Projet - Concevoir un pipeline Hadoop résilient et dimensionné

- **Module :** Big Data et traitement massif
- **Durée :** 6 heures
- **Organisation :** équipes de 2 personnes
- **Environnement :** Hadoop sur Docker Compose (cluster fourni, identique à celui du TD2)
- **Technologies autorisées :** HDFS, YARN, MapReduce Streaming et/ou Hive
- **Technologie exclue pour ce projet :** Spark, traité dans l'autre partie du module
- **AWS :** interdit pour ce projet (aucun risque de facturation)

1. Contexte : le lendemain de la grande promotion

Vous rejoignez l'équipe data de **MarchéPlus**, une enseigne de commerce en ligne. Après une grande campagne promotionnelle, la direction veut recevoir chaque matin un rapport fiable avant l'ouverture des équipes commerciales.

Aujourd'hui, les données sont traitées par un script exécuté sur une machine unique. Le rapport est parfois en retard, les relances produisent des résultats incohérents et personne ne sait combien de temps les données doivent rester disponibles.

Votre équipe doit proposer et démontrer une solution Hadoop pour ce traitement. Il ne s'agit pas seulement de faire fonctionner un job : vous devez **faire des choix d'architecture, les justifier et expliciter leurs limites**.

Besoin métier

Chaque nuit, trois flux sont déposés :

- les commandes ;
- les consultations de produits ;
- les annulations et retours.

Le traitement doit produire, pour chaque journée :

- le chiffre d'affaires net par catégorie ;
- le nombre de commandes par catégorie ;
- les cinq produits les plus consultés par catégorie ;
- une liste des anomalies de données à examiner.

Les équipes métier veulent consulter les résultats **avant 6 h**. Une journée de données peut être corrigée ou complétée le lendemain : le pipeline doit donc permettre une reprise ou un recalcul sans publier silencieusement un résultat incohérent.

Contraintes communiquées par l'entreprise

- En moyenne, **120 Go de données compressées** arrivent chaque nuit.
- Pendant les promotions, le volume peut atteindre **1,5 fois le volume moyen**.
- Les fichiers arrivent entre minuit et 3 h ; le traitement doit se terminer avant 5 h 30, afin de laisser une marge avant la publication.

- Les données brutes doivent être conservées **90 jours**.
- Les données intermédiaires de calcul ne sont nécessaires que pendant le traitement et les reprises.
- Les résultats agrégés doivent rester disponibles pendant **deux ans**.
- Les identifiants clients ne doivent pas apparaître dans les rapports métier.
- L'équipe dispose d'un budget limité et ne veut pas laisser un cluster actif en permanence : **le cluster est considéré comme jetable** (créé pour le traitement, détruit ensuite).
- Une interruption d'un nœud ne doit pas entraîner automatiquement la perte définitive des données ou du résultat.

À vous d'identifier les ambiguïtés.

Le cahier des charges ne précise pas, par exemple, si les fichiers tardifs modifient les rapports déjà publiés ni quelle règle métier appliquer à une commande annulée après la clôture de la journée. Formulez des hypothèses explicites plutôt que d'ignorer ces questions.

2. Données et environnement mis à disposition

Environnement fourni

- un fichier `docker-compose.yml` de cluster Hadoop (1 NameNode, 1 ResourceManager, 2 à 3 DataNodes/NodeManagers), identique à celui du TD1 ;
- un dossier hôte `persistant/` monté dans le conteneur client : il **simule le stockage externe durable** (équivalent d'un object storage type S3). Ce qui n'est pas dans `persistant/` est considéré comme perdu quand le cluster est détruit ;
- un jeu de données réduit `marcheplus_lot_J.csv` (quelques milliers de lignes) et un second fichier `marcheplus_lot_J_correctif.csv` simulant l'arrivée tardive ;
- un squelette de scripts (`mapper.py`, `reducer.py` et un exemple de requête Hive) qui compte simplement les lignes par catégorie : il sert de point de départ, pas de solution.

Format des données

Un seul format de fichier CSV pour les trois flux :

```
date_evenement,commande_id,client_id,produit_id,categorie,type_evenement,montant,horodatage
```

Colonne	Description
date_evenement	Jour métier auquel l'événement est rattaché (AAAA-MM-JJ)
commande_id	Identifiant de la commande (vide pour une consultation)
client_id	Identifiant client (donnée à ne pas exposer dans les rapports)
produit_id / categorie	Produit et sa catégorie
type_evenement	`COMMANDE`, `CONSULTATION`, `ANNULATION` ou `RETOUR`
montant	Montant en euros (vide attendu pour `CONSULTATION`) ; pour `ANNULATION` et `RETOUR`, montant remboursé, positif
horodatage	Date et heure réelles d'enregistrement de l'événement

Un événement est dit **tardif** lorsque son `horodatage` est postérieur à la fin de la journée désignée par `date_evenement` (ou à la clôture de lot que vous définirez).

