# MapReduce & YARN

**Fiche de Révision Examen**

*Pr. Btihal El Ghali*

## 1 Introduction à MapReduce

### 1.1 Exemple Introductif : Calcul du Chiffre d'Affaires

**Entreprise Bamboo - 2022**

**Objectif :** Calculer les chiffres d'affaires mensuels
**Données :** 1000 lignes de ventes

- 1 personne seule : 1 heure
- 20 personnes en parallèle : 3 minutes

**Principe :** Diviser pour régner - Traitement parallèle

### 1.2 Définition de MapReduce

**Qu'est-ce que MapReduce ?**

Framework de programmation Hadoop pour traiter de grandes quantités de données en parallèle de manière fiable et tolérante aux pannes.

**Fonctionnement :**

1. Les blocs de données sont lus et traités par les tâches **Map** s'exécutant sur les DataNodes où les blocs sont stockés
2. Les sorties des tâches Map sont **shufflées** (mélangées)
3. Les données shufflées sont envoyées aux tâches **Reducer**

## 2 Composants de MapReduce

### 2.1 Le Mappeur (Mapper)

**Définition**

Programme relativement petit avec une tâche fonctionnelle simple

**Responsabilités :**

- Lire une partie des données d'entrée (un bloc)
- Interpréter les données
- Filtrer, combiner ou transformer si nécessaire
- Produire un flux de paires `<Key, Value>`

**Nombre de Mappers**

$$\text{Nombre de Maps} = \lceil \frac{\text{Taille du fichier}}{\text{Taille d'un bloc}} \rceil$$

**Exemple :** Fichier 1 Go, bloc 128 Mo $\rightarrow$ 8 mappers

### 2.2 Le Réducteur (Reducer)

**Définition**

Petit programme chargé de réduire les valeurs associées à une clé

**Fonctionnement :**

1. Données émises par mappeurs regroupées localement par `<clé>`
2. Pour chaque clé unique : un nœud (réducteur) est choisi
3. Il traite toutes les valeurs de tous les mappeurs pour cette clé
4. Il les réduit à une seule valeur

**Sortie des Reducers**

**Fichiers générés :** 1 fichier par reducer sur HDFS
**Nombre de reducers :**

- Minimum : 1
- Maximum : 4 (généralement)

### 2.3 Le Combinateur (Combiner)

**Définition**

Aussi appelé "mini-réducteur" - S'exécute sur le même nœud que le mappeur

**Rôle :**

- Traite les données de sortie du mappeur **AVANT** de les transmettre au reducer
- **Objectif :** Minimiser les données transférées entre mappeurs et réducteurs
- **Avantage :** Réduire l'encombrement du trafic réseau

**Important**

Le Combiner n'est PAS obligatoire, mais améliore les performances réseau

## 3 Exemple Pratique : Température Maximale

**Calcul de la température maximale mensuelle**

**Objectif :** Calculer la température max de chaque mois
**Données d'entrée :**

```
Date        Température
01012010    26
18012010    27
30012010    29
02022010    18
...
```

### 3.1 Sans Combiner

**Phase Map (Node 1) :**

```
01012010 26  →  01 26
```

```
18012010 27  →  01 27
30012010 29  →  01 29
02022010 18  →  02 18
```

**Phase Shuffle :** Regroupement par clé (mois)

**Phase Reduce :**

```
Reducer 1: 01 → [26, 27, 29, 25, 31, 27] → 01 31
Reducer 2: 02 → [18, 17, 19, 15, 13, 14] → 02 19
```

### 3.2 Avec Combiner

**Phase Map + Combiner (Node 1) :**

```
Mapping:          Combining:
01 26             01 29  (max local)
01 27      →      02 19  (max local)
01 29
02 18
02 17
02 19
```

**Avantage :** Moins de données transférées sur le réseau !

## 4 Architecture MapReduce (Hadoop 1.x)

### 4.1 Concepts Clés

**Job vs Tâche**

**Job :**

- Composé de nombreuses tâches
- Représente ce que le client veut exécuter
- Un seul job par application

**Tâche :**

- Un programme Map OU Reduce
- Représente une tâche individuelle

### 4.2 Composants Hadoop 1.x

| Type | Composant | Rôle |
|---|---|---|
| Master | JobTracker | Gestion des ressources et traitement de données |
| Slave | TaskTracker | Exécution des tâches assignées |

