<http://blog.jobbole.com/84089/>

# Hadoop WordCount实例

package org.apache.hadoop.examples;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

public static class TokenizerMapper

extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new **IntWritable**(1);

private Text word = new **Text**();

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

StringTokenizer itr = new **StringTokenizer**(value.**toString**());

while (itr.**hasMoreTokens**()) {

word.**set**(itr.**nextToken**());

context.**write**(word, one);

}

}

}

public static class IntSumReducer

extends Reducer<Text,IntWritable,Text,IntWritable> {

private IntWritable result = new **IntWritable**();

public void reduce(Text key, Iterable<IntWritable> values,

Context context

) throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.**get**();

}

result.**set**(sum);

context.**write**(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new **Configuration**();

String[] otherArgs = new **GenericOptionsParser**(conf, args).**getRemainingArgs**();

if (otherArgs.length != 2) {

System.err.**println**("Usage: wordcount <in> <out>");

System.**exit**(2);

}

Job job = new **Job**(conf, "word count");

job.**setJarByClass**(WordCount.class);

job.**setMapperClass**(TokenizerMapper.class);

job.**setCombinerClass**(IntSumReducer.class);

job.**setReducerClass**(IntSumReducer.class);

job.**setOutputKeyClass**(Text.class);

job.**setOutputValueClass**(IntWritable.class);

FileInputFormat.**addInputPath**(job, new **Path**(otherArgs[0]));

FileOutputFormat.**setOutputPath**(job, new **Path**(otherArgs[1]));

System.**exit**(job.**waitForCompletion**(true) ? 0 : 1);

}

}

map函数的定义如下：

public void map(Object key, Text value, Context context)

Object key和Text value就是输入的key和value了，Context context是可以记录输入的key和value，比如context.write(word, one)

context也会记录map运行的状态

reduce函数定义如下：

public void reduce(Text key, Iterable<IntWritable> values, Context context)

values是迭代器类型的，也就是说reduce输入的key对应的是一组值的value，reduce中的context和map中的context作用一样。

在main函数中，首先

Configuration conf = new **Configuration**();

mapreduce程序运行前要初始化Configuration，这个类主要是读取mapreduce系统配置信息，信息包括HDFS和mapreduce，也就是读取hadoop的配置文件（core-site.xml、hdfs-site.xml、mapred-site.xml等）。