分散処理アプリ演習　第15回
HBase演習

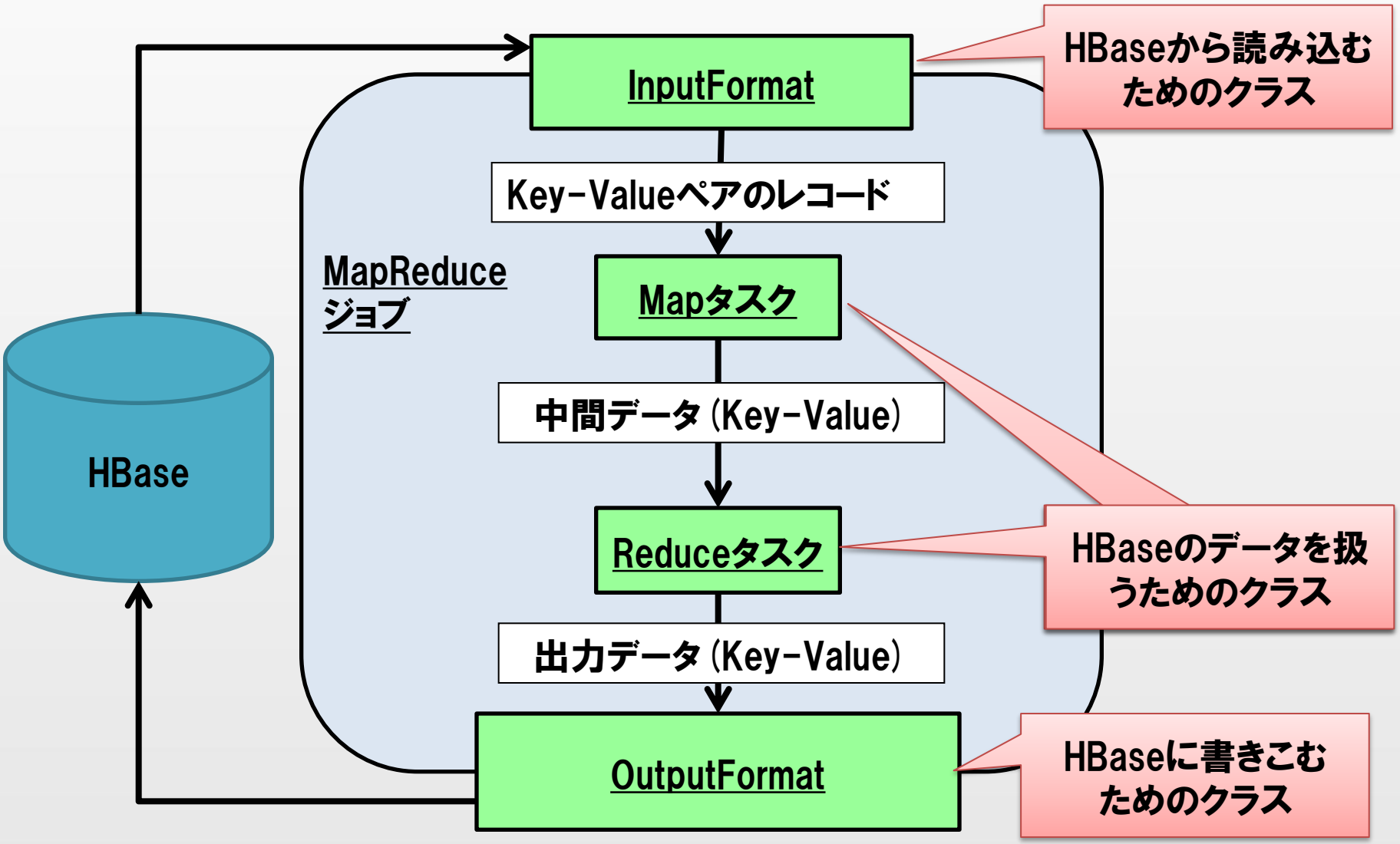
（株）NTTデータ

講義内容

1. HBase での MapReduce
   - HBase用のクラスの紹介、【演習】ツイートログのHashTagをカウントするアプリ演習
2. レポート課題
   - tweetログの活用
3. 分散処理アプリ演習全体のまとめ

1. HBase での MapRedece

MapReduceジョブの構成要素



InputFormat/Mapper クラス

- TableInputFomat
  - HBaseのテーブルからデータを取り出し、Key-Valueの形式で出力する。
  - 入力型：HBaseから取得
  - 出力型：<ImmutableBytesWritable,Result>
    - キー：行キー（Byte配列）が格納される
    - 値：スキャン結果が格納される
  - TableInputFomatは、Resultクラスを出力するため、mapper内でHTableクラスを作成し、HBaseにアクセスする必要はない。
- TableMapper
  - HBaseのテーブルを入力とするMapperクラス。
  - 入力型：<ImmutableBytesWritable,Result>
  - 出力型：<任意,任意>