### 4.3 JobTracker (Master)

**Responsabilités du JobTracker**

**Gestion des Ressources :**

— Maintenir liste des nœuds actifs

— Maintenir liste des slots map/reduce disponibles et occupés

— Affecter les slots disponibles aux tâches

**Traitement de Données :**

— Ordonner aux TaskTrackers de démarrer les tâches

— Surveiller l'exécution des tâches

— Redémarrer les tâches ayant échoué

### 4.4 TaskTracker (Slave)

**Responsabilités :**

— Exécuter les tâches assignées par JobTracker

— Signaler périodiquement le progrès au JobTracker

## 5 Limitations de MapReduce v1

### 5.1 Limitation 1 : Unicité du JobTracker

**Single Point Of Failure (SPOF)**

**Problème :**

— JobTracker est unique

— S'il tombe en panne : TOUS les travaux sont perdus

— Jobs sont à SPOF

**Bottleneck chez Yahoo :**

— 5,000 nœuds avec 1 seule machine JobTracker

— 40,000 tâches exécutées en parallèle

### 5.2 Limitation 2 : Slots Non Interchangeables

**Problème de Flexibilité**

**Situation :**

— Nombre fixe de slots Map et slots Reduce (défini par admin)

— Slots NON interchangeables

— Si tous slots Map pris : on ne peut pas utiliser slots Reduce libres

— Vice versa également

**Conséquence :** Mauvaise utilisation du cluster

### 5.3 Limitation 3 : Uniquement MapReduce

**Applications Non Supportées**

Hadoop 1.0 supporte UNIQUEMENT les jobs programmés en MapReduce
**Applications non supportées :**

- **Giraph** : Traitement de graphes (Facebook)
- **Spark** : Traitement en mémoire vive
- **Storm & Flink** : Ingestion de flux temps réel

## 6 YARN - La Solution (Hadoop 2.x)

### 6.1 Origine

**Yahoo ! - 2010**

Ingénieurs de Yahoo ! ont développé une architecture complètement nouvelle de Hadoop qui corrige toutes les limitations de MapReduce v1

### 6.2 YARN - Yet Another Resource Negotiator

**Principe de YARN**

Au lieu d'avoir un seul JobTracker, YARN introduit 3 composants :

1. **Resource Manager (RM)** : Gestionnaire de cluster
   - Suit les nœuds actifs
   - Suit les ressources disponibles
   - Affecte les ressources aux tâches
2. **Application Master (AM)** : JobTracker court-vécu
   - Un AM par Job soumis
   - Contrôle l'exécution des tâches de ce job uniquement
3. **Node Manager (NM)** : Remplace TaskTracker
   - Démarre les Application Masters
   - Gère les conteneurs

### 6.3 Avantages de YARN

- **Coordination distribuée :** Cycle de vie des jobs réparti sur toutes les machines
- **Plus de tâches en parallèle :** Grâce aux AM distribués
- **Évolutivité accrue :** Plus de bottleneck sur un seul JobTracker
- **Pas de SPOF :** La panne d'un AM n'affecte qu'un seul job

## 7 Composants de YARN

## 7.1 Resource Manager (RM)

**Master Daemon**

S'exécute en tant que master daemon sur une machine dédiée

**Responsabilités :**

— Connaît les nœuds actifs
— Connaît les ressources disponibles sur le cluster
— Affecte les ressources aux applications

**Équivalent Hadoop 1.x**

JobTracker (partie gestion des ressources)

## 7.2 Application Master (AM)

**Short-Living JobTracker**

Démarré lorsque l'utilisateur soumet une application - Un AM par Job

**Responsabilités :**

— Coordonne l'exécution de toutes les tâches de l'application
— Surveille les tâches
— Redémarre les tâches ayant échoué
— Demande les ressources au Resource Manager

**Équivalent Hadoop 1.x**

JobTracker (partie traitement de données)

## 7.3 Node Manager (NM)

**Remplace TaskTracker**

Gère les ressources et processus sur chaque nœud esclave

**Responsabilités :**

— Fournit des ressources de calcul sous forme de **conteneurs**
— Gère les processus à l'intérieur des conteneurs
— Lance les Application Masters
— Plus de nombre fixe de slots : **conteneurs dynamiques**

**Équivalent Hadoop 1.x**

TaskTracker

## 7.4 Container

**Unité de Ressources Dynamique**

Remplace les slots fixes de Hadoop 1.x

**Caractéristiques :**

— Peut exécuter différents types de tâches
— Créés dynamiquement (pas de nombre fixe)
— Taille variable selon ressources : mémoire, CPU, disque, E/S réseau
— **Interchangeables** : Plus de distinction Map/Reduce

