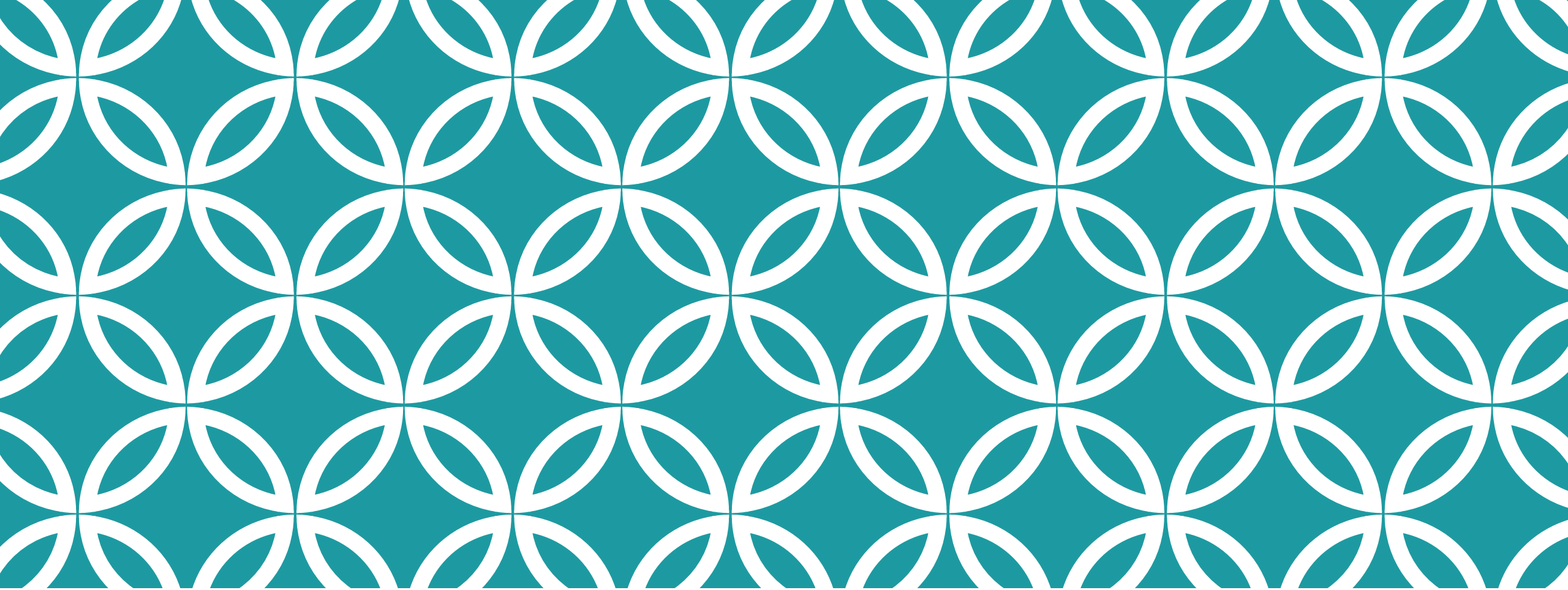
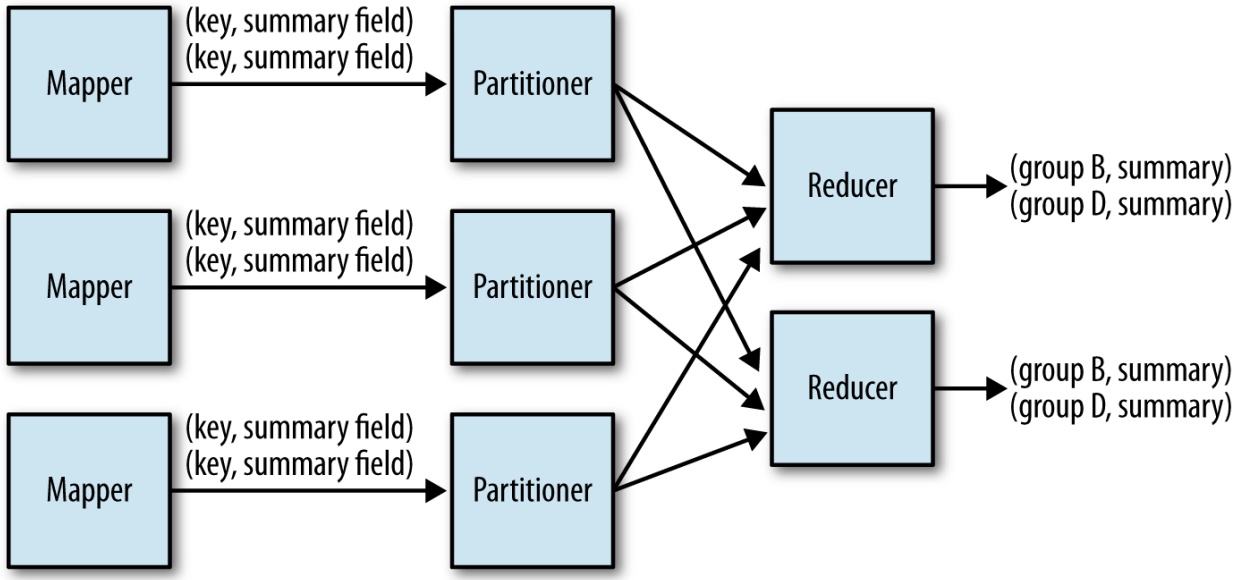


