Cet examen vise controler vos connaissances lissue du module Hadoop, Spark & MapReduce. Merci d'indiquer votre nom et prenom sur la page de garde.
Si certaines reponses au QCM vous paraissent ambigues / plusieurs reponses semnblent possibles, choisissez la plus precisc. Une reponse par question. Lisez bien les questions.Certaines sont susceptibles de vous induire en erreur. Vous pouvez également rajouter une courte justification si vous n'etes pas 100% sur de la réponse (une reponse fausse pour une raison
valable peut exceptionnellement être acceptcée. Dans le contexte de l'examen, un tuple designe un couple (clef;valeur).

La derniere question est une question ouverte, elle compte pour plus de points. Aucun document n'est autorisé.

Q1.

Parmi les propositions suivantes, selectionnez celle qui decrit le mieux l'operation shuffle au sein du paradigme map/reduce tel qu'implementé dans Hadoop.

- [x] Une operation executee **apres map** qui **regroupe** ensemble les tuples, ayant **la même clef**
- [ ] Une operation executee apres reduce qui regroupe ensembles les tuples ayant la même clef
- [ ] Une operation executee après map qui mélange les tuples ensemble pour produire un ensemble representatif
- [ ] Une operation executee apres reduce qui melange les tuples ensemble pour produire un ensemble representatif

1. **Réponse:** `1.` Une operation executee apres map qui regroupe ensemble les tuples, ayant **la même clef**

   Explication: Dans le paradigme MapReduce tel qu'implémenté dans Hadoop, après l'étape Map et avant l'étape Reduce, il y a une phase intermédiaire appelée Shuffle (et Sort). Pendant cette phase, les tuples générés par les fonctions Map sont regroupés par clé. Ainsi, tous les tuples ayant la même clé sont envoyés au même réducteur.

Q2.

Quelles sont les quatre étapes principales du paradigme map/reduce tel que decrit dans le cours ?

- [ ] input, distribute, map, reduce
- [x] split, map, shuffle, reduce
- [ ] input, map,reduce, output
- [ ] split, map, reduce,analyze

1. **Réponse:** `2`. split, map, shuffle, reduce

   Explication: Le paradigme MapReduce comporte plusieurs étapes:

   - **Split**: Les données d'entrée sont divisées en fragments (splits) qui sont ensuite traités par des fonctions Map distinctes.
   - **Map**: Chaque split est traité par une fonction Map qui produit des tuples (clé, valeur).
   - **Shuffle (et Sort)**: Après le Map, les tuples sont regroupés par clé afin que tous les tuples ayant la même clé soient envoyés au même réducteur.
   - **Reduce**: Les tuples regroupés par clé sont traités par la fonction Reduce pour produire une sortie consolidée.

Q3.

Selectionnez ci-dessous la proposition qui est **fausse**.

- [x] Un programme map/reduce execute sur un fichier texte doit toujours decouper les donnees d'entree par ligne
- [ ] L'operation map produit des tuples a partir des donnees d'entree
- [ ] L'operation reduce produit des tuples
- [ ] Le paradigme map/reduce permet le developpement de programes dont l'execution peut être parallelisee facilement

1. **Réponse:** `1.` Un programme map/reduce execute sur un fichier texte doit toujours decouper les donnees d'entree par ligne

   Explication: Bien que de nombreux programmes MapReduce divisent les fichiers texte par ligne, ce n'est pas une exigence stricte. Le découpage des données d'entrée dépend du `InputFormat` utilisé. Par exemple, le `TextInputFormat` divise les données par ligne, mais d'autres formats peuvent diviser les données différemment.

Q4.

Qu'a-t on generalement en **entree** d'une fonction **reduce** ?

- [x] Une **serie de valeurs** associees à une **clef distincte**
- [ ] Un ensemble de couple (clef;valeur)
- [ ] Des fragments de donnees d'entree
- [ ] Un unique couple (clef;valeur) ou une valeur littérale

1. **Réponse:** `1.` Une serie de valeurs associees à une clef distincte

   Explication: La fonction Reduce reçoit une clé et une série de valeurs associées à cette clé unique. La fonction peut alors effectuer une opération (comme une sommation) sur ces valeurs et produire une ou plusieurs sorties.

Q5.

Dans le contexte de Hadoop, quel est i'interet de HDFS ?