**Équivalent Hadoop 1.x**

Slots (Map slots + Reduce slots fixes)

## 8 Processus de Soumission d'Application YARN

1. **Client** soumet application au Resource Manager
2. **Resource Manager** alloue un conteneur pour l'Application Master
3. **Node Manager** lance l'Application Master
4. **Application Master** s'enregistre auprès du Resource Manager
5. **Application Master** négocie les ressources (conteneurs)
6. **Node Managers** lancent les conteneurs pour les tâches
7. **Tâches** s'exécutent et reportent leur progrès à l'AM
8. **Application Master** informe le RM de la complétion
9. **Application Master** se termine

## 9 MapReduce v2 (MRv2)

**Qu'est-ce que MRv2 ?**

Dans YARN, MapReduce est dégradé en rôle d'application distribuée et s'appelle MRv2

**MRv2 :**

— Ré-implémentation du moteur classique MapReduce
— S'exécute au-dessus de YARN
— N'est plus le seul framework possible
— Devient une application parmi d'autres

## 10 Analogie : Start-up → Grande Entreprise

### 10.1 MapReduce (Hadoop 1.x) = Start-up

**Petite Entreprise**

**Structure :**

— **1 seul chef de projet** (JobTracker)
  — Gère tous les projets
  — Gère les ressources humaines
  — Gère le staffing
— **Développeurs** (TaskTrackers)
  — Réalisent les projets
  — Sous la direction du chef de projet unique

**Problème :** Le chef de projet devient un goulot d'étranglement !

### 10.2 YARN (Hadoop 2.x) = Grande Entreprise

**Entreprise Évoluée**

**Nouvelle Structure :**

- **Gestionnaire de ressources** (Resource Manager)
  - Visibilité globale sur la société
  - Alloue les développeurs aux projets
- **Plusieurs chefs de projet** (Application Masters)
  - Un chef par projet
  - Demande des ressources au gestionnaire
- **Chefs d'équipe** (Node Managers)
  - Responsables d'un petit nombre de développeurs
- **Développeurs** (Containers)
  - Flexibles, peuvent travailler sur différents projets

**Avantage :** Distribution de la charge et spécialisation !

## 11 Comparaison Hadoop 1.x vs 2.x

### 11.1 Améliorations Mesurées (Yahoo)

**Résultats après Hadoop 2.0**

- $\times$**2** nombre de tâches exécutées par jour
- $\times$**2** utilisation du CPU
- $\times$**2.5** nombre de tâches de tous les jobs

### 11.2 Tableau Comparatif

| **Critère** | **Hadoop 1.x** | **Hadoop 2.x (YARN)** |
|---|---|---|
| **Gestion Ressources** | JobTracker unique | Resource Manager |
| **Gestion Jobs** | JobTracker unique | Application Master (1 par job) |
| **Exécution Tâches** | TaskTracker | Node Manager |
| **Unité Ressource** | Slots fixes (Map/Reduce) | Containers dynamiques |
| **SPOF** | Oui (JobTracker) | Non (RM redondant possible) |
| **Scalabilité** | Limitée (1 JobTracker) | Excellente (AM distribués) |
| **Flexibilité Slots** | Non interchangeables | Containers interchangeables |
| **Applications** | MapReduce uniquement | MapReduce, Spark, Giraph, Storm, Flink, etc. |
| **Utilisation Cluster** | Moyenne | Optimale |

### 11.3 Équivalences des Composants

| **Rôle** | Hadoop 1.x | Hadoop 2.x |
|---|---|---|
| Gestion ressources | JobTracker | Resource Manager |
| Gestion job | JobTracker | Application Master |
| Exécution tâches | TaskTracker | Node Manager |
| Unité de calcul | Slot (Map/Reduce) | Container |

# 12 Architecture Complète

## 12.1 Hadoop 1.x - Calcul Distribué

**HDFS + MapReduce v1**

**Couche Stockage :**

— NameNode (Master)
— Secondary NameNode
— DataNodes (Slaves)

**Couche Traitement :**

— JobTracker (Master)
— TaskTrackers (Slaves)

## 12.2 Hadoop 2.x - Calcul Distribué

**HDFS + YARN + Applications**

**Couche Stockage :**

— Active NameNode (Master)
— Standby NameNode (HA)
— DataNodes (Slaves)

**Couche Gestion Ressources :**

— Resource Manager (Master)
— Node Managers (Slaves)

**Couche Applications :**

— Application Masters (1 par job)
— Containers (dynamiques)

**Applications Supportées :**

— MapReduce v2
— Apache Spark
— Apache Giraph
— Apache Storm
— Apache Flink
— Etc.