Pattern de Résumé

Pattern de résumé :



Le calcul du min, du max et de la moyenne se font de la même manière et ne pose aucun problème. Le mappeur envoie un message <clé, résuméTemporaire> partiel, qui est ensuite agrégé par le combiner et le reducer s'occupe de faire le calcul du résuméFinal avec le informations de résumé temporaire.

Pattern de résumé : Médiane, écartype, …

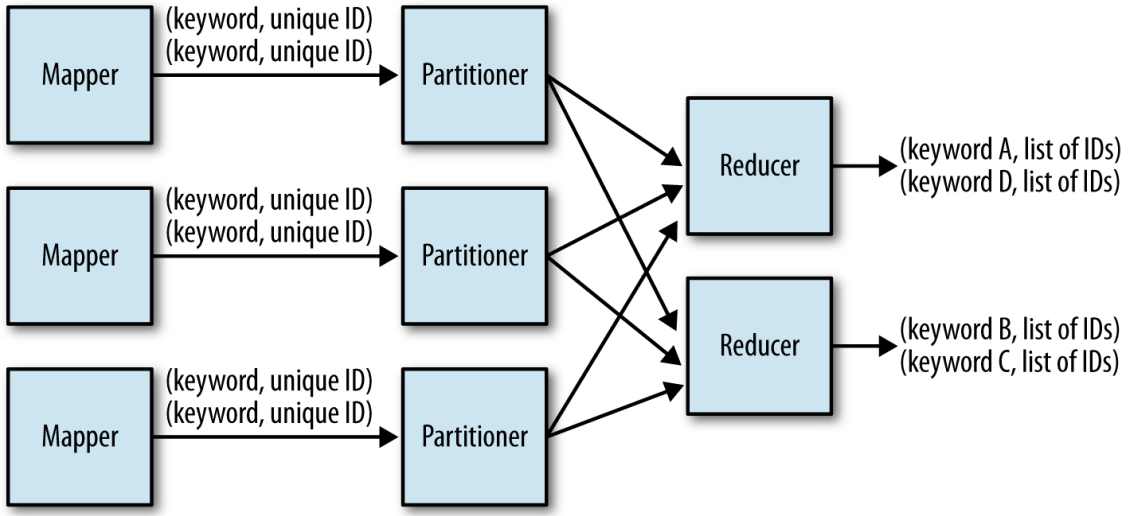
Parfois il n’est pas possible de créer des résumés temporaire comme ce que l’on a fait pour la moyenne. C’est le cas pour le calcul de la médiane ou de l’écart type.

Une optimisation est quand même possible, il consiste à « compresser » les messages qui ont la même valeur. Pour cela:

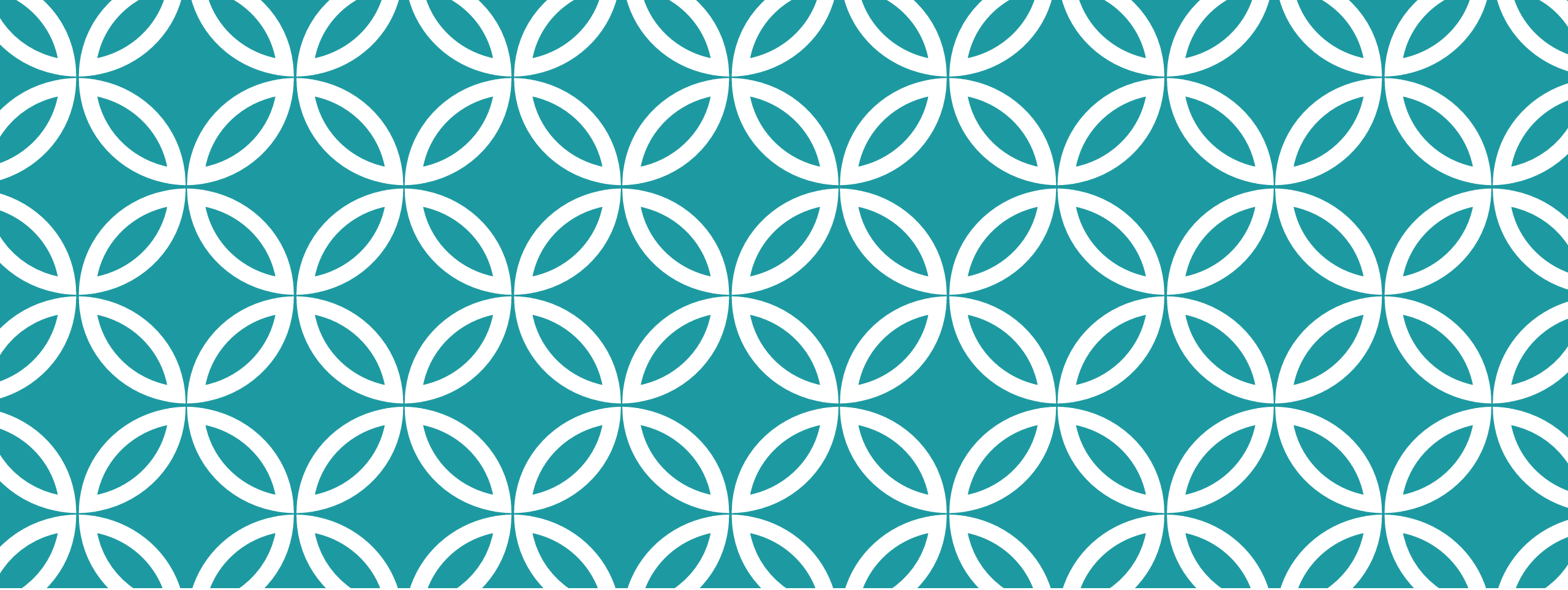
- Au lieu de faire passer des messages de type <clé, valeur> on fait passer des messages de type <clé, SortedMapWritable< valuer, count> >, en d’autre terme on fait passer un histogramme de fréquence des valeurs. Cette histogramme peut être combiné sans perte d’information.

- Avec cette optimisation une valeur n’est transmise que #Datanode fois au plus sur le réseau. Le temps de calcul est donc lié au nombre de valeur différentes et non plus au nombre de valeurs.

Pattern de résumé : Index inversé

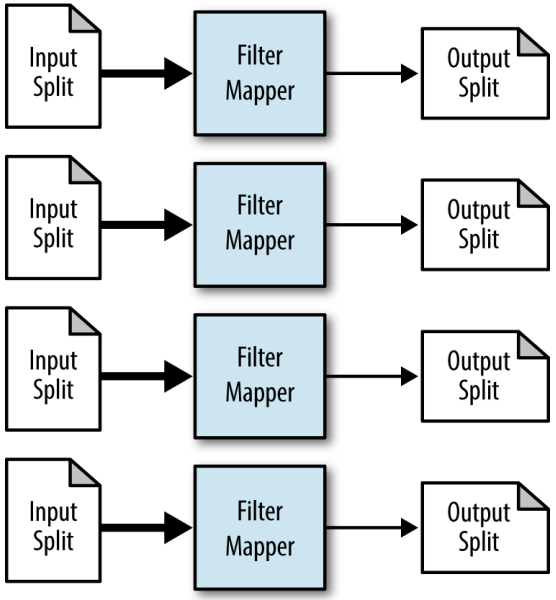


Un index inversé consiste à créer un fichier qui liste tous les index qui ont une valeur particulière. Cela revient, a « renverséé » la clé et la valeur pour créer un fichier de type <valeur, {clés}>. Le code du mappeur consiste juste à faire le renversement et le code du reducer à concaténer les clés pour les passer à l'OutputFormat.