- [ ] HDFS permet de lire des donnees depuis n'importe ou dans le monde, du moment qu'une connexion est disponible
- [ ] HDFS permet de stocker des donnees sans disque dur, en utilisant la RAM uniquement
- [ ] C'est la seule manière pour Hadoop de lire les donnees d'entrée d'un programme
- [x] HDFS permet de stocker des donnees de manière **distribuée sur un cluster** Hadoop

1. **Réponse:** `4` . HDFS permet de stocker des donnees de maniere distribuée sur un cluster Hadoop

   Explication: HDFS, ou Hadoop Distributed File System, est le système de fichiers distribué de Hadoop. Son principal avantage est qu'il permet de stocker de grands volumes de données de manière distribuée sur plusieurs machines, offrant ainsi une redondance et une résilience en cas de défaillance d'un nœud. De plus, il est conçu pour travailler efficacement avec le paradigme MapReduce.

Q6.

Quelle est la principale difference fondamentale, non liée aux performances, dans l'application du paradigme map/reduce entre un framework comme Hadoop et un frameworkcomme Spark ?

- [x] Le moteur d'éxcution sauvegarde un maximum de donnees **en memoire** dans **Spark** par rapport a Hadoop
- [ ] Les frameworks comme Spark permettent de supporter des clusters bien plus larges qu'Hadoop en terme de machines.
- [ ] Les frameworks comme Spark ont une interpretation moins formelle du paradigme; et non limitée necessairement à des fonctions rigides < map > et 《 reduce 》 implementer.
- [ ] Les frameworks comme Spark supportent plus de langages de programmation que Java uniquement, au contraire de Hadoop.

1. **Réponse:** `1` . Le moteur d'éxcution sauvegarde un maximum de donnees en memoire dans Spark parrapport a Hadoop

   Explication: La principale différence entre Spark et Hadoop est que Spark tente de garder les données en mémoire autant que possible, contrairement à Hadoop qui écrit souvent des données intermédiaires sur le disque. Cela donne à Spark un avantage en termes de vitesse, surtout pour les applications nécessitant de multiples opérations sur les données.

Q7.

Comment HDFS assure-t'il la tolerance la panne / évite-t'il la perte de donnees en cas de panne ?

- [x] Il duplique les donnees sur les **disques** durs de **plusieurs noeuds**
- [ ] Il duplique les donnees dans la memoire vive (RAM) de plusieurs noeuds
- [ ] Il n'évite pas la perte de donnees, mais il peut récuperer les donnees manquantes depuis leur source initale a nouveau en cas d'échec
- [ ] Il demmande explicitement à l'utilisateur de fournir les donnees manquantes en cas de panne

1. **Réponse:** `1.` Il duplique les donnees sur les disques durs de plusieurs noeuds

   Explication: HDFS assure la tolérance aux pannes en dupliquant chaque bloc de données sur plusieurs nœuds du cluster. Par défaut, chaque bloc est dupliqué trois fois, mais cette valeur est configurable.

Q8.

Dans la classe qui contient la fonction reduce d'un programme Java Hadoop, on a cette déclaration:

```
1   public void reduce(Text a1, Iterable<Text> a2, Context a3)
```

A quoi correspondent respectivement les arguments a1 et a2 ?

- [ ] a1 est un couple (clef;valeur); et a2 la liste des fichiers d'entree sur HDFS
- [ ] a1 est une clef, et a2 une liste iterable de couples (clef;valeur)
- [ ] a1 est une clef, et a2 un couple (clef;valeur) unique
- [x] a1 est une clef; et a2 une **liste iterable** de valeurs

    1. **Réponse:** `4.` a1 est une clef; et a2 une liste iterable de valeurs

       Explication: Dans la fonction `reduce`, `a1` est la clé et `a2` est une liste itérable des valeurs associées à cette clé.

Q9.

Quelles sont les trois classes principales qu'on doit implmenter à minima dans un programme map/reduce Hadoop ?

- [ ] Driver, Configuration, FileSystem
- [ ] Driver, Chopper, Executer
- [ ] Driver, MapReducer, Analyzer
- [x] Driver,Mapper,Reducer

    1. **Réponse:** `4` . Driver,Mapper,Reducer

       Explication: Dans un programme MapReduce avec Hadoop, on doit typiquement implémenter les classes `Driver`, `Mapper` et `Reducer`.

Q10.

On a un fichier texte < in.txt > contenant des données au format suivant:

```
John,1987
Jane,1988
Bob,1992
```

On excute le programme Spark suivant:

```
data=sc.textFile('hdfs:///in.txt')
tuples=data.map(lambda x: (x.split(',')[0], int(x.split(',')[1])))
results=tuples.groupBy(lambda x: x[1]-x[1]%20)
```

Quelle tache accompli-t'il ?

- [ ] Il regroupe les enregistrements par date de naissance.
- [ ] Il filtre les enregistrements par date de naissance, en enlevant les personnes agées de moins de 20 ans.
- [ ] Il regroupe les enregistrements par date de naissance, en **séparant** les personnes agées **de** moins de 20 ans des autres.
- [x] Il **regroupe** les enregistrements par **double-décennie** de naissance (1960, 1980, etc,).