# 13 Points Clés pour l'Examen

## 13.1 Formules et Calculs

**Formules Essentielles**

**Nombre de Mappers :**

$$\text{Nb Mappers} = \lceil \frac{\text{Taille fichier}}{\text{Taille bloc HDFS}} \rceil$$

**Exemple :** 1 Go / 128 Mo = 8 mappers

**Nombre de Reducers :**

- Minimum : 1
- Maximum : 4 (généralement)
- Déterminé par le nombre de clés uniques

**Fichiers de sortie :**

$$\text{Nb fichiers sortie} = \text{Nb reducers}$$

## 13.2 Concepts à Maîtriser

**MapReduce**

1. Mapper : Produit paires <Key, Value>
2. Shuffle : Regroupe par clé
3. Combiner : Mini-reducer local (optionnel)
4. Reducer : Agrège valeurs par clé

**Hadoop 1.x**

- **JobTracker** : Master unique (SPOF)
- **TaskTracker** : Slaves
- **Slots** : Fixes, non interchangeables
- **Applications** : MapReduce uniquement

**Hadoop 2.x (YARN)**

- **Resource Manager** : Gestion ressources globale
- **Application Master** : 1 par job (short-lived)
- **Node Manager** : Gestion conteneurs
- **Containers** : Dynamiques, interchangeables
- **Applications** : Multiples frameworks

## 13.3 Limitations Hadoop 1.x

**3 Limitations Majeures**

1. **SPOF JobTracker** : Panne = perte de tous les jobs
2. **Slots fixes** : Map/Reduce non interchangeables
3. **MapReduce only** : Pas d'autres frameworks

## 13.4 Avantages YARN

**Améliorations YARN**

- Pas de SPOF (AM distribués)
- Containers dynamiques et flexibles
- Support multi-frameworks
- Meilleure utilisation cluster
- Scalabilité accrue

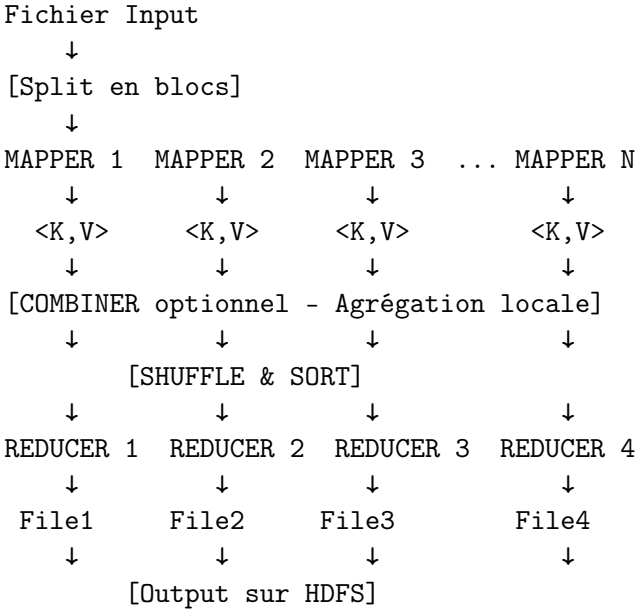
## 13.5 Questions Types

**Questions Fréquentes**

**Q1 :** Combien de mappers pour un fichier de 500 Mo avec blocs de 128 Mo ?
**R1 :** $\lceil 500/128 \rceil = 4$ mappers
**Q2 :** Quelle est la différence entre JobTracker et Application Master ?
**R2 :** JobTracker est unique et gère tous les jobs (SPOF). Application Master : un par job, distribué.
**Q3 :** Pourquoi utiliser un Combiner ?
**R3 :** Réduire le trafic réseau en pré-agrégeant les données localement avant le shuffle.
**Q4 :** Quelle amélioration apporte YARN pour les slots ?
**R4 :** Containers dynamiques et interchangeables vs slots fixes Map/Reduce.
**Q5 :** Combien de fichiers génère un job avec 3 reducers ?
**R5 :** 3 fichiers (1 par reducer)

## 13.6 Schéma Mental

**Pipeline MapReduce Complet**




*Bonne révision et bon courage pour l'examen !*