ALGORITHMIQUE DISTRIBUÉE
MIF12

TRI

Isabelle GUERIN LASSOUS
perso.ens-lyon.fr/isabelle.guerin-lassous/index-AD.htm
isabelle.guerin-lassous@univ-lyon1.fr

Tri en distribué : problème 1

- Hypothèse
  - Système / Graphe avec n nœuds
  - Chaque nœud a un identifiant unique
    - ID_1 , ID_2 , ... ID_n
  - Chaque nœud stocke une valeur

- Objectif du problème 1
  - Trier l'ensemble des valeurs
  - À la fin du tri, le nœud ID_i stocke la i^e plus petite valeur

Problème 1
Algorithme simple

- Chaque nœud envoie sa valeur à un nœud spécifique (leader par ex.)
- Ce dernier trie (séquentiellement) les données
- Ce dernier envoie chaque valeur à chaque nœud concerné
  - Nœud ID_1 reçoit la $1^{ère}$ valeur (plus petite valeur)
  - Nœud ID_2 reçoit le 2^{e} valeur (2^{e} plus petite valeur)
  - Etc.

- Algorithme pas très rapide
  - Au moins aussi lent que l'algorithme séquentiel
- Il faut que le leader puisse recevoir toutes les valeurs
- Nombre de messages va dépendre de la position du leader par rapport aux autres nœuds

Problème 1
Algorithme sur une topologie en chaîne

- Algorithme basé sur des permutations de valeurs entre nœuds voisins
- Tant qu'il y a des valeurs à permuter
  - Étape 1
    - Pour i impair, le nœud ID_i échange avec le nœud ID_{i+1} si nécessaire pour le tri
  - Étape 2
    - Pour i pair, le nœud ID_i échange avec le nœud ID_{i+1} si nécessaire pour le tri

Problème 1
Algorithme sur une topologie en chaîne

- Tri pair-impair
- On peut avoir jusqu'à n étapes
  - Avec étape 1 ou étape 2 considérée comme 1 étape

Problème 1
Algorithme sur une grille

- À chaque phase, on applique un tri pair-impair sur les lignes ou sur les colonnes
- Tant que les valeurs ne sont pas triées, répéter
  - Étape 1 : tri pair-impair sur les lignes
    - Ligne impaire : tri croissant (de la gauche vers la droite)
    - Ligne paire : tri décroissant (de la gauche vers la droite)
  - Étape 2 : tri pair-impair sur les colonnes
    - Tri croissant du haut vers le bas

Problème 1
Algorithme sur une grille

- Shear sort
- Ceiling($\log_2 n$) + 1 étapes (n nombre de nœuds)
  - Étape 1 ou étape 2 = 1 étape

Problème 1
Topologie quelconque

- Algorithmes précédents utilisent des topologies spécifiques
- Utiliser virtuellement ces topologies
- Renuméroter les nœuds pour s'approcher au mieux de ces topologies

Tri en distribué : problème 2

- Hypothèse
  - Système / Graphe avec n nœuds
  - Chaque nœud a un identifiant unique
    - ID_1 , ID_2 , ... ID_n
  - Chaque nœud stocke un ensemble de valeurs
    - Si L valeurs au total, L/n valeurs par nœud

- Objectif du problème 2
  - Trier l'ensemble des valeurs
  - À la fin du tri, le nœud ID_i stocke les i^e L/n plus petites valeurs

Problème 2
Algorithme sur une topologie en chaîne

- Algorithme de tri pair-impair
- Chaque nœud trie ses valeurs localement
- Tant qu'il y a des valeurs à permuter
  - Étape 1
    - Pour i impair, le nœud ID_i échange avec le nœud ID_{i+1} leurs valeurs
    - Chaque nœud effectue une fusion des 2 listes de valeurs déjà triées
    - Le nœud ID_i garde les L/n plus petites valeurs et le nœud ID_{i+1} garde les L/n plus grandes
  - Étape 2
    - Pour i pair, le nœud ID_i échange avec le nœud ID_{i+1} leurs valeurs
    - Chaque nœud effectue une fusion des 2 listes de valeurs déjà triées
    - Le nœud ID_i garde les L/n plus petites valeurs et le nœud ID_{i+1} garde les L/n plus grandes

Problème 2
Algorithme bucket sort distribué

- Valeurs à trier réparties dans [a ; b]
  - b – a = K
  - Hypothèse : K divisible par n
- Nœud ID_i doit récupérer les valeurs dans [a+(i-1)K/n ; a+iK/n[
- Algorithme
  - Chaque nœud prépare les valeurs à envoyer aux autres nœuds
    - Les valeurs dans [a ; a+K/n[sont le pour le nœud ID_1
    - Les valeurs dans [a+K/n ; a+2K/n[sont pour le nœud ID_2
    - Etc.
  - Chaque nœud trie les valeurs reçues
- Commentaires
  - Selon la répartition des valeurs dans [a ; b], il est possible qu'un nœud reçoive plus de L/n valeurs
  - Il faudrait une répartition uniforme dans [a ; b]