MapReduce

Malo Jaffré, Pablo Rauzy

ENS

16 avril 2010

Qu'est ce que c'est ?

Conceptuellement

Données

- MapReduce est un framework de calcul distribué sur de grosses quantités de données.

Parallélisme

- MapReduce permet de répartir la charge sur un grand nombre de serveurs.
- Distribution "haut niveau" avec une abstraction quasi-totale du la couche matérielle : ajouter des machines suffit à augmenter la capacité de calcul de manière "plug-and-play" (scalable-friendly).

Implémentations

- la librairie MapReduce de Google existe dans plusieurs langages de programmation dont C++, C#, Erlang, Java, Python, Ruby...
- D'autre librairie implémente le concept : par exemple Hadoop de la fondation Apache et Elastic de chez Amazon.

Abstraction et encapsulation

- MapReduce gère entièrement le cluster et la répartition de la charge.
- Cela permet de faire du calcul distribué sans aucune connaissance dans le domaine (Cloud).

Qu'est ce que c'est ?

Utilisateurs

- "The Yahoo ! Search Webmap is a Hadoop application that runs on a more than 10,000 core Linux cluster and produces data that is now used in every Yahoo ! Web search query."
- Google : "the size of one phase of the computation [of the index] dropped from approximately 3800 line of C++ code to approximately 700 lines when expressed using MapReduce."
- "Facebook has multiple Hadoop clusters deployed now - with the biggest having about 2500 cpu cores and 1 PetaByte of disk space. We are loading over 250 gigabytes of compressed data (over 2 terabytes uncompressed) into the Hadoop file system every day and have hundreds of jobs running each day against these data sets."
- Hadoop est aussi utilisé par Twitter, Amazon, Rackspace, LinkedIn, IBM, Veoh, Last.fm (en Bash ! !), Microsoft...

map/reduce vs MapReduce

map/reduce

On travaille ici essentiellement avec des listes :

- `map` est une fonction qui applique une autre fonction à tout les éléments d'une liste (renvoie une liste).
- `reduce` applique récursivement une fonction aux éléments d'une liste par paires en utilisant une "graine" la première fois puis le résultat de l'application précédente (renvoie un élément).

MapReduce

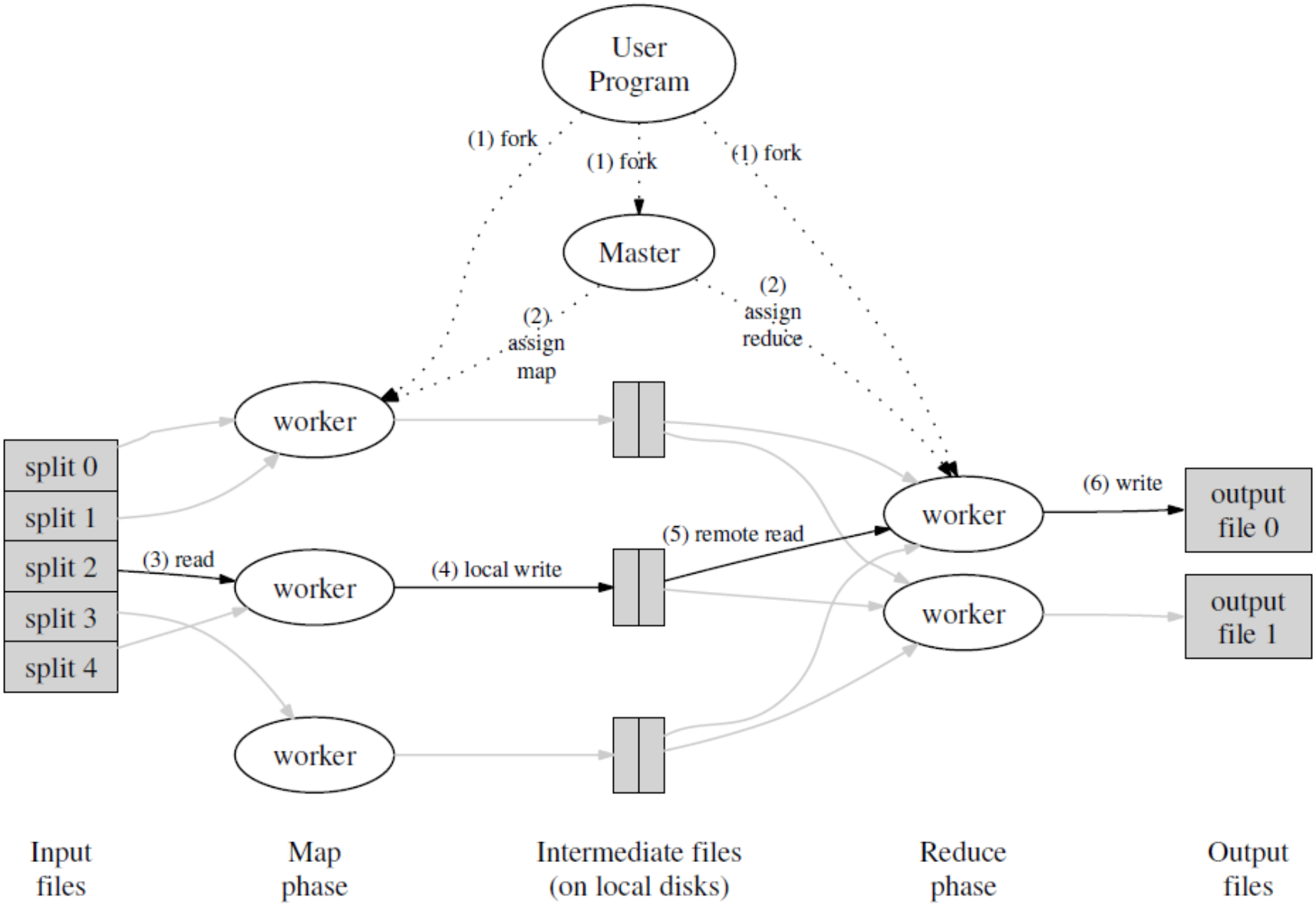
- Map : La fonction Map prend en argument un enregistrement et calcule une liste de couples clef/valeur intermédiaires.
- Reduce : La fonction Reduce prend en argument une clef et la liste des valeurs intermédiaire générées par les différentes instances de la fonction Map pour cette clef et fait quelque chose avec...

