Compléxités et modèles

Master 2 Informatique - UFR S.A.T

Pr. Ousmane THIARE

ousmane.thiare@ugb.edu.sn
http://www.ousmanethiare.com/

16 avril 2020

Chapitre 2 : Complexités et modèles

Chapitre 2 : Complexités et modèles

À partir de maintenant on suppose un mode de communication point-à-point modélisé par un graphe connexe et simple, a priori non orienté. On peut communiquer directement entre deux entités si elle sont connectées par une arête du graphe.

En séquentiel, on utilise surtout la complexité en temps, plus rarement celle en espace bien qu'en pratique l'espace se révèle bien plus limitant que le temps. Pour un programme qui prend un peu trop de temps, il suffit d'être patient. Un programme qui prend un peu trop d'espace doit être réécrit ou alors il faut changer le hardware. C'est donc plus contraignant en pratique.
En distribué, les notions de complexités sont plus subtiles. Les notions de nombre de messages échangés ou de volume de communication font leur apparition. On utilisera essentiellement les complexités en temps et en nombre de messages. Ces complexités dépendant, le plus souvent, du graphe sur lequel s'exécute l'algorithme. Et comme on va le voir, les définitions diffèrent aussi suivant que l'on considère un système synchrone ou asynchrone.

En distribué, les entrées (l'instance) et les sorties d'un algorithme (ou d'un programme) sont stockées sur les sommets du graphe. Aussi, par l'expression « durée d'exécution un algorithme A sur un graphe G » on entend la durée entre le démarrage du premier processeur de G exécutant A et l'arrêt du dernier processeur.

Definition (temps synchrone)

La complexité en temps d'un algorithme distribué A sur un graphe G en mode synchrone, noté Temps(A,G), est le nombre de tops horloge générés durant l'exécution de A sur G dans le pire des cas (c'est-à-dire sur toutes les entrées valides de A).

Mesures de complexité
Modèles représentatifs
Rappels sur les graphes

Remarques

- En synchrone, on parle aussi du nombre de rondes (rounds en Anglais) puisque qu'un algorithme synchrone est une succession de cycles (rondes) Send/Receive/calcul.

Remarques

- En synchrone, on parle aussi du nombre de rondes (rounds en Anglais) puisque qu'un algorithme synchrone est une succession de cycles (rondes) Send/Receive/calcul.
- Une remarque évidente mais fondamentale est qu'un algorithme de complexité en temps synchrone t implique que pendant son exécution, tout sommet ne peut interagir qu'avec des sommets situés à une distance inférieure ou égale à t.

Mesures de complexité
Modèles représentatifs
Rappels sur les graphes

Definition (temps asynchrone)

La complexité en temps d'un algorithme distribué A sur un graphe G en mode asynchrone, noté Temps(A, G), est le nombre d'unité de temps durant l'exécution de A sur G dans le pire des cas (c'est-à-dire sur toutes les entrées valides de A et tous les scénarii possibles), chaque message prenant un temps au plus unitaire pour traverser une arête.

Remarques

- La remarque précédente ne s'applique plus en mode asynchrone, puisqu'il est possible que les messages transitent entre voisin bien plus vite que le temps unitaire, et donc en temps $< t$ entre sommets à distance t.

Remarques

- Un scénario possible pour le mode asynchrone est que tous les messages sont émis en même temps et traversent chaque arête en temps 1 exactement, des messages pouvant se croiser sur une même arête. Il s'agit du scénario synchrone. Autrement dit, la complexité en temps en mode asynchrone est toujours au moins égale à celle synchrone, puisqu'on maximise le temps sur tous les scenarii possibles. En particulier, si durant l'exécution de A deux sommets à distance t interagissent, c'est que la complexité en temps (synchrone ou pas) de A doit être d'au moins t.

Remarques

- Lors d'un scénario synchrone s'exécutant sur un système asynchrone, tous les messages traversent les arêtes à la vitesse du lien le plus lent, ce qui n'est pas le scénario le plus avantageux. L'enjeu d'un algorithme asynchrone est de pouvoir fonctionner dans n'importe quel scénario.

Definition (nombre de messages)

La complexité en nombre de messages d'un algorithme distribué A sur un graphe G, noté Message(A,G), est le nombre de messages échangés durant l'exécution de A sur G dans le pire des cas (sur toutes les entrées valides de A et tous les scenarii possibles en mode asynchrone).

Remarques

- Résoudre une tâche dans un système distribué prend un certain temps et nécessite un certain volume de communication. Et la plupart du temps, il existe un compromis en ces deux valeurs. Bien sûr c'est très schématique. Il existe d'autres mesures de complexité, comme la complexité en nombre de bits, donnant le nombre total de bits échangés lors d'une exécution. C'est une mesure plus fine que le nombre de messages puisqu'à nombre de message identique, on pourrait préférer l'algorithme utilisant des messages plus courts.