Pattern de filtrage

Pattern de filtrage: Simple



Le mappeur peur être utilisé directement comme un filtre. Il suffit pour cela de ne renvoyer que les <clé, valeurs> passées par le RecordReader que l'on veut garder. Il n'y a donc pas besoin de combiner ni de reducer pour faire un filtre simple.

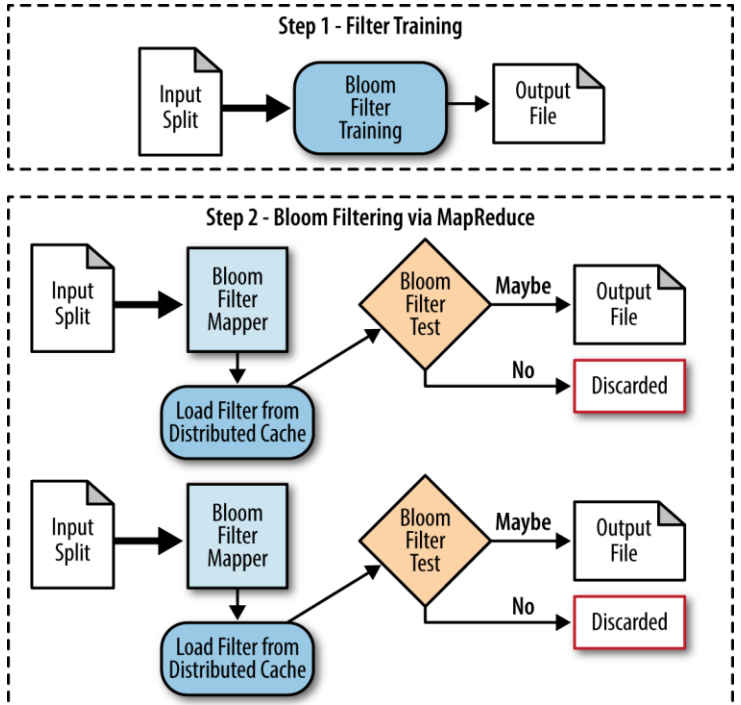
Pattern de filtrage: exemple grep

```
public static class GrepMapper extends Mapper<Object, Text, NullWritable, Text> {
 private String mapRegex = null;
 public void setup(Context context) throws IOException,
   InterruptedException {
   mapRegex = context.getConfiguration().get("mapregex");
}

public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
  if (value.toString().matches(mapRegex))  context.write(NullWritable.get(), value);
 }

}
```

Pattern de filtrage: exemple sampling

```
public static class SamplingMapper extends Mapper<Object, Text, NullWritable, Text> {
 private Double percentage;
 protected void setup(Context context) throws IOException, InterruptedException {
  percentage = context.getConfiguration().getDouble("sampling_percentage", 1.);
 }
 public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
  if (Math.random() < percentage) context.write(NullWritable.get(), value);
 }
}
```

Pattern de filtrage: Filtre de bloom



Le filtrage par filtre de bloom consiste à utiliser une structure de donnée permettant de tester sans faux négatif si un élément est dans un ensemble mais avec un nombre de faux positifs dépendant de la taille du filtre. Cela permet de faire des tests très rapide sans parcourir l'ensemble des éléments d'un ensemble.

Pattern de filtrage: Filtre de bloom

Un filtre de bloom est constitué d'un vecteur de bits et d'un ensemble de fonctions de hachage. A chaque fois que l'on insère un élément dans le filtre tous les bits correspondants aux clés de hash retournées par les fonctions de hash sont mis à 1.

Pour tester si un élément est dans le filtre il suffit de vérifier que tous ses bits sont à 1.

Hadoop fournit une filtre de Bloom de type writbale que vous pouvez donc enregistrer et lire dans vos mappeur/reducer via le cache distribué.

org.apache.hadoop.util.bloom.BloomFilter

Pattern de filtrage: Top K

Le design pattern du top K consiste à sélectionner les K plus grands/petits/etc.. éléments d'un ensemble. Le calcul des K plus grands étant associatif, on peut distribuer le calcul efficacement. Le problème est que l'on ne peut pas savoir si un élément fait partie des K plus grands éléments locaux tant que l'on a pas parcouru tous les éléments locaux.