Code sample

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

MapReduce expliqué



Tolérance aux erreurs et performances

- Si l'erreur intervient lors d'une opération Map ou Reduce, la tâche et simplement réattribuée à un autre *worker* disponible.
- Une tâche peut être relancée même après avoir finie avec succès si le *mapper* décède avant que le *reducer* ne récupère les valeurs intermédiaires.
- Si le *master* plante, les emplacements des valeurs intermédiaires sont perdues et il faut donc recommencer entièrement la tâche.
- Quand certains enregistrement font systématiquement échouer le programme, ils sont vite ignorés afin d'éviter qu'ils soit infiniment relancés.
- Une politique de *backup* permet de dupliquer les tâches qui prennent le plus de temps quand la plupart des *workers* ont déjà fini leur travail.
- Prise en compte de l'emplacement géographique des données (plus proche = plus rapide, moins cher)

En pratique : grep

- La fonction Map renvoie un enregistrement s'il match le motif.
- La fonction Reduce est simplement la fonction identité, son rôle est juste d'écrire les valeurs intermédiaires sur la sortie.

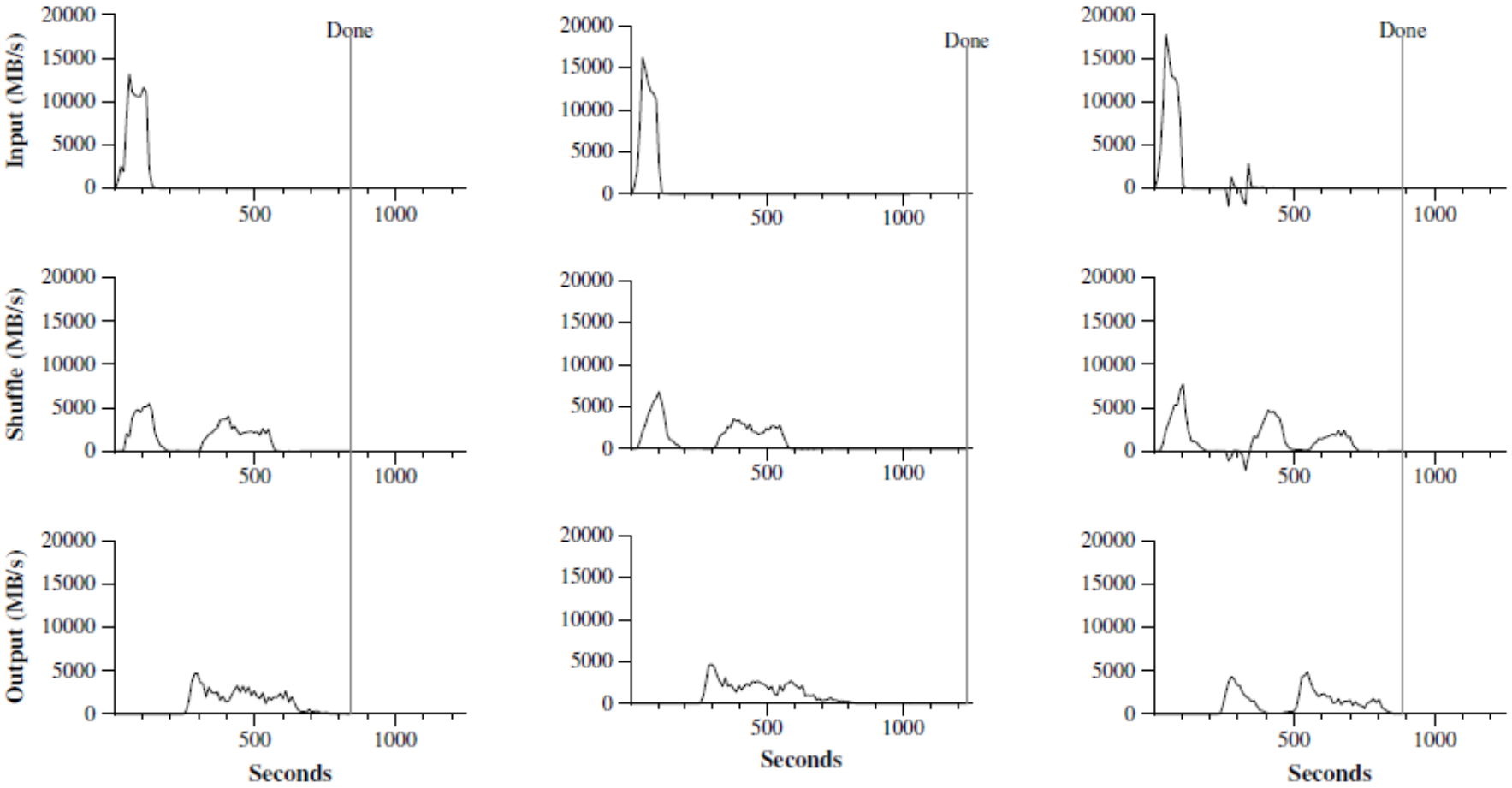
Caractéristiques

- 1To de données.
- 1800 serveurs.
- Les données sont découpées en 15000 morceaux d'environ 64Mo.
- Un seul serveur effectue la reduction (afin d'avoir les résultats dans un seul fichier).

Performances

- Démarrage relativement lent dû au temps de propagation du programme ($\pm$1mn).
- Les Maps sont tous finis au bout d'environ 80s.
- L'opération se termine en 150s.
- On atteint un pic de lecture de 30Go/s.

Performances : graphique



(a) Normal execution

(b) No backup tasks

(c) 200 tasks killed

L'utilisation dans le domaine des bases de données

De plus en plus on a besoin de manipuler des quantités de données de plus en plus grosses (web).

On a vu en cours comment distribuer les données, il reste à distribuer les calculs. C'est là que MapReduce intervient.

