hadoop

分布式计算引擎。
common
hdfs　　//分布式存储
mapreduce　　//编程模型(分布式)
yarn

并发和并行

并行格局大，针对集群进行分布式计算。
并发格局小，针对单个节点的应对并发请求能力。

spark与hadoop区别内存计算

1.
2.
3.

Spark

快如闪电集群计算引擎。
应用于大规模数据处理快速通用引擎。
内存计算。

[Speed]
计算速度是hadoop的100x.
Spark有高级DAG(Direct acycle graph,有向无环图)执行引擎。

[易于使用]
使用java,scala,python,R,SQL编写App。
提供了80+高级算子，能够轻松构建并行应用。
也可以使用scala，python，r的shell进行交互式操作

[通用性]
对SQL，流计算，复杂分析进行组合应用。
spark提供了类库栈，包括SQL，MLlib，graphx，Spark streaming.

[架构]
Spark core
spark SQL
spark streaming
spark mllib
spark graphx

[到处运行]
spark可以运行在hadoop,mesos,standalone,clound.
可以访问多种数据源，hdfs，hbase，hive，Cassandra, S3.

spark集群部署模式

1.local
2.standalone
3.mesos
4.yarn

安装spark[local模式]

1.下载spark-2.1.0-bin-hadoop2.7.tgz
2.解压
3.配置环境变量
[/etc/profile]
...
export SPARK_HOME=/soft
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin

4.source
source /etc/profile

5.进入spark-shell
$>spark/bin/spark-shell
$scaka>1 + 1

RDD
---

resilient distributed dataset ,弹性分布式数据集。
等价于java中的集合比如list.

实现word count

1.分布实现
```
//1.加载文件
scala>val rdd1 = sc.textFile("/homec/centos/1.txt")

//2.压扁每行
scala>val rdd2 = rdd1.flatMap(_.split(" "))

//3.标1成对
scala>val rdd3 = rdd2.map(w=>(w,1))

//4.按照key聚合每个key下的所有值
scala>val rdd4 = rdd3.reduceByKey(_+_)

//5.显式数据
scala>rdd4.collect()
```

2.一步实现

```
$scala>sc.textFile("file:///home/centos/1.txt").flatMap(_.split("
")).map((_,1)).reduceByKey(_+_).collect
```

3.气温值最大值聚合(分布完成)
```
//1.加载文件
scala>val rdd1 = sc.textFile("/home/centos/temp.dat")
```

```
//2.加载文件
scala>val rdd2 = rdd1.map(line=>{
        val arr = line.split(" ") ;
        (arr(0).toInt,arr(1).toInt)
        })

//3.按key聚合取出最大值
scala>val rdd3 = rdd2.reduceByKey((a,b)=> if(a >b) a else b)

//4.按年排序
scala>val rdd4 = rdd3.sortByKey()

//5.显式
scala>rdd4.collect()
```

idea下编写spark程序

1.创建java项目，选择scala类库
2.添加maven支持，引入依赖

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.oldboy</groupId>
    <artifactId>myspark</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-core_2.11</artifactId>
            <version>2.1.0</version>
        </dependency>
    </dependencies>
</project>
```

3.编程

```
[scala版]
import org.apache.spark.{SparkConf, SparkContext}

/**
  * Created by Administrator on 2018/5/8.
  */
object WCAppScala {
    def main(args: Array[String]): Unit = {
        //1.创建spark配置对象
        val conf = new SparkConf()
        conf.setAppName("wcApp")
        conf.setMaster("local")
```

```
                        spark-day01.笔记.txt
                        //2.创建spark上下文件对象
                        val sc = new SparkContext(conf)

                        //3.加载文件
                        val rdd1 = sc.textFile("d:/mr/1.txt")

                        //4.压扁
                        val rdd2 = rdd1.flatMap(_.split(" "))

                        //5.标1成对
                        val rdd3 = rdd2.map(w => (w,1))

                        //6.化简
                        val rdd4 = rdd3.reduceByKey(_ + _)

                        //收集数据
                        val arr = rdd4.collect()

                        arr.foreach(println)

                        //
                }
}

[java版]
package com.oldboy.spark.java;

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.FlatMapFunction;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFunction;
import scala.Tuple2;

import java.util.Arrays;
import java.util.Iterator;
import java.util.List;

/**
 *
 */
public class WCAppJava {
        public static void main(String[] args) {
                //1.创建配置对象
                SparkConf conf = new SparkConf() ;
                conf.setAppName("wcApp") ;
                conf.setMaster("local") ;

                //2.创建java版的上下文
                JavaSparkContext sc = new JavaSparkContext(conf)
;

                //3.加载文件
                JavaRDD<String> rdd1 =
```

```
sc.textFile("d:/mr/1.txt");

                                //4.压扁
                                JavaRDD<String> rdd2 = rdd1.flatMap(new
FlatMapFunction<String, String>() {
                                        public Iterator<String> call(String s)
throws Exception {
                                                String[] arr = s.split(" ");
                                                return
Arrays.asList(arr).iterator();
                                        }
                                }) ;

                                //5.标一成对
                                JavaPairRDD<String,Integer> rdd3 =
rdd2.mapToPair(new PairFunction<String, String, Integer>() {
                                        public Tuple2<String, Integer>
call(String s) throws Exception {
                                                return new Tuple2<String,
Integer>(s , 1);
                                        }
                                }) ;

                                //6.化简
                                JavaPairRDD<String,Integer> rdd4 =
rdd3.reduceByKey(new Function2<Integer, Integer, Integer>() {
                                        public Integer call(Integer v1, Integer
v2) throws Exception {
                                                return v1 + v2;
                                        }
                                }) ;

                                //7.收集
                                List<Tuple2<String,Integer>> list =
rdd4.collect();
                                for(Tuple2<String,Integer> t : list){
                                        System.out.println(t._1() + " : " +
t._2);
                                }
                        }
                }
```

练习

1.最高气温，最低气温一次聚合得出
2.最高气温，最低气温、平均气温一次聚合得出

```
package com.oldboy.spark.java;

import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFunction;
```

```
import scala.Tuple2;
import scala.Tuple4;

import java.util.List;