Reducer/OutputFomat クラス

- TableReducer
  - HBaseのテーブルに出力するReducerクラス。
  - 入力型:<任意,任意>
  - 出力型:<任意,Writable>

- TableOutputFomat
  - HBaseのテーブルに出力する。
  - 入力型:<任意,Writable>
    - キー:使用しないため、任意。ソート不要なので、NullWritableを指定するのが良い。
    - 値:Put もしくは Deleteのみ
  - 出力型:HBaseに出力

Util クラス

- TableMapReduceUtilクラス
  - MapReduceジョブからHBaseへアクセスするための、Jobクラスの設定には、TableMapReduceUtilを利用する。

メソッド		内容
static void	initTableMapperJob ()	MapでTableInputFomatを利用するための設定を行う
static void	initTableReducerJob ()	ReduceでTableOutputFormatを使用する為の設定を行う
static void	limitNumReduceTasks ()	Reducerの個数の上限を設定する
static void	setNumReduceTasks ()	Reducerの個数をテーブルのリージョン数に設定する

Jobクラス

- 入力(HBaseのテーブル)、出力(HBaseのテーブル)とした場合のJobクラスの例

```
Configuration conf = new Configuration();
Job job = new Job(conf, "sampleJob");
job.setJarByClass(SampleJob.class);

// 入力データ条件指定
Scan s = new Scan();

TableMapReduceUtil.initTableMapperJob(
                "<inputTableName>",          // 入力テーブル名
                s,                           // スキャン条件
                SampleTableMapper.class,     // Mapperクラス
                Text.class,                  // Map出力(Key)
                Text.class,                  // Map出力(Value)
                job);

TableMapReduceUtil.initTableReducerJob(
                "<outputTableName>",         // 出力テーブル名
                SampleTableReducer.class,    // Reducerクラス
                job);
```

Mapper例

- TableMapper
- 使用例

Map出力Key（=Reduce入力Key）

Map出力value（=Reduce入力value）

Map入力Key（=InputFormat出力Key）

Map入力value（=InputFormat出力value）

```
static  class SampleTableMapper extends TableMapper<Text, Text> {
  private Text key = new Text();
  private Text value = new Text();

  @Override
  protected void map(ImmutableBytesWritable row, Result result, Context context)
                                          throws IOException, InterruptedException {

    String s_key = Bytes.toString(row.get());
    key.set(s_key);

    KeyValue[] kvList = result.raw();
    for (KeyValue kv : kvList) {
      value.set(kv.getValue());
      context.write(key,value);
    }
  }
}
```