Remarques

- Comme en séquentiel, on peut aussi définir toute une série de complexités, selon qu'on analyse l'algorithme sur toutes les instances ou pas : analyse dans pire des cas, dans un cas moyen, selon une distribution particulière, etc. En asynchrone on peut également limiter l'ensemble des scenarii possibles.

Parmi les nombreux modèles qui existent dans la littérature, on en distingue trois qui correspondent à des ensembles d'hypothèses différents. Chaque ensemble d'hypothèses permet d'étudier un aspect particulier du calcul distribué. Les trois aspects sont : la localité, la congestion, l'asynchronisme.
Bien sûr, d'autres aspects du calcul distribué pourrait être étudiés comme : la tolérance aux pannes, le changement de topologie, la fiabilité (liens/processeurs qui, sans être en panne, sont défaillant), la sécurité.

Mesures de complexité
Modèles représentatifs
Rappels sur les graphes

Les hypothèses suivantes sont communes aux trois modèles :

- **Pas d'erreur :** il n'y a pas de crash lien ou processeur, pas de changement de topologie. Les liens et les processeurs sont fiables. Un message envoyé fini toujours par arriver. Les messages peuvent se croiser sur une arête (pas de collision) mais il ne peuvent pas se doubler (les messages arrivent dans le même ordre d'émission).

Les hypothèses suivantes sont communes aux trois modèles :

- **Pas d'erreur :** il n'y a pas de crash lien ou processeur, pas de changement de topologie. Les liens et les processeurs sont fiables. Un message envoyé fini toujours par arriver. Les messages peuvent se croiser sur une arête (pas de collision) mais il ne peuvent pas se doubler (les messages arrivent dans le même ordre d'émission).
- **Calculs locaux gratuits :** en particulier faire un ou plusieurs Send dans le même cycle de calcul à un coût nul. Finalement, seules les communications sont prises en compte.

Les hypothèses suivantes sont communes aux trois modèles :

- **Identité unique :** les processeurs on une identité codée par un entier de O(log n) bits n étant le nombre de processeurs du graphe. Le réseau n'est pas anonyme. Ces identités font partie de l'instance du problème, l'instance étant codée par les sommets du graphe.

La troisième hypothèse (identité unique) ne se révèle pas fondamentale (et donc peut être supprimer) pour certains problèmes que nous allons aborder dans la suite. Elle est en fait nécessaire pour les problèmes consistant à casser la symétrie (symetry breaking en Anglais), comme l'élection d'un leader (que nous n'aborderons pas) ou la coloration (voir le chapitre 6). Sans cette hypothèse, ce type de problème ne sont alors pas toujours solubles.

LOCAL pour étudier la nature locale d'un problème.

- mode synchrone, tous les processeurs démarrent le calcul au même top

CONGEST pour étudier l'effet du volume des communications.

- mode synchrone, tous les processeurs démarrent le calcul au même top

ASYNC pour étudier l'effet de l'asynchronisme.

- mode asynchrone

LOCAL pour étudier la nature locale d'un problème.

- mode synchrone, tous les processeurs démarrent le calcul au même top
- message de taille illimitée

CONGEST pour étudier l'effet du volume des communications.

- mode synchrone, tous les processeurs démarrent le calcul au même top
- message de taille limitée à O(log n) bits

ASYNC pour étudier l'effet de l'asynchronisme.

- mode asynchrone

Bien sûr, on peut raffiner et multiplier les modèles (à l'infini ou presque) selon de nouvelles hypothèse, avec par exemple le modèle **ASYNC** et messages limité, **ASYNC** et messages illimités, etc.
Exemple : Dans un noeud interne d'un arbre, envoyer l'ID de ses fils vers son père.

Soit G = (V, E) un graphe. On note V (G) = V l'ensemble des sommets et E(G) = E l'ensemble des arêtes de G.
Graphe de bases : chemin, arbres, cycle, graphe complet, grille
Chemin entre deux sommets, connexité
La longueur d'un chemin est le nombre d'arêtes qui le compose, et la distance entre deux sommets u, v de G, noté dist(u, v), est la longueur d'un plus court chemin entre u et v.
Diamètre de G est la valeur diam(G) $= max_{u,v} dist(u, v)$.

Excentricité d'un sommet, centre d'un graphe (=sommet d'excentricité minimum, exemple, il peut en avoir plusieurs), rayon d'un graphe rayon(G) (=excentricité d'un centre).
Arbre : profondeur d'un sommet depth(v), profondeur d'arbre depth(T).
Voisinage dans G d'un sommet u à distance r :
$boule(u, r) = \{v \in V(G) : dist(u, v) \leq r\}$.
La somme des degrés des sommets d'un graphe vaut deux fois son nombre d'arêtes : $\sum_{u \in V(G)} deg(u) = 2m$.
Si G est connexe alors $m \geq n - 1$, et donc $m = \Omega(n)$.