Pour implémenter ce le top K on va utiliser les fonctions setup et cleanup des map. Au setup, nous allons initialiser un arbre binaire équilibre TreeMap (Java) ou SortedMapWritable(Hadoop). La map sera initialisé au setup, la fonction ne fera que lire les éléments et les insérer dans la treemap si nécessaire. Dans ce cas la fonction map n'envoie pas de message. C'est la fonction cleanup (appelé après le traitement de tout le flux) qui va envoyer les messages <clé, valuer> ou le message < null, SortedMapWritable<valeur, clé>>.

Pattern de filtrage: Top K

```
public static class TopKMapper extends Mapper<Object, Text, NullWritable, Entreprise> {
public int k = 0;
private TreeMap<Float, Entreprise> topKEntreprises = new TreeMap<Float, Entreprise>();
@Override
public void setup(Context context) {
 Configuration conf = context.getConfiguration();
 k = conf.getInt("k", 10);
 String dateString = conf.get("date", "1430723161");
}
```

Pattern de filtrage: Top K

```
public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String tokens[] = value.toString().split("\\|");
        …..
        topKEntreprises.put(var, new Entreprise(name, last, var, open, hight, low, var_an, time));
        if (topKEntreprises.size() > k)
                topKEntreprises.remove(topKEntreprises.firstKey());

}
```

Pattern de filtrage: Top K

```
public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
        String tokens[] = value.toString().split("\\|");
        …..
        topKEntreprises.put(var, new Entreprise(name, last, var, open, hight, low, var_an, time));
        if (topKEntreprises.size() > k)
                topKEntreprises.remove(topKEntreprises.firstKey());

}

protected void cleanup(Context context) throws IOException, InterruptedException {
  for (Entreprise c : topKEntreprises.values()) {
   context.write(NullWritable.get(), c);
  }
}
```

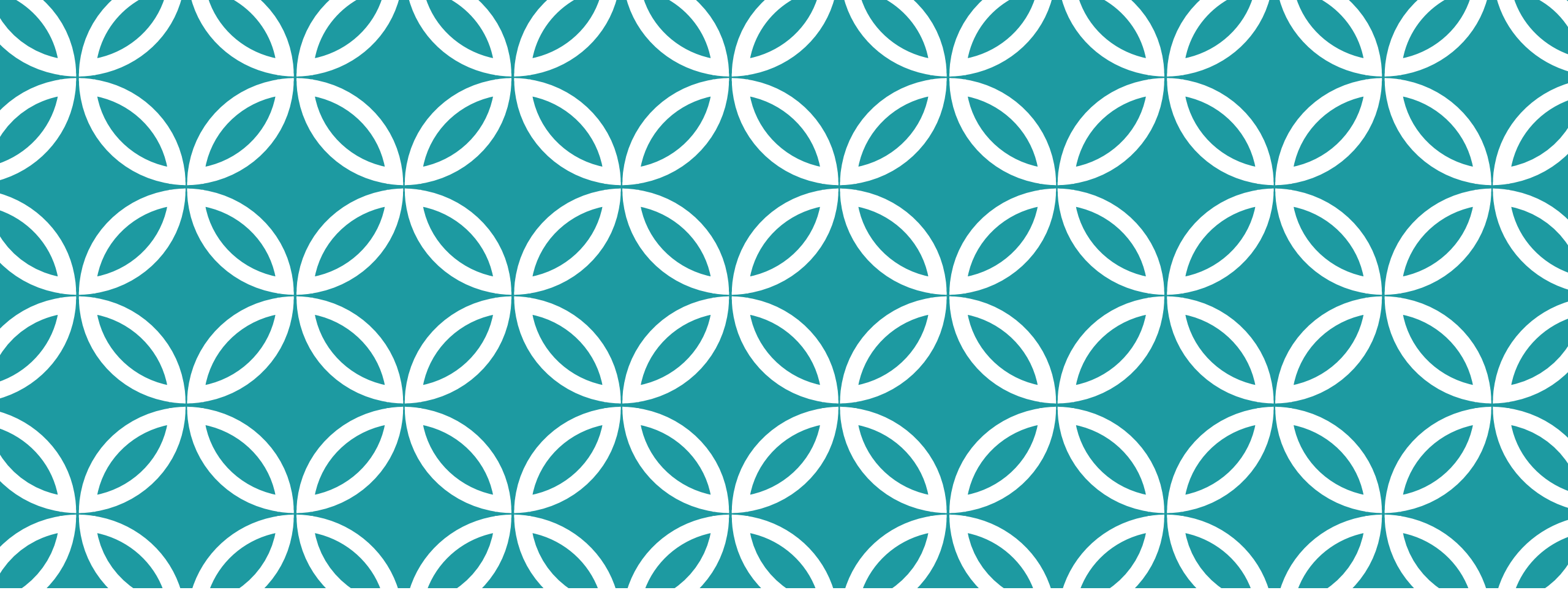
Pattern de filtrage: Top K

```
public static class TopKReducer extends Reducer<NullWritable, Entreprise, NullWritable, Entreprise> {
public int k = 0;
private TreeMap<Float, Entreprise> topKEntreprises = new TreeMap<Float, Entreprise>();
public void setup(Context context) {
  Configuration conf = context.getConfiguration();
  k = conf.getInt("k", 10);
}
public void reduce(NullWritable key, Iterable<Entreprise> values, Context context) throws IOException,
InterruptedException {
 for (Entreprise value : values) {
   topKEntreprises.put(value.getVar(), new Entreprise(value));
   if (topKEntreprises.size() > k)
    topKEntreprises.remove(topKEntreprises.firstKey());
 }
 for (Entreprise c : topKEntreprises.descendingMap().values()) {
   context.write(key, c);
 }
}
}
```