Reducer例

- TableReducer

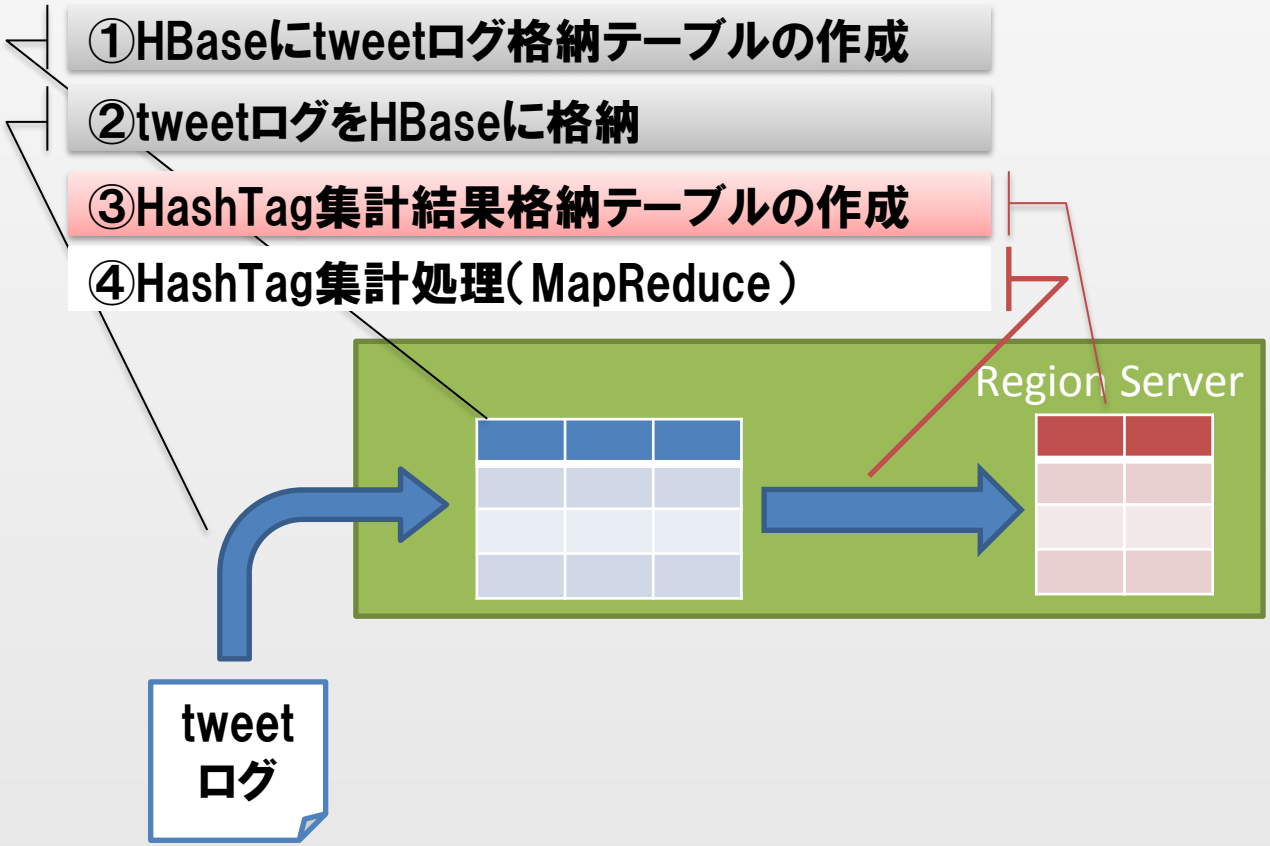
- 使用例

Reduce入力Key　Reduce入力value　Reduce出力Key

```
static class SampleTableReducer extends TableReducer<Text, Text, NullWritable> {

  @Override
  protected void reduce(Text key, Iterable<Text> values, Context context)
                                        throws IOException, InterruptedException {

    Put p = new Put(key.getBytes());

    for ( Text value : values ) {
      p.add(Bytes.toBytes("twitter"),
            HConstants.EMPTY_BYTE_ARRAY,
            value.getBytes());
    }
    context.write(NullWritable.get(), p);
  }
}
```

ツイート ログを使ったアプリ演習

- 演習内容
  - tweetログを解析し、HashTagの利用数をカウントする。
    （特につぶやかれているHashTagから傾向を分析する）
- 演習項目



演習③HashTag集計結果格納テーブルの作成

- 演習③ HashTag集計結果格納テーブルの作成
  - ③-1：以下のHashTag集計結果を格納するテーブルを作成する。
    - 格納対象：HashTag、集計結果
    - テーブル名： HashTagCountTable

