquelle le maître communique le plus lentement, celle qui a le rapport temps de calcul sur temps de communication le plus élevé, le plus petit, celle dont le rapport entre la somme du temps (calcul + communication) sur le temps de communication est le plus élevé, le plus petit,...Pour chacune de ces heuristiques expliquez l'intuition (s'il y en a une) qui est derrière et comparez les performances en faisant varier le nombre de tâches, la taille des tâches, ...

Le script `test.sh` *et les fichiers* `deployment_generic_1.txt` *et* `deployment_generic_2.txt` *vous aideront à réaliser cette étude.* □

▷ **Question 4** *Formalisez le problème du calcul du débit optimal en régime permanent sous forme d'un programme linéaire.* □

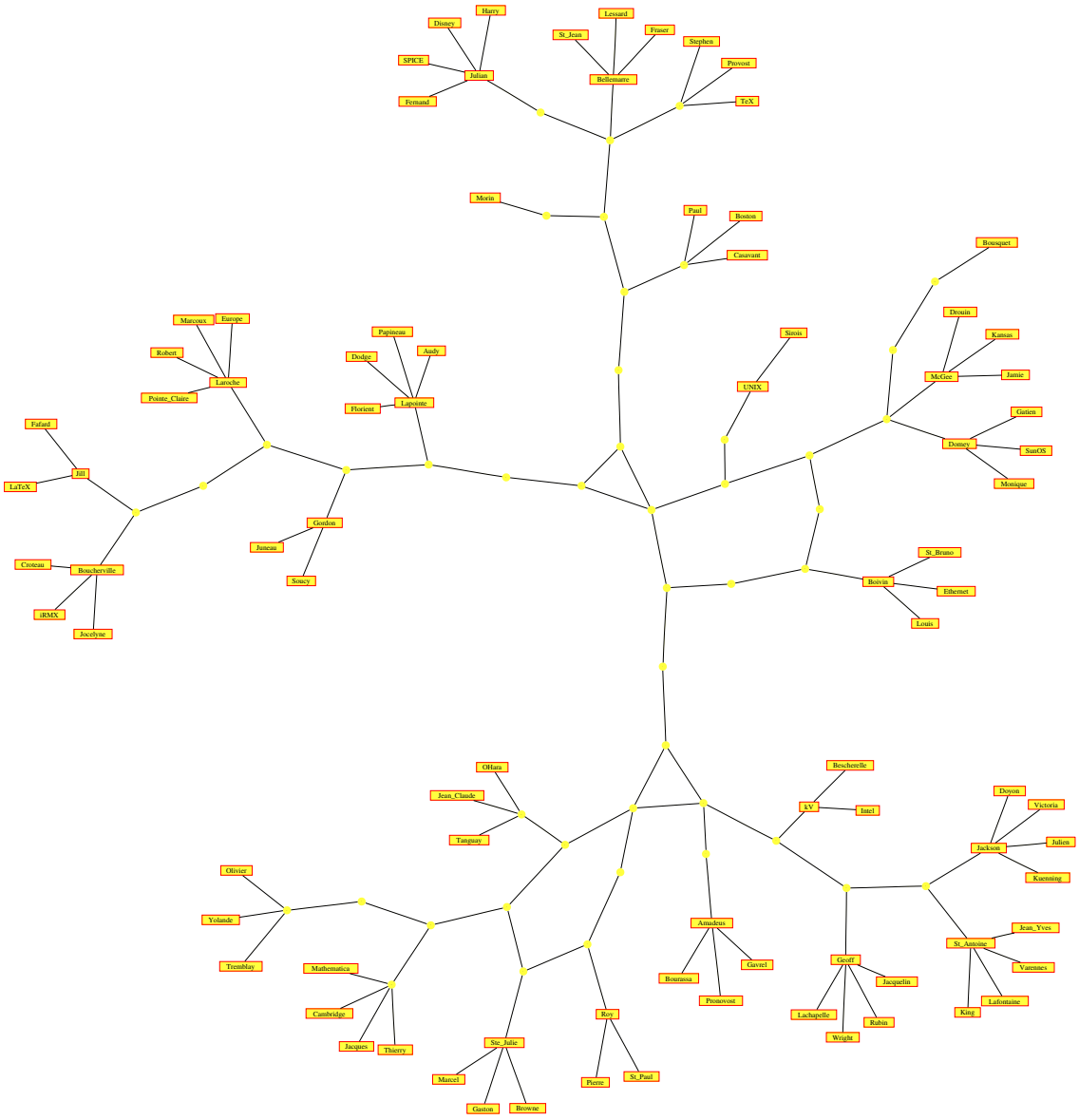

FIG. 1 – Plate-forme de test

▷ **Question 5** *Aucun recouvrement calcul/communication n'est effectué pour l'instant au niveau des esclaves. Introduisez un mécanisme de requête par anticipation permettant ce palier ce problème. Dans quelles situations ce type de mécanisme est-il utile ? Dans le cas où ce mécanisme est inefficace, quel en est la raison et comment peut-on résoudre ce problème ?* □

▷ **Question 6** *La hiérarchie de la plateforme n'est pas prise en compte et aucun parallélisme dans les communications n'est donc exploité. Comment remédier à ce problème ?* □