ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

# SUJET D'EXAMEN

**MATIERE : Big Data**

| | | | | | |
|---|---|---|---|---|---|
| **FILIERE :** | **IIR** | **NIVEAU :** | **4IIR** | **SEMESTRE :** | **S2** |
| **SESSION :** | **Normale** | **DUREE :** | **2H** | **DOCUMENTS :** | **Non autorisés** |
| **ANNEE UNIVERSITAIRE 2023-2024** | | | **ENSEIGNANT :** | **Dr. Abdelali El Gourari** | |

---

## Exercice 1 :

Vous êtes nouvel ingénieur Data dans une entreprise qui traite plusieurs téraoctets de données par jour (logs, images, documents). On vous demande d'expliquer aux parties prenantes non techniques les principes fondamentaux de l'architecture Big Data utilisée dans vos projets.

1. **Le Big Data en contexte réel**
   Expliquez les 5 caractéristiques majeures du Big Data connues sous le nom des "5V". Pour chaque V (Volume, Vélocité, Variété, Véracité, Valeur), donnez un exemple concret issu d'un environnement réel :

   - Volume :
   - Vélocité :
   - Variété :
   - Véracité :
   - Valeur :

2. **Comprendre Hadoop dans l'entreprise**
   L'architecture Hadoop repose sur trois composants majeurs. Associez chacun à sa fonction en quelques lignes :

   - **HDFS** : Quel est son rôle dans le stockage des données massives ?
   - **MapReduce** : Quel type de traitement effectue-t-il ? À quel coût en ressources ?
   - **YARN** : À quoi sert-il dans un cluster Hadoop multi-utilisateurs ?

3. **Architecture de Spark et intégration avec Hadoop**
   Votre entreprise utilise désormais Apache Spark pour accélérer les traitements.
   Expliquez brièvement le rôle des composants suivants dans une exécution Spark typique :

   - Driver Program
   - SparkContext
   - Cluster Manager (ex: YARN)
   - Executors
   - DataNodes (HDFS)

Puis, expliquez comment Spark accède aux données stockées dans HDFS et ce qui le rend plus rapide que MapReduce classique.

## Exercice 2 :

Vous êtes chargé d'extraire le contenu textuel de milliers de fichiers PDF stockés dans HDFS (`hdfs:///data/pdfs/*.pdf`) afin de constituer un corpus exploitable par une IA. Vous utiliserez PySpark, avec la bibliothèque pdfplumber.

1. Expliquez comment lire une liste de fichiers PDF présents dans un répertoire HDFS avec Spark.
    - Quel type de fichier est utilisé pour la lecture ?
    - Quelle fonction permet de créer un DataFrame contenant les chemins des fichiers ?
2. Complétez la fonction suivante (UDF) pour extraire le texte brut d'un fichier PDF :

```
import pdfplumber

def extract_clean_text(path):
    try:
        with pdfplumber.open(path) as pdf:
            text = ''
            for page in pdf.pages:
                text += page.extract_text() + '\n'
        # Nettoyage
        text = _________  # supprimer les espaces multiples
        return _________  # retirer les espaces inutiles
    except:
        return ""
```

3. Donnez le code PySpark pour :
    - Créer un DataFrame avec les chemins des fichiers PDF.
    - Appliquer la fonction d'extraction ci-dessus sur chaque fichier.
    - Ajouter une colonne length contenant le nombre de caractères du texte extrait.
4. Expliquez pourquoi Spark est adapté au traitement de fichiers PDF à grande échelle, même si ceux-ci ne sont pas structurés.

## Exercice 3:

Une entreprise de médias souhaite analyser la répartition de ses fichiers images stockés dans HDFS sous le chemin `hdfs:///medias/photos/*.jpg`. Vous êtes chargé de concevoir un traitement distribué avec Spark pour :

- lire les métadonnées des images,
- calculer la taille en Ko de chaque image,
- classer les images par tranches de taille définies par l'équipe projet.

**Tranches à utiliser :**

- Petite taille : `≤ 200 Ko`
- Moyenne taille : `201 – 500 Ko`
- Grande taille : `501 – 1024 Ko`
- Très grande taille : `> 1024 Ko`

Complétez ce script Spark pour effectuer les étapes ci-dessous :

- Lire les images en format binaire.
- Calculer leur taille en Ko.
- Catégoriser chaque image dans une tranche de taille.
- Compter le nombre d'images dans chaque tranche.

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, when

spark = SparkSession.builder.appName("AnalyseImages").getOrCreate()

df = spark.read.format("binaryFile").load("hdfs:///medias/photos/*.jpg")

df_ko = df.withColumn("taille_ko", col("length") / 1024)

df_catégorisé = df_ko.withColumn("tranche", when(__________, "petite")
                                  .when(__________, "moyenne")
                                  .when(__________, "grande")
                                  .otherwise("très_grande"))

resultat = df_catégorisé.groupBy("tranche").count()
resultat.show()
```

2. **MapReduce vs Spark**
   Comparez ce traitement avec une implémentation classique en Hadoop MapReduce. Quelles seraient les étapes :

   - du Mapper ?
   - du Reducer ?
   - du Job complet ?

3. **Analyse de performance**
   Expliquez pourquoi Spark est nettement plus rapide que MapReduce dans ce cas :

   - sur la gestion des fichiers,
   - sur la mise en cache,
   - sur le parallélisme et la tolérance aux pannes.