Main(String args[]){

Job job =;

Job.setMapperClass();

Job.setCombinerClass();

Job.setReducerClass();

Job.setNumReduceTasks();

Job.setInputFormatClass();

Job.setOutputFormatClass();

Job.setInputPath(args[0]);

Job.setOutputPath(args[1])

}

Public class MyCombiner extends Reducer{

Public void reduce(KEYIN key, Iterable<VALUEIN> values, Context context){

}

}

Custom writable class

Class Stock implements Writable{

String symbol:

Float opening;

Float closing;

}

Every value used in MapReduce needs to implement the writable interface

Classes that implement Writable Interface:

* IntWritable
* ArrayWritable
* …..

Override write and readFields

Every Key used in MapReduce needs to implement the writableComparable interface

Classes that implement Writable Interface:

* BooleanWritable
* ByteWritable

Class CustomPartitioner extends Partitioner{

String k = key.toString();

If(k.substring(4).equals(k1))

Return 0;

…….

}

Class MyPartitioner extends Partitioner{

int getPartition(KEY key, VALUE value, int numPartitions){

if(key == K1){

return 0;

}else if(key == K2 && 0<val <4K){

return 0;

}else if(key == K2 && 5<val <10<){

return 1;

}

//return hash(key) % numPartitions;

}

}

Prog in slide

1 comment1 timestamp

1 comment2 timestamp

1 comment3 timestamp

map(){

emit(userid, “firstdate:lastdate:1”)

}

public class MinMaxCountTuple implements Writable {

Date minDate;

Date maxDate;

int date;

}

**Distinct Pattern:**

map(){

emit(key,NullWritable.get())

}

reduce(){

emit(key,NullWritable.get())

}