    1. **Réponse:** `4`. Il regroupe les enregistrements par double-décennie de naissance (1960, 1980, etc.).

       Explication: Le code Spark divise l'année de naissance par 20 et arrondit, regroupant ainsi les personnes par double-décennie.

Q11.

On imagine qu'on souhaite développer un programme Hadoop capable de detecter et regrouper les anagrammes (mots composés des mêmes lettres mais pas dans le même ordre, par exemple < beau > et < aube >) à partir d'une liste de mots en entrée.

Au sein du programme, et en sortie de la fonction map, on decide d'utiliser comme clef des couples (clef,valeur) le nombre de lettres du mot fourni en entree; I'idée étant que les anagrammes ayant le même nombre de lettres, cela permettra de les regrouper.Pourquoi cette approche, bien que fonctionnelle, n'est-elle pas la plus optimale ?

- [ ] Parce qu'elle se contente de produire des couples (clef;valeur) en comptant le nombre de lettres, alors qu'on pourrait détecter les anagrammes des la fonction map en analysant toute la liste.

- [x] Parce qu'elle ne tire pas parti au mieux de l'**execution parallele;** on aura au maximum N groupes, N étant la taille du plus long mot présent dans la liste.
- [ ] Parce qu'elle va necessiter un code trop complexe dés la fonction map; on va devoir analyser tout le mot recu pour compter les lettres.
- [ ] Parce que cette approche va necessiter des tris inutiles entre map et reduce; on devrait tenir compte des syllabes, pas des lettres.

1. **Réponse:** `2` . Parce qu'elle ne tire pas parti au mieux de l'execution parallele; on aura au maximum N groupes, N étant la taille du plus long mot présent dans la liste.

   Explication: Bien que cette approche puisse fonctionner, elle n'est pas optimale en termes d'exécution parallèle. En utilisant simplement la longueur du mot comme clé, on limite le parallélisme possible, car tous les mots de la même longueur se retrouveront dans le même groupe. Une meilleure clé serait peut-être une représentation des lettres du mot triées, ce qui permettrait de regrouper tous les anagrammes, indépendamment de leur longueur.

Q12.

Dans la classe principale d'un programme map/reduce Hadoop, on excute la ligne

```
job.setMapOutputKeyClass(Text.class);
```

Et dans la classe implmentant l'opération reduce, on a la ligne:

```
public class MyReduce extends ...<IntWritable, Floatwritable,
IntWritable, FloatWritable>
```

Quel est le problème ici avec la seconde ligne de code (en ignorant la partie < ... > qui contiendrait normalement le nom de la classe mère) ?

- [x] Le **premier** type spécifié dans les types génériques devrait être "Text"
- [ ] Le premier et troisieme type spécifiés dans les types génériques devraient être "Text'
- [ ] Tous les types spécifiés dans les types génériques devraient être "Text"
- [ ] Le second type spécifié dans les types generiques devrait être "Text"

1. **Réponse:** `1.` Le premier type spécifié dans les types génériques devrait être "Text"

Explication: Le problème est que le type de sortie clé du mappeur est défini comme `Text` (dans `job.setMapOutputKeyClass(Text.class)`), mais le reducer utilise `IntWritable` comme type de clé d'entrée. Les types de clés de sortie du mappeur doivent correspondre aux types de clés d'entrée du reducer.

Q13.

Dans le contexte de Spark, qu'est ce qu'un RDD ?

- [ ] Un "“Random Distributed Datapoint"; permet d'obtenir un ensemble statistiquement significatif de nos données
- [ ] Un "Random Data Drive"; un espace de stockage de données quelque part sur le cluster
- [x] Un "**Resilient** Distributed Dataset"; la principale abstraction Spark pour des donnees sur lesquelles on travaille
- [ ] Un "Replicated Deconstructed Datapoint"; le principal mecanisme de tolérance à la panne de Spark

1. **Réponse:** `3`. Un "Resilient Distributed Dataset"; la principale abstraction Spark pour des donnees sur lesquelles on travaille

   Explication: Un RDD, ou Resilient Distributed Dataset, est l'abstraction de données fondamentale de Spark. Il s'agit d'une collection distribuée et immuable d'objets qui peut être traitée en parallèle.

Q14.*

On éxecute le code Spark suivant:

```
rdd=sc.parallelize([(1,20), (8,12), (5,8), (1,8), (3,3)])
res=rdd.mapValues(lambda x: 1 if x<10 else x)
```

... la premiere ligne chargeant les couples (clef;valeur) indiques dans le RDD < data >. Combien de groupes seront presents à l'issue de l'excution dans le RDD < res > ?