Pattern de filtrage: Eléments distinct

Le pattern distinct consiste à supprimer les doublons dans les données. Il consiste uniquement à utiliser le mécanisme de « shuffling » pour effectuer l'opération.

```
public static class DistinctMapper extends Mapper<Object, Text, Text, NullWritable> {
public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
    context.write(value, NullWritable.get());
  }
}

public static class DistinctReducer extends
Reducer<Text, NullWritable, Text, NullWritable> {
public void reduce(Text key, Iterable<NullWritable> values, Context context) throws IOException,
InterruptedException {
    context.write(key, NullWritable.get());
  }
}
```



Pattern de jointure

Inner join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.

Inner JOIN A + B sur la clé ID: Tous les éléments qui ont la même valeur de ID sont fusionnés

Left outer join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
4	12946	New York, NY	null	null	null
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.
9	3443	Oakland, CA	null	null	null

Left Outer Join A + B sur la clé ID: Tous les éléments qui ont la même valeur de ID sont fusionnés ce de A sont gardé avec des champs null dans les colonnes de B

Right outer join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.
null	null	null	8	48675	HTML is not a subset of XML!

Right Outer Join A + B sur la clé ID: Tous les éléments qui ont la même valeur de ID sont fusionnés ce de B sont gardé avec des champs null dans les colonnes de A

Full outer join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
4	12946	New York, NY	null	null	null
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.
null	null	null	8	48675	HTML is not a subset of XML!
9	3443	Oakland, CA	null	null	null

Full outer Join A + B sur la clé ID: Tous les éléments qui ont la même valeur de ID sont fusionnés ce de A et B sont gardé avec des champs null dans les colonnes de B ou B

Anti join

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
4	12946	New York, NY	null	null	null
null	null	null	8	48675	HTML is not a subset of XML!
9	3443	Oakland, CA	null	null	null

