CentraleSupélec

Big Data : Informatique pour les données et calculs massifs

4 – Schémas élémentaires de parallélisation

Stéphane Vialle

CentraleSupélec université PARIS-SACLAY | École doctorale Sciences et technologies de l'information et de la communication (STIC) | RISEGrid | Grand Est

Stephane.Vialle@centralesupelec.fr
http://www.metz.supelec.fr/~vialle

1 - Outils de synchronisation

- *Verrous*
- *Sémaphores*
- *Variables conditionnelles*

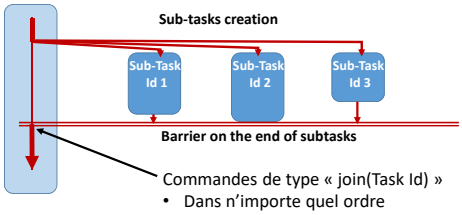
} Autre cours

- Barrières sur fin de tâches
- Barrières génériques

1 - Outils de synchronisation

Barrières sur fin de tâches

- Chaque tâche arrivant sur une barrière se bloque
- La dernière du groupe à se bloquer débloque toutes les tâches
- Concept de « groupe de tâches » associé à une barrière



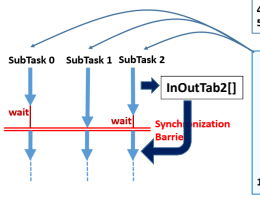
Commandes de type « join(Task Id) »

- Dans n'importe quel ordre
- Ou bien un « join([task Id]) »

1 - Outils de synchronisation

Barrières génériques

Ex : Calculs itératifs & Synchronisation à chaque itération




```
1 // Main task code
2 Data_t InTab1[N];
3 Data_t InOutTab2[N];
4 Data_t OutTab3[N];
5 .... // - Create and launch SubTasks
```

```
1 // SubTask code
2 int Me = .... // Ranking each task
3 ....
4 // - read InTab1, write into InOutTab2
5 f(InTab1,InOutTab2,N/3*Me, N/3*(Me+1))
6 // - Wait all tasks have finished to write into InOutTab2
7 Barrier(TaskGroupId)
8 // - read InOutTab2, write into OutTab3
9 g(InOutTab2,OutTab3,N/3*Me, N/3*(Me+1))
10 ....
```

Barrière : pour être sur que le tableau InOutTab2[ ] a été entièrement généré avant d'être exploité

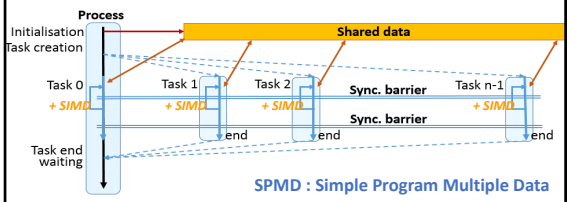
1 - Outils de synchronisation

Mise en œuvre de barrières génériques

- Assez simple et rapide en mémoire partagée (dans un PC)
  → sémaphores, PAS d'attente active…
- Plus complexe et plus couteux en mémoire distribué (clusters)
  → échanges de messages entre PC

Traverser une barrière de synchronisation à toujours un coût (même si les tâches terminent en même temps)
→ En mettre le moins possible !

2 – Schéma SPMD (HPC)

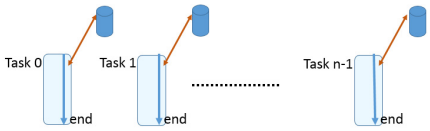


SPMD : Simple Program Multiple Data

- 1 seul programme répété dans toutes les tâches
- Toutes les tâches s'exécutent en parallèle mais sur des données différentes
- Synchronisation seulement sur des barrières
- Divergences possibles entre les tâches, avec respect des barrières

Rmq : + code SIMD dans chaque tâche (HPC)
(SPMD + Single Instruction Multiple Data …)

2 – Schéma SPMD (HPC)

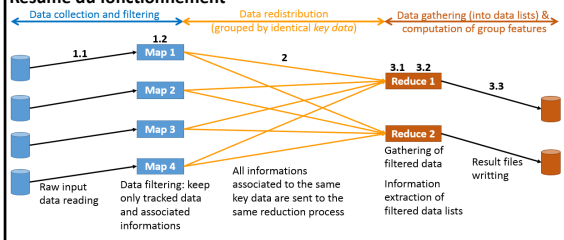


SPMD (Simple Program Multiple Data) Embarrassingly Parallel

- 1 seul programme répété dans toutes les tâches
- Toutes les tâches s'exécutent en parallèle mais sur des données différentes

3 – Schéma Map-Reduce (Big Data)

Résumé du fonctionnement

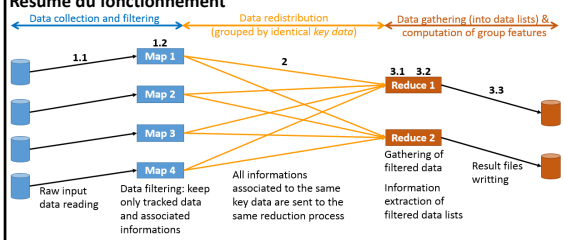


Map → Shuffle & Sort → Reduce

À base de paires (clé – valeur(s))

3 – Schéma Map-Reduce (Big Data)

Résumé du fonctionnement



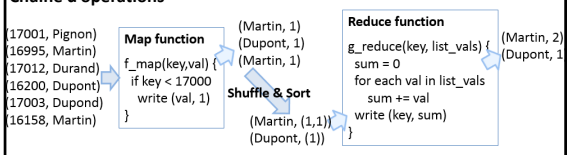
La collecte et le filtrage des données

La redistribution des données en ensembles cohérents

Le regroupement et le calcul de caracté-ristiques de groupes

3 – Schéma Map-Reduce (Big Data)

Chaîne d'opérations

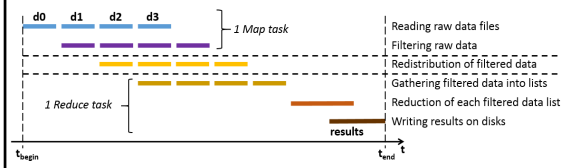


Suite de transformations de paires « clé-valeur ».

La fonction *Reduce* d'*Hadoop* impose que toutes les valeurs associées à une même clé soient stockées dans la mémoire du nœud.

3 – Schéma Map-Reduce (Big Data)

Pipelining



- Pour augmenter le parallélisme, et réduire le temps d'exécution global
- Pour tenter de masquer les temps d'IO (masquer les lectures et écritures de fichiers temporaires)

4 – Impacts de la volumétrie

- **Amener les traitements aux données**
  plus rapides que l'inverse pour de gros volumes sur des sites distants
- **Organiser des traitements par blocs**
  traiter des problèmes qui ne tiennent pas en RAM
- **Paralléliser efficacement les traitements**
  tâches *Map* indépendantes, tâches *Reduce* indépendantes, communications optimisées et recouvertes avec les calculs, tunning possible en *Hadoop*
- **Pipeliner les traitements et les communications**
  masquer les coûts des communications, réduire les temps d'exec.

\+ **Rendre simples les développements applicatifs**
métier de Data Scientist ≠ Expert d'informatique distribuée

→ **Map-Reduce est souvent un bon compromis**

Schémas élémentaires de parallélisation