Enfin bref, comme tout le reste, ça va dans le cloud...

L'implémentation de quelques unes des opérations de l'algèbre PSJRU

Sélection

- Map retourne l'enregistrement s'il correspond aux critères de sélection.
- Reduce est l'identité.

Jointure

- Map renvoie une paire intermédiaire composée du champ de jointure en clef et l'enregistrement en valeur.
- Reduce reçoit pour chaque clef de jointure la liste des enregistrements correspondants et se charge de faire le produit cartésien de ces enregistrements.

Valeur ajoutée

Nous avons vu en cours qu'il est possible de traduire les requêtes SQL ne contenant pas de négations dans l'algèbre PSJRU.

Et comme l'algèbre PSJRU peut facilement être implémentée en MapReduce, il est donc assez aisé de faire évoluer les SGBD actuel vers le cloud grâce à MapReduce.

Oracle a commencé à implémenter des fonctions MapReduce dans certains de ses SGBD.

#NoSQL

Les données sont stockées sur des systèmes de fichiers distribués sous forme de paires clef/valeur.

HadoopDB

- Une base de données implémenté entièrement en MapReduce avec HDFS (équivalent de GFS).
- Permet de très bonnes performances sur des gros jeux de données.
- Permet une bonne fiabilité par une redondance des données.

En contrepartie...

- Ne permet pas de faire des requêtes aussi tordues qu'une bdd relationnelle classique.
- L'écriture peut poser des problèmes (mise à jour et leur propagation)
- Certains reprochent aussi un manque de formalisme.

Ce n'est en fait pas adapté au même type de problèmes.

Conclusion

Cette technologie permet entre autres la mise à jour de l'index de Google en temps réel alors qu'il n'y a pas si longtemps, la *Google Dance* n'avait lieu environ qu'une fois par mois, autant dire qu'il était impossible d'indéxer les tweets...

À quand la fusion des différentes bases de données personnelles (téléphone, laptop, ordinateur de la maison, du travail, réseaux sociaux...) dans le cloud ?