Ordonancement de Tâches et de Communications pour les Datacenters.

Encadrant : Stéphane Perennes, Dr2 CNRS.
Co-Encradant(s) : Nathann Cohen Cr1 CNRS, Frédéric Giroire, Cr1 CNRS.
Équipe d'accueil Projet COATI, CNRS-INRIA-i3S, Sophia Antipolis
Localisation : INRIA-Sophia Antipolis.
Durée 2-6 Mois.

Objectifs
Le sujet comporte de nombreuses questions dont certaines sont puremment théoriques et probablement difficiles, d'autres sont plus simples et pratiques. Enfin l'approche expérimentale reste ouverte en cas de blocage au niveau théorique.

Profil : Optimisation combinatoire, théorie des graphes, Recherche opérationnelle, algorithmique, complexité.

Remarque 1 *Par soucis de simplicité nous présentons en général le(s) problème(s) dans le cas unitaire et uniforme ; en pratique les modèles induisent des analogues pondérés qui sont peu ou prou équivalent du point de vue de la complexité à ces cas canoniques.*

1 Contexte : Data Center and Scheduling

Certains problèmes d'ordonnancement et de placement peu ou pas du tout étudiés apparaissent quand on modélise le problème de la réalisation d'un ensemble de tâches dans un data-center. Les tâches à réaliser sont organisées selon un graphe de tâches-machine qui représente des dépendances acycliques (DAG). L'originalité provient de la prise en compte des communications : on souhaite représenter le fait que la dépendance de la tache T_2 envers T_1 (représentée par l 'arc (T_1, T_2)) implique l'envoi d'une certaine quantité de données $c(T_1.T_2)$ depuis la machine ayant réalisé T_1 vers la machine réalisant T_2.

Le problème qui nous intéresse initialement se modélise donc comme suit :
En entrée :
- Un DAG pondéré.
  Un graphe acyclique $G = (V, E)$ de $M = |V(G)|$ tâches unitaires.
  le graphe comporte des *tâches-machine* unitaires (les sommets du graphe) et des tâches-réseau (les arcs du graphe) où l´arc (t_i, t_j) induit $c(t_i, t_j)$ communications.
- Un réseau capable d'effectuer B communications par unité de temps.
- N machines identiques chacune capable d'effectuer une tâche par unité de temps.

En sortie :
Un *ordonnancement* : cela consiste à placer et ordonner correctement [1] les tâches machines (resp. réseau) sur les machines et le réseau. La principale particularité du problème est que : **si les tâches $t_i, t_j \in V$ sont effectuées par la même machine alors la tâche réseau (t_i, t_j) (si elle existe) n'induit pas de communication (ou du moins le coût réseau de celle-ci devient nul), en effet les données sont présentes en local**.

L'objectif :
En général on souhaite minimiser le temps de complétion (*Makespan*), mais on pourra aussi étudier d'autres critères tels le temps moyen de complétion.

1. on entend par là le respect des capacités des machines, du réseau ainsi que le respect des dépendances.

2 Quelques questions associées au problème général

2.1 l'ordonnancement classique

Dans le cas où les coûts de communication sont nuls le problème est loin d'être bien résolu.

- Un algorithme glouton (datant des années 80) donne une 2-approximation très simple qui repose sur l'observation suivante : si Opt est le temps de complétion minimum on a $Opt \geq M/N$ et $Opt \geq D(G)$ où $D(G)$ est la profondeur du DAG. Or l'algorithme glouton assure la complétion du calcul en un temps $\leq D(G) + M/N$.
- Une réduction simple (à $3SAT$) montre que décider si le temps de complétion est 3 ou 4 permet de décider si une formule booléenne est satisfaisable. Ainsi un algorithme d'approximation polynomial offrant une garantie meilleure que $4/3$ est à priori exclus.
- Enfin, l'écart $4/3$ s'étend à $4T/3T$ par simple répétition de la même instance.

Ainsi bien que le problème soit extrêmement épuré et générique on ne sait pas si oui ou non un algorithme polynomial offrant une approximation meilleure que 2 existe. Cette question, où il s'agit de déterminer plus exactement le facteur d'approximation pour l'ordonnancement de tâches est probablement difficile, mais on peut se contenter de réponses partielles, étudier le cas de DAG particuliers ou encore donner des résultats expérimentaux.

Question 1 *Quel est le facteur d'approximation du problème d'ordonnancement de tâches (sur plusieurs machines).*

2.2 Le cas des communications systématiques

Dans ce cas on effectue toujours les communications (même en local). Il est facile alors d'adapter l'algorithme de 2-approximation du cas sans communication afin d'obtenir une 3-approximation.

Mais peut on faire mieux ?

Question 2 *Déterminer un algorithme d'approximation avec un facteur meilleur que* 3 *dans le cas* avec communications systématiques.

2.3 Le cas des tâches déjà placées

Une stratégie afin de résoudre le cas général pourrait être de générer des placements de tâches et d'engendrer des critères de bon placement. Mais pour ce faire il faudrait à minima comprendre le coût (temps miminal d'exécution) du graphe des tâches lorsque celles-ci sont préaffectées à des machines.

Notons ici que lorsque les tâches sont affectées les communications à effectuer sont parfaitement déterminées et bien évidemment affectées à la machine réseau. Il n'y a plus alors de raison de distinguer la machine réseau des autres, ce qui permet de simplifier et d'uniformiser l'énoncé.

Il est assez facile de montrer que le problème d ?ordonnancement associé est NP complet et que déterminer un approximation meilleure que $5/4$ est aussi un problème NP complet. Cependant nous ne disposons pas d'algorithme avec un facteur d'approximation constant.

Question 3 *étant donné un graphe acyclique de taches et une affectation des tâches à des machines déterminer une bonne approximation de l'ordonnancement optimal.*

2.4 Le problème Général.

2.4.1 Démontrer un résultat de non approximation

Nous pensons que le problème général n'est pas approximable à un facteur constant (ie n'est pas dans la classe APX). Une des raisons est qu'il semble que le problème général implique (dans certains cas particuliers) le découpage du graphe des tâches en k parties équilibrées ayant un petit bord. Plus précisément on cherche à ce que les parties soit *petites* mais sans toutefois que le nombre d'arêtes les séparant soit *trop grand*.

Notons $I = \{1, 2, \ldots k\}$; et étant donné $G = (V, E)$, pour $U, W \subset V$ nous notons $E[U, W]$ les arêtes reliant U et W.

Question 4 *Le problème suivant est il approximable : étant donné un graphe $G = V, E$ et un entier k on souhaite déterminer :*

$$Min \left\{ max_{i \in I} |V_i| + \sum_{i \neq j, i, j \in I} E[V_i, V_j], \text{ over all partitions } V_1, \ldots V_k \text{ of } V \right\}$$

Remarque 2 *Notons que ce problème naturel dans le cadre du calcul réparti usuel, on peut considérer le graphe des taches comme représentant un schéma calcul stationnaire ou les arêtes représente l'intensité des échanges de données. Si on veut optimiser le temps de calcul sur k machines avec une bande passante réseau de 1 on doit résoudre la question précédente. Cependant nous n´avons pas trouvé de travaux sur ce problème, bien que les travaux portant sur le problème de partitions équilibrées soient nombreux.*

2.4.2 Obtenir des résultats positifs

Les directions sont nombreuses, par exemple on peut penser à :
- des algorithmes offrant des garanties logarithmiques.
- des algorithmes pour des instances particulières (familles de graphes,...)
- des heuristiques non prouvées mais expérimentalement pertinentes,