演習③HashTag集計結果格納テーブルの作成【回答例】

- 回答例

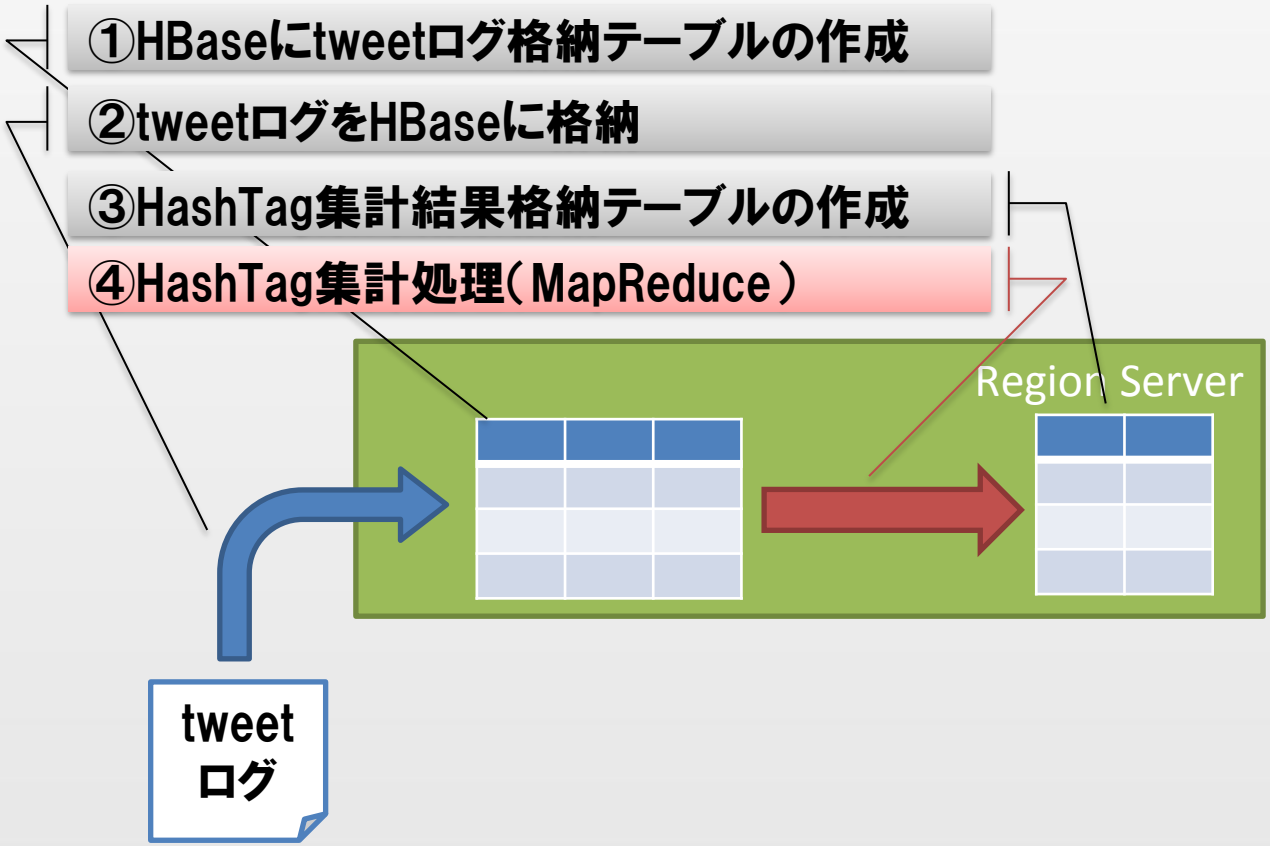
HashTagCountTable

行キー（hashtag）	count

```
hbase(main):001:0> create 'HashTagCountTable', 'count'
```

ツイート ログを使ったアプリ演習

- 演習内容
  - tweetログを解析し、HashTagの利用数をカウントする。
    （特につぶやかれているHashTagから傾向を分析する）
- 演習項目

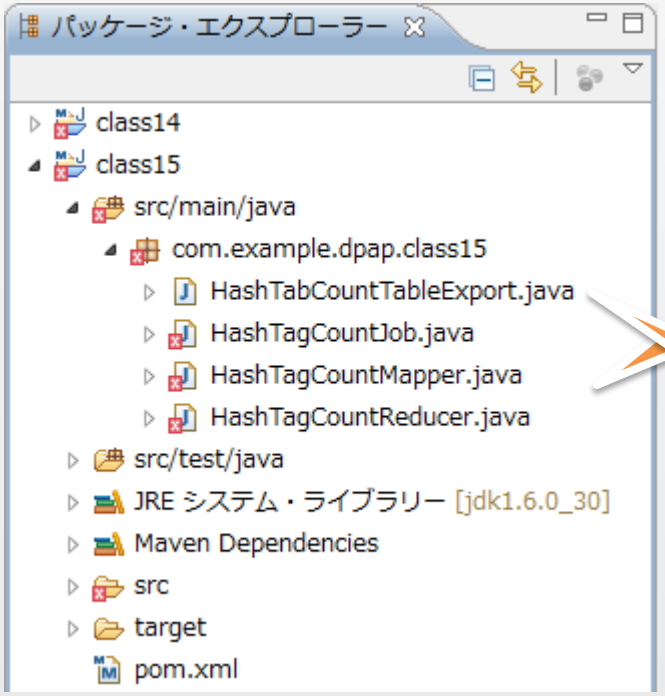


演習④HashTag集計処理(MapReduce)