Le jeu contient des cas à détecter ou à traiter :

- lignes dupliquées ;
- champs vides (certains attendus, d'autres anormaux : à vous de distinguer) ;
- montants invalides (négatifs, non numériques, aberrants) ;
- événements tardifs ;
- lignes d'en-tête ;
- annulations ou retours associés à une commande, y compris une annulation dont la commande est absente du lot.

Vous pouvez compléter le jeu ou fabriquer quelques lignes problématiques si cela vous aide à tester vos choix.

Règles métier à définir

Avant de coder, décidez et documentez :

- comment calculer le chiffre d'affaires net ;
- comment traiter les doublons (qu'est-ce qui identifie un doublon ?) ;
- que faire des lignes invalides ou incomplètes ;
- comment classer les événements tardifs ;
- comment départager les produits à égalité dans le top 5 ;
- quelles données peuvent apparaître dans les sorties.

Il n'y a pas nécessairement une seule réponse possible. En revanche, une règle non documentée ne pourra pas être considérée comme un choix maîtrisé.

3. Votre mission

Concevez une solution Hadoop qui répond au besoin de MarchéPlus, puis démontrez-en une version réduite sur l'environnement fourni.

Le travail est organisé en **paliers**. Le palier 1 est obligatoire ; les paliers 2 et 3 sont à traiter dans l'ordre selon le temps disponible. Une équipe qui réussit parfaitement le palier 1 et documente bien ses choix vaut mieux qu'une équipe qui survole les trois.

Palier 1 (obligatoire) : un pipeline qui marche et dont on peut vérifier le résultat

A. Architecture et parcours des données

Définissez le chemin des données depuis leur arrivée jusqu'à la publication des résultats. Précisez notamment :

- où se trouvent les données brutes ;
- quelles données sont stockées dans HDFS et pendant combien de temps ;
- où se trouvent les résultats qui doivent survivre à la suppression du cluster (dans ce projet : le dossier `persistant/`) ;
- quels composants interviennent dans le traitement (rôle de HDFS, de YARN, du moteur de traitement) ;
- à quel moment un résultat peut être considéré comme publiable.

B. Traitement et qualité des données

Implémentez au minimum :

- une agrégation du chiffre d'affaires net par catégorie ;

- un traitement des lignes invalides, avec une sortie séparée permettant de les examiner (aucune ligne ne doit disparaître sans trace) ;
- une vérification de la cohérence d'au moins un résultat (par exemple : lignes en entrée = lignes valides + lignes rejetées + doublons écartés ; ou total du CA recalculé par une seconde méthode).

Le traitement peut utiliser **MapReduce Streaming**, **Hive**, ou une combinaison des deux. Expliquez votre choix. **Attendu : le contrôle et le rejet des lignes invalides doivent passer par du MapReduce Streaming, ou le choix contraire doit être explicitement justifié.**

N'utilisez pas Spark pour cette séance.

Palier 2 : résilience, relance et publication

C. Relance et publication des résultats

Décrivez **et démontrez** ce qui se passe si :

- le job échoue à mi-parcours ;
- le job est relancé avec le même lot ;
- des fichiers de sortie existent déjà ;
- un fichier tardif ou corrigé arrive le lendemain (utilisez le fichier correctif fourni).

Proposez un mécanisme qui évite qu'un résultat incomplet ou périmé soit présenté comme le résultat officiel. Pistes possibles (non imposées) : séparation sorties temporaires / validées, renommage de répertoire, fichier marqueur de validation, répertoires versionnés par lot. Expliquez votre logique et ses limites.

Pour rendre l'échec concret, provoquez-le réellement, par exemple : `yarn application -kill` en cours de job, arrêt d'un DataNode (`docker stop`), ou ligne piégée qui fait planter le reducer. Décrivez ce que vous observez (état YARN, fichiers laissés dans HDFS, comportement à la relance).

D. Persistance indépendante du cluster

Démontrez que vos résultats validés survivent à la destruction du cluster : copie vers `persistant/`, puis `docker compose down -v`, recréation du cluster, et restauration ou republication depuis `persistant/`.

Expliquez pourquoi **une réplication HDFS ne suffit pas à assurer la persistance des données après la destruction du cluster**.

Palier 3 : dimensionnement et extensions

E. Stockage et capacité (scénario réel)

Estimez le volume à stocker **dans l'espace de travail HDFS** (et non dans le stockage externe de 90 jours / 2 ans, que vous traiterez séparément en une ligne), en tenant compte :

- du volume moyen et du volume de pointe ;
- du **taux de décompression** que vous supposez (les 120 Go sont compressés : le volume décompressé est bien plus grand, précisez votre hypothèse) ;
- de la durée de conservation retenue pour les données de travail ;
- du facteur de réplication HDFS ;
- de la marge de capacité que vous souhaitez garder.

F. Dimensionnement et montée en charge

Proposez une configuration cible pour le scénario réel, même si vous ne pouvez tester que sur un petit cluster. Distinguez les rôles des nœuds et expliquez :

- dans quel cas vous ajouteriez des nœuds de calcul ;
- dans quel cas vous ajouteriez des nœuds qui stockent les blocs HDFS ;
- pourquoi ces deux décisions ne sont pas équivalentes ;
- quels indicateurs vous surveilleriez pour vérifier que le dimensionnement convient ;
- quelles limites pourraient apparaître lorsque le volume augmente.