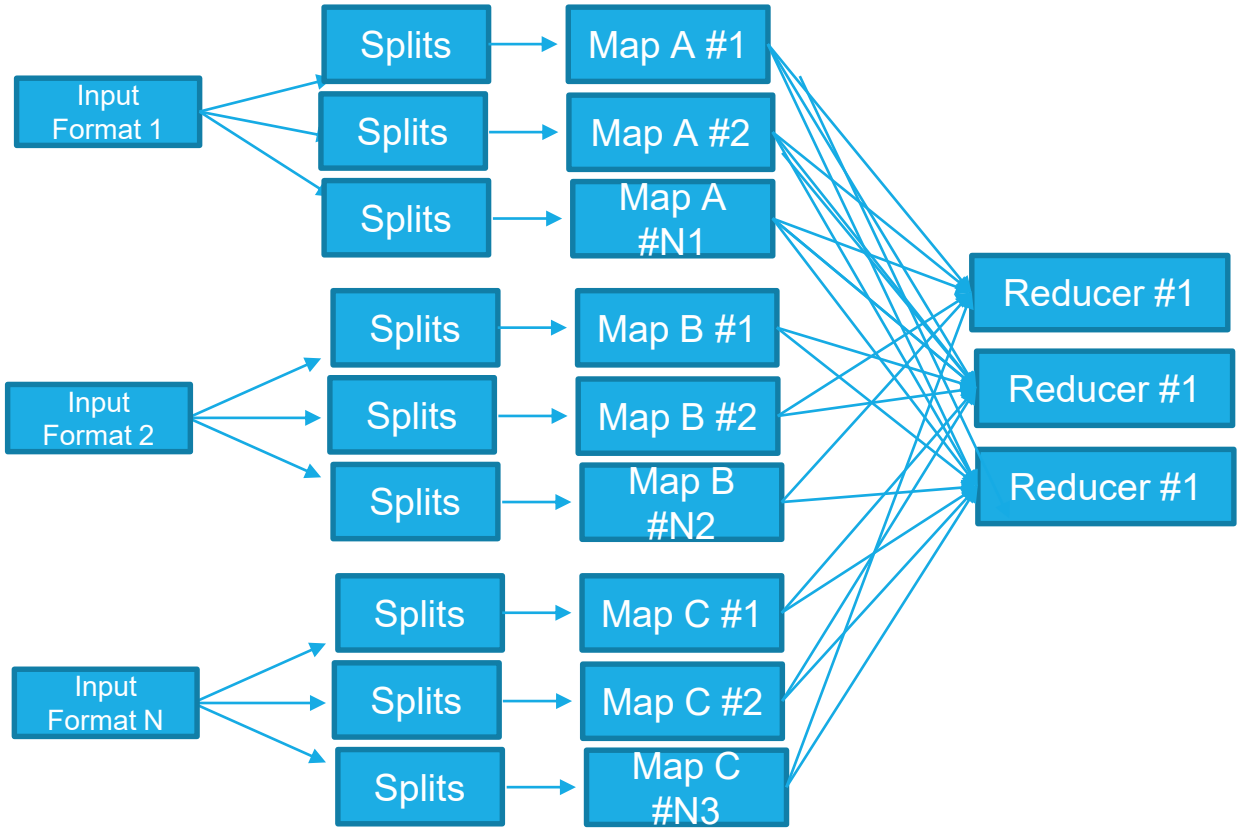
Cartesian product

User ID	Reputation	Location
3	3738	New York, NY
4	12946	New York, NY
5	17556	San Diego, CA
9	3443	Oakland, CA

User ID	Post ID	Text
3	35314	Not sure why this is getting downvoted.
3	48002	Hehe, of course, it's all true!
5	44921	Please see my post below.
5	44920	Thank you very much for your reply.
8	48675	HTML is not a subset of XML!

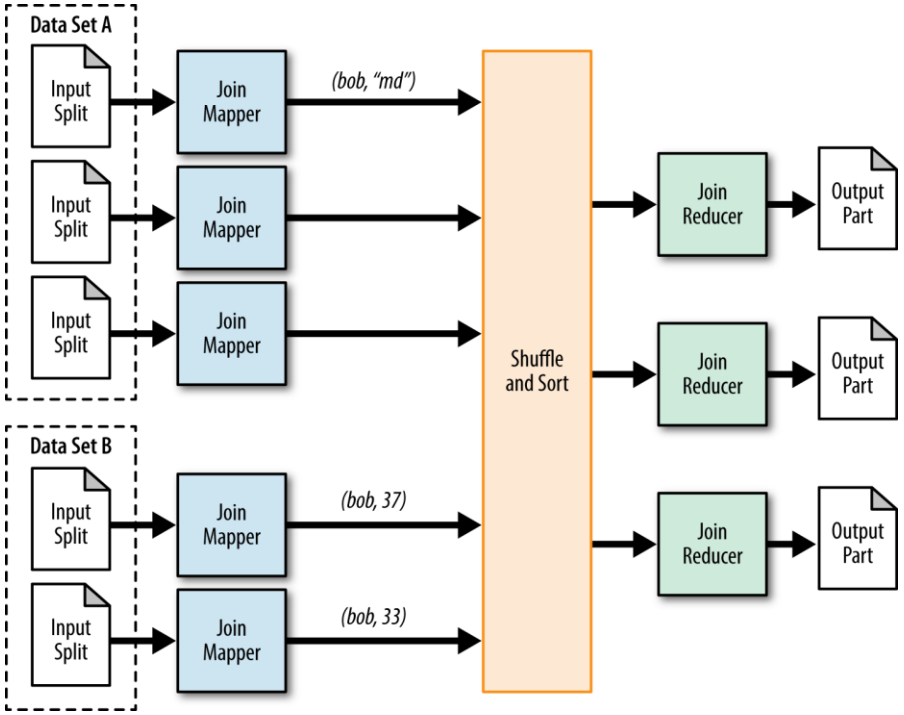
A.User ID	A.Reputation	A.Location	B.User ID	B.Post ID	B.Text
3	3738	New York, NY	3	35314	Not sure why this is getting downvoted.
3	3738	New York, NY	3	48002	Hehe, of course, it's all true!
3	3738	New York, NY	5	44921	Please see my post below.
3	3738	New York, NY	5	44920	Thank you very much for your reply.
3	3738	New York, NY	8	48675	HTML is not a subset of XML!
4	12946	New York, NY	3	35314	Not sure why this is getting downvoted.
4	12946	New York, NY	3	48002	Hehe, of course, it's all true!
4	12946	New York, NY	5	44921	Please see my post below.
4	12946	New York, NY	5	44920	Thank you very much for your reply.
4	12946	New York, NY	8	48675	HTML is not a subset of XML!
5	17556	San Diego, CA	3	35314	Not sure why this is getting downvoted.
5	17556	San Diego, CA	3	48002	Hehe, of course, it's all true!
5	17556	San Diego, CA	5	44921	Please see my post below.
5	17556	San Diego, CA	5	44920	Thank you very much for your reply.
5	17556	San Diego, CA	8	48675	HTML is not a subset of XML!
9	3443	Oakland, CA	3	35314	Not sure why this is getting downvoted.
9	3443	Oakland, CA	3	48002	Hehe, of course, it's all true!
9	3443	Oakland, CA	5	44921	Please see my post below.
9	3443	Oakland, CA	5	44920	Thank you very much for your reply.
9	3443	Oakland, CA	8	48675	HTML is not a subset of XML!