- 演習④MapRedeceで、HashTagを集計する。
  - ④-1 TwitterTableを入力とし、HashTagを集計した結果を演習③で作成したテーブルに格納するMapReduceジョブを作成する。
  - ④-2 1で作成したジョブを実行し、HashTagの集計処理をする。
  - ④-3 集計結果を確認する。

演習④HashTag集計処理(MapReduce)(実習環境)

- 演習環境
  - ソース格納場所
    - EclipseのClass15フォルダに必要な資材を格納

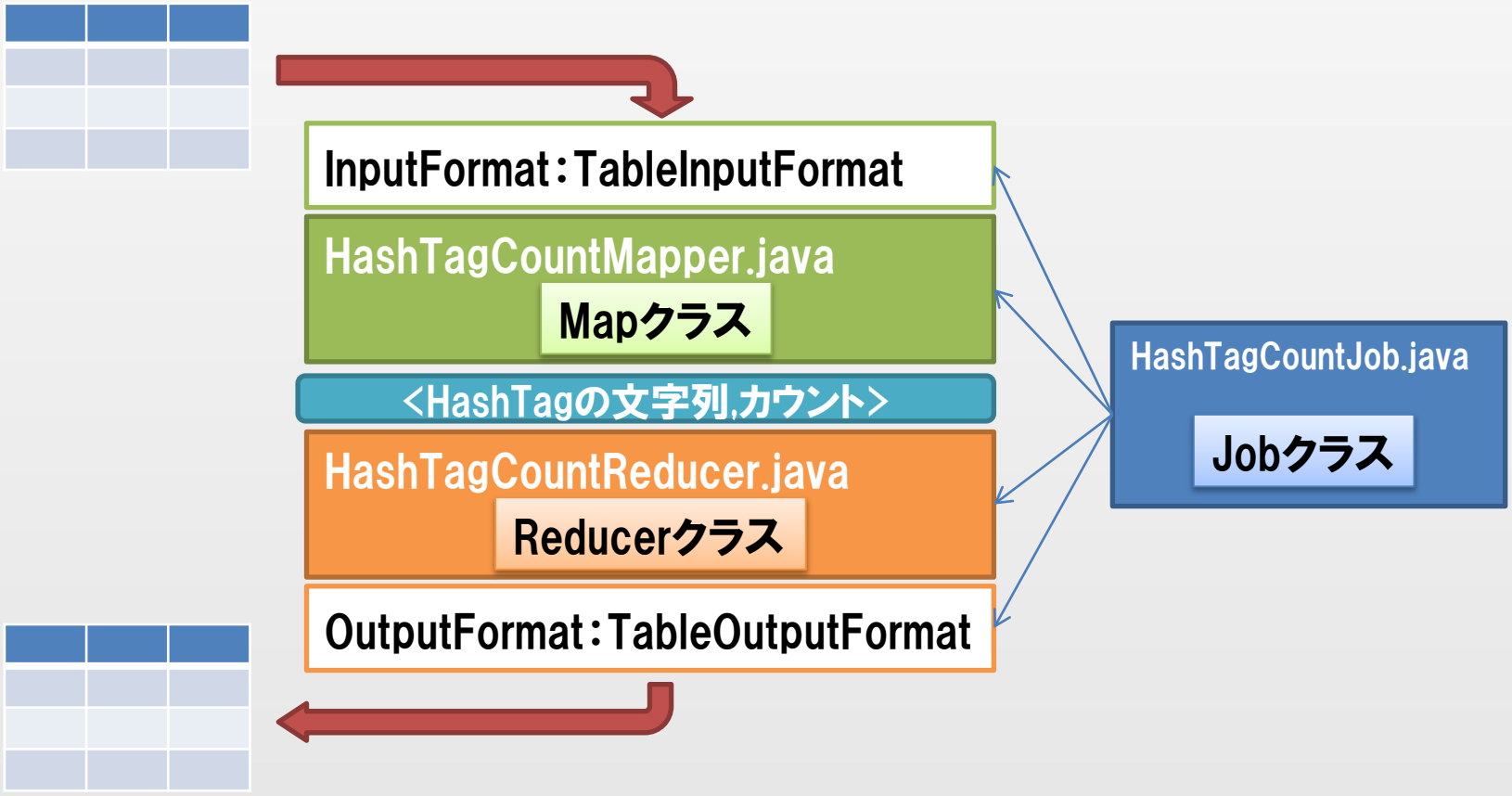


演習で作成するソース
HashTagCountJob.java
HashTagCountMapper.java
HashTagCountReducer.java
(未完成)

HBaseの中身確認用
HashTabCountTableExport.java
(完成済)

演習④HashTag集計処理（MapReduce）（アプリ概要）

- 作成ファイル
  - HashTagCountJob.java（Jobクラス）
  - HashTagCountMapper.java（Mapクラス）
  - HashTagCountReducer.java（Reducerクラス）

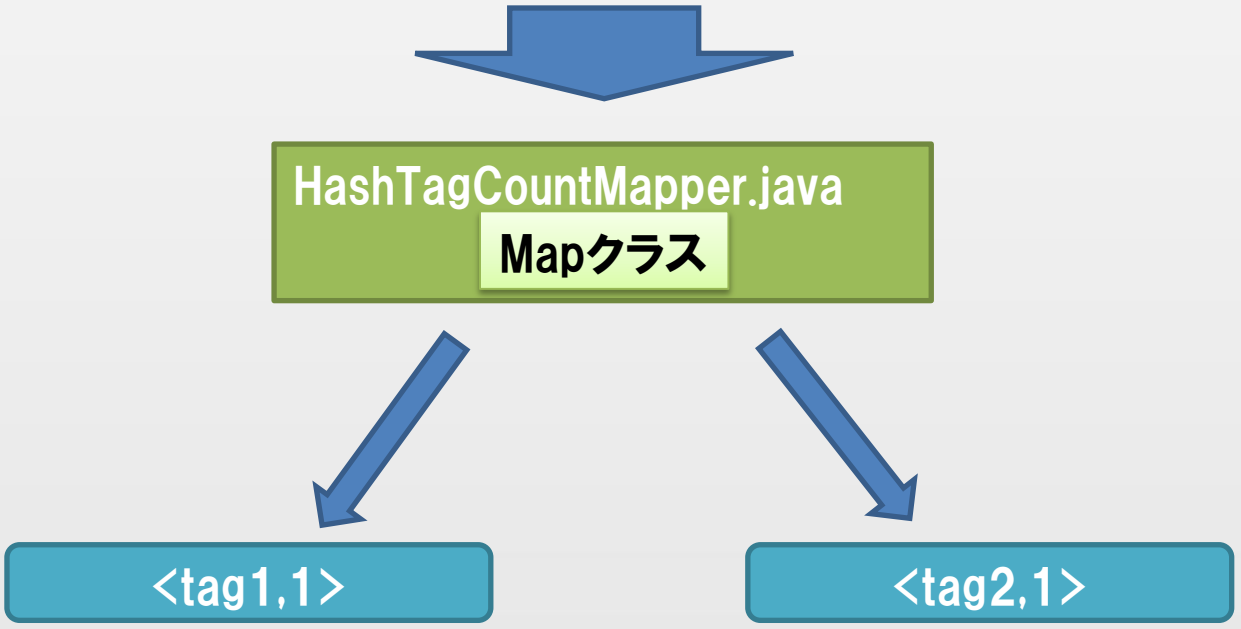


演習④HashTag集計処理(MapReduce)(Map概要)

- HashTag集計概要(Map)

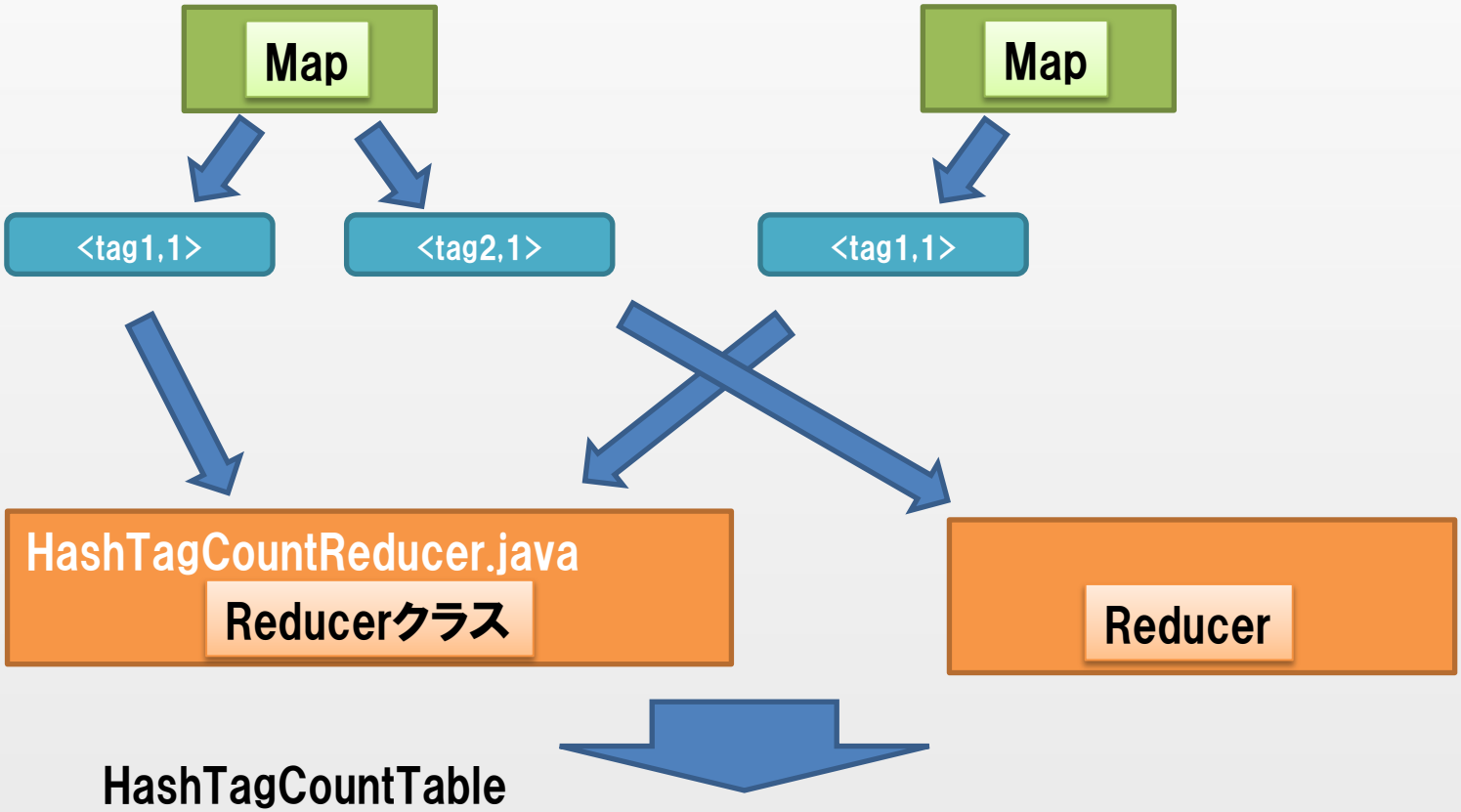
TwitterTable

行キー (UserID+Tweet ID)	tweet		
	text	hashtag1	hashtag2
333160681-133733411750809600	@ReplyUserSan Test Message #tag1 #tag2	tag1	tag2



演習④HashTag集計処理(MapReduce)(Reduce概要)

- HashTag集計概要(Reduce)



HashTagCountTable

	count
tag1	2
tag2	1

実行手順

- 実行方法
  - 事前に"HashTagCountTable"を作成しておく必要がある。

```
$ hadoop jar ~/workspace/Class15/target/hbaseCount-0.1.jar ¥
com.example.dpap.class15.HashTagCountJob
```

- 【補足】HashTagCountTableに格納されたデータを標準出力に表示する
  - HashTabCountTableExport.java

表示させる閾値
（この数字以上の回数の
hashtagのみ表示）

```
$ hadoop jar ~/workspace/Class15/target/hbaseCount-0.1.jar ¥
com.example.dpap.class15.HashTabCountTableExport 10 2> /dev/null
```

不要なメッセージは、
/dev/nullに捨てる

  - Linuxのコマンドを利用して、hashtagの利用数順にソート

```
$ hadoop jar ~/workspace/Class15/target/hbaseCount-0.1.jar ¥
com.example.dpap.class15.HashTabCountTableExport 10 2> /dev/null ¥
| sort -t "    " -k3 -rn
```

出力結果をsortコマンドでソートする

区切り文字はタブ
※ctrl+vを押したあと、tabを入力

回答解説

- 別紙で解説

まとめ

本講義で学んだ内容

- HBase演習
  - MapReduceでHBaseにアクセスする方法
  - tweetログ解析

2. レポート課題

レポート課題：tweetログの活用

- 以下の問題から1つ選び、回答してください。
  - ① ツイートされた時間で対象となるtweetをしぼり、hashtagをカウントする。
  - ② 「TwitterTable」にツイートしたユーザのフォロアー数（followers_count）を格納し、フォロアー数に応じてHashTagのカウントを重み付けする。
    （例えば、フォロアーが1～10人なら×1、20～50人なら×2、50～人なら×3）
  - ③ 「TwitterTable」にツイートしたユーザのロケーション（location）を格納し、ロケーション毎にHashTagをカウントする。
- 提出物
  - ソースコード（適宜コメントを記載すること），jarファイル，実行結果（上位のみで良い）、解答の方針/プログラムの説明
  - 未完成であっても部分点を出すので提出すること
- 提出方法：上記のファイル群をtar.gzまたはzipで1つにまとめてLMSに提出
- 提出期限：当日周知
- 質問先：当日周知

3. 分散処理アプリ演習全体のまとめ

講義計画 1日目

- 第1回：Hadoopの概要
- 第2回：MapReduceアプリケーションの概要
  - 文献単語解析アプリを題材として、Hadoop（HDFS、MapReduce）の基礎について解説し、演習を行う。
- 第3回：MapReduceプログラミング基礎
- 第4回：MapReduceによるレコメンデーションエンジンの実装
  - レコメンデーションアプリを題材として、MapReduceアプリケーションの代表的な適用領域の一つである集計・統計処理について説明するとともに、MapReduceプログラミングの基礎および実践的な実装テクニックについて解説し、演習を行う。
  - まず、MapReduceアプリケーション実装の基本として、必要なクラスや設定等を説明する。次に、実践的な実装テクニックとして、MapとReduceの使い分け、ジョブの分割指針等を解説する。さらに、代表的なMapReduceの適用領域として、集計・統計処理の例であるレコメンデーションについて取り上げ、レコメンデーションアプリを実装する演習を行う。

講義計画 2日目

- 第5回：Hadoop動作詳細
- 第6回：MapReduceプログラミング応用
- 第7回：MapReduceアプリケーションのテスト
- 第8回：MapReduceアプリケーションのチューニング
  - POSデータ分析アプリを題材として、Hadoopの動作詳細、高度なMapReduceプログラミング、テスト方法、性能チューニング方法、について解説し、演習を行う。
  - まず、Hadoopの構成要素であるHDFSとMapReduceについて詳細な挙動を説明する。Hadoopフレームワークとしてのデータの管理方法や分散処理の仕組みについて第1回-第2回で説明した内容を掘り下げて解説する。次に、POSデータを集計するためのアプリケーションをJavaでのMapReduceプログラミングにより実装する。この中で、HadoopのMapReduceフレームワークが提供する各種機能を利用したテクニックについて解説する。そして実装したアプリケーションは、テストやデバッグを経て、分散環境で動作させる。このとき性能に関する観点やチューニングポイントについて説明する。

講義計画 3日目

- 第9回：Hadoopクラスタの運用
  - Hadoopの運用・監視方法について解説する。
  - アプリケーションの動作状況を把握するためにHadoopの持つ統計情報をGangliaにて確認する。
- 第10回：Hive概要
- 第11回：Hive演習
- 第12回：Pig概要・演習
  - POSデータ分析アプリを題材として、HiveやPigによるアプリ開発方法について解説し、演習を行う。
  - SQLライクなクエリ言語をサポートするMapReduceのインターフェイス「Hive」について解説する。MapReduceとの関係やRDBMSとの違いを解説したのち、POSシステムを題材とした演習を行う。さらにHiveとの比較としてPigについても解説・演習を行う。

講義計画 4日目

- 第13回：HBase概要
- 第14回：HBaseスキーマ設計
- 第15回：HBase演習
  - twitterログ解析アプリを題材として、HBaseを利用したアプリ開発方法について解説し、演習を行う。
  - まず、HBaseの概要として、Key-Valueストア、RDBMSやHDFSとの比較、HBaseの採用基準・適用領域等について説明し、次に、HBaseの機能やアーキテクチャを解説する。また、HBaseのスキーマ設計のポイントについて説明する。さらに、HBaseを用いたアプリを実装する演習を行う。