Faites apparaître au moins un calcul de capacité **et** un calcul de débit (par exemple : débit nécessaire pour traiter le volume de pointe dans la fenêtre disponible). Indiquez les hypothèses utilisées et précisez pourquoi une extrapolation à partir du jeu de données réduit reste approximative.

G. Extensions (si le temps le permet)

- top 5 des produits les plus consultés par catégorie, avec règle de départage documentée ;
- nombre de commandes par catégorie ;
- liste d'anomalies enrichie (annulation sans commande, montants aberrants, etc.) ;
- pseudonymisation ou suppression de `client_id` avant toute sortie métier.

4. Contraintes de réalisation

- Travaillez avec l’environnement local fourni.
- **N’utilisez pas AWS** pour ce projet.
- Le cluster de démonstration ne doit pas être présenté comme équivalent à une infrastructure de production.
- Ne vous contentez pas d’un schéma théorique : votre pipeline doit produire un résultat vérifiable sur le jeu de données réduit.
- Vous pouvez vous répartir le travail, mais chaque membre de l’équipe doit être capable d’expliquer l’architecture et les limites de la solution (des questions individuelles seront posées à l'oral).

5. Déroulé indicatif de la séance

Durée indicative	Travail attendu
30 min	Lire le cas, relever les ambiguïtés et formuler les hypothèses
30 min	Dessiner l’architecture, répartir les tâches
1 h 45	Palier 1 : mise en œuvre du traitement et de la qualité des données
1 h 15	Palier 2 : échec, relance, publication, persistance externe
45 min	Palier 3 : estimation de capacité, scaling, limites
1 h 15	Rédaction finale du dossier, puis restitution (10 à 15 min par équipe, dont questions individuelles)

Vous pouvez organiser votre travail autrement, mais réservez du temps pour les tests et la restitution. Les paliers 2 et 3 peuvent être menés en parallèle par des sous-groupes différents.

6. Livrable

Remettez un dossier court comprenant :

1. un schéma d’architecture montrant les données, les stockages et les traitements ;
2. un registre des hypothèses et décisions ;
3. les scripts, requêtes ou commandes nécessaires pour reproduire le traitement ;
4. une preuve d’exécution : sortie du job, nombre de lignes traitées et résultat métier obtenu ;
5. un test de reprise ou d’échec, accompagné d’une explication du comportement observé ;
6. la preuve de survie des résultats après recréation du cluster ;
7. une estimation de capacité et de dimensionnement, avec vos hypothèses ;
8. une synthèse précisant ce qui devrait changer avant un déploiement réel.

Registre des hypothèses et décisions

Sujet	Hypothèse ou choix	Justification	Conséquence ou limite
Fichier tardif			
Annulation après clôture			
Doublons			
Lignes invalides			
Données conservées dans HDFS			
Données conservées hors cluster			
Relance du job			
Publication du résultat			
Taux de décompression			
Dimensionnement			

Restitution orale

Votre présentation doit répondre à la question suivante :

Quelle décision de votre équipe vous semble la plus importante pour éviter un résultat erroné ou une dépense inutile, et pourquoi ?

7. Critères d’évaluation

Critère	Insuffisant	Satisfaisant	Excellent
Cohérence de l’architecture	Schéma flou ou composants mal placés	Parcours complet, rôles corrects	Parcours complet, justifié par les contraintes (budget, rétention, cluster jetable)
Compréhension de Hadoop	Confusion HDFS / YARN / moteur	Rôles correctement distingués	Rôles distingués et illustrés par des observations sur le cluster
Qualité du traitement	Résultat non vérifiable, lignes perdues	CA net correct, rejets tracés, un contrôle de cohérence	Réconciliation complète des lignes, règles toutes documentées et testées
Gestion de la reprise	Reprise seulement décrite	Échec provoqué, relance sans doublon de résultat	Mécanisme de publication clair, cas du fichier tardif traité, limites identifiées
Dimensionnement	Chiffres sans hypothèses	Calcul de capacité avec hypothèses explicites	Capacité et débit, distinction calcul/stockage argumentée, indicateurs de supervision
Prise de recul	Prototype présenté comme production	Limites listées	Limites hiérarchisées, évolutions réalistes avant déploiement
Défense des choix	Réponses récitées, un seul membre répond	Chacun explique sa partie	Chacun explique l'ensemble et les compromis, y compris les alternatives écartées



Une solution qui traite le petit fichier le plus rapidement possible n'est pas nécessairement la meilleure. Vous serez aussi évalués sur votre capacité à distinguer **stockage temporaire et persistant**, **calcul et stockage**, **résultat calculé et résultat validé**, et à expliquer les compromis retenus.

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/eadl/bloc5/fm3/td3?rev=1793526684

Last update: **2026/11/01 10:51**