Multiple InputFormat




```
public static void main(String[] args) throws Exception {
Configuration conf = getConf();
Job job = Job.getInstance(conf, "MultipleInputExemple");
job.setJarByClass(PostCommentBuildingDriver.class);
MultipleInputs.addInputPath(job,
                new Path(args[0]), InputFormat1.class, MapperA.class);
MultipleInputs.addInputPath(job,
                new Path(args[1]), InputFormat2.class, MapperB.class);
MultipleInputs.addInputPath(job,
                new Path(args[N]), InputFormatN.class, MapperN.class);
job.setReducerClass(Reducer.class)
job.setOutputFormatClass(OutputFormat.class);
TextOutputFormat.setOutputPath(job, new Path(args[N+1]));
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);
System.exit(job.waitForCompletion(true) ? 0 : 2);
}
```

Pattern d'organisation: Reduce side Join



Le principe du Reduce side join est de se servir d'un MultipleInput pour lire simultanément plusieurs fichiers et utiliser le mécanisme de distribution de message pour regrouper les entrée qui partage une même clé. La jointure est ensuite faite complètement dans le reducer. Toutes les données sont donc transféré sur les reducer.

Pattern d'organisation: Reduce side Join

```
public static class MapperA extends Mapper<Object, Text, Text, Text> {
private Text outkey = new Text();
private Text outvalue = new Text();
public void map(Object key, Text value, Context context)vthrows
IOException, InterruptedException {
outkey.set(GETKEY(value));
outvalue.set("A" + GETVALUE(value));
context.write(outkey, outvalue);
}
}
public static class MapperB extends Mapper<Object, Text, Text, Text> {
private Text outkey = new Text();
private Text outvalue = new Text();
public void map(Object key, Text value, Context context)vthrows
IOException, InterruptedException {
outkey.set(GETKEY(value));
outvalue.set("B" + GETVALUE(value));
context.write(outkey, outvalue);
}
}
```

Pattern d'organisation: Reduce side Join

```
if (joinType.equalsIgnoreCase("inner")) {
// If both lists are not empty, join A with B
if (!listA.isEmpty() && !listB.isEmpty()) {
for (Text A : listA) {
 for (Text B : listB) {
   context.write(A, B);
}
```

Pattern d'organisation: Reduce side Join

```
if (joinType.equalsIgnoreCase("leftouter")) {
// For each entry in A,
for (Text A : listA) {
// If list B is not empty, join A and B
if (!listB.isEmpty()) {
for (Text B : listB) {
context.write(A, B);
}
} else {
// Else, output A by itself
context.write(A, EMPTY_TEXT);
}
```