- [ ] Il y aura cinq groupes.
- [ ] Il y en aura deux.
- [x] Il y aura un groupe par clef distincte.
- [ ] Il y aura un groupe par valeur distincte.

1. **Réponse:** `3.` Il y aura un groupe par clef distincte.

   Explication: La méthode `mapValues` applique une fonction à chaque valeur du RDD sans changer les clés. Par conséquent, le nombre de groupes distincts restera le même, correspondant au nombre de clés distinctes, qui est déterminé par les premiers éléments de chaque tuple dans le RDD.

Q15.

Comment Spark assure-t'il la tolerance à la panne pour les données ?

- [ ] Il duplique les donnees en blocs; tout bloc perdu est present sur au moins un autre noeud du cluster
- [ ] Il stocke des checkpoints sur le disque; toute donnée perdue peut-être récupérée depuis le dernier checkpoint
- [ ] Il persiste les donnees sur le cluster manager central; si un nceud perd des donnees, on peut les récupérer depuis ce cluster manager central
- [x] il **stocke** la séquence d'**operations** permettant d'aboutir aux donnees; si un noeud est perdu.les donnees en question peuvent être recalculées partir de cette séquence

  1. **Réponse:** `4.` il stocke la séquence d'operations permettant d'aboutir aux donnees; si un noeud est perdu.les donnees en question peuvent être recalculées à partir de cette séquence

     Explication: Spark assure la tolérance aux pannes en conservant la séquence d'opérations (linéage) qui a été appliquée à l'ensemble de données de base pour obtenir le RDD. En cas de perte d'une partition d'un RDD due à une défaillance d'un nœud, Spark peut recalculer cette partition à partir de l'ensemble de données d'origine en appliquant la séquence d'opérations stockée.

Q16.

On execute le code Spark suivant:

```
data1=sc.parallelize([1,2,3,4])
data2=datal.map(lambda x: (x,x%2))
```

Combien d'elements aura-ton dans data2 apres I'evaluation ?

- Chaque élément de `data1` est mappé vers un tuple. Par conséquent, le nombre d'éléments dans `data2` sera le même que celui de `data1`.

  Réponse: `data2` aura 4 éléments.

Q17.

On execute le code Spark suivant:

```
1  data1=sc.parallelize([1,2,10,10])
2  data2=datal.reduce(lambda x,y: x*y if x==2 else x+y)
```

La valeur data2 renvoyée sera un entier. Quelle sera sa valeur ?

- L'opération `reduce` fonctionnera de la manière suivante :
  - Pour 1 et 2 : `1+2` (car `x` n'est pas égal à 2) = 3
  - Pour 3 (résultat précédent) et 10 : `3+10` = 13
  - Pour 13 (résultat précédent) et 10 : `13+10` = 23

  Réponse: La valeur de `data2` sera 23.

Q18.

On imagine qu'on dispose d'une liste de ventes effectuées dans un magasin. Pour chacune d'entre elle, on a le **montant**, **le jour de la semaine ou la vente a été effectuée**, la **date** exacte de la vente, et le nom du produit.

Les donnees sont tres larges et on vous demande en consequence d'implementer un programme map/reduce Hadoop pour établir des statistiques sur le montant des ventes **selon le jour de la semaine,** mais seulement partir de l'annee 2000. Les donnees sont fragmentées par vente (une ligne = une vente).

Indiquez quelle clef vous choisiriez dans votre programme, de quels types de variable seraient vos clefs et valeurs, et decrivez rapidement les opérations que feraient alors vos fonctions map et reduce.

- **Reponse**:
  - **Clef:** Vous choisiriez le jour de la semaine comme clef puisque vous souhaitez regrouper les ventes en fonction de cela.

    Types de variable:

- Clef: Text (pour le jour de la semaine)
- Valeur: DoubleWritable (pour le montant de la vente)

Fonction map:

1. Parsez chaque ligne pour extraire le montant, le jour de la semaine, la date de vente et le nom du produit.
2. Si la date de vente est postérieure à l'année 2000, émettez le couple (jour de la semaine, montant de la vente).

Fonction reduce:

1. Pour chaque clef (jour de la semaine), sommez tous les montants de vente associés.
2. Émettez le résultat comme (jour de la semaine, somme totale des ventes).

Explication: La phase `map` du processus consiste à filtrer les données pour ne considérer que les ventes postérieures à l'année 2000 et à émettre le jour de la semaine comme clef avec le montant de vente comme valeur. Pendant la phase `reduce`, toutes les valeurs (montants des ventes) pour un jour de la semaine donné sont additionnées pour donner une somme totale des ventes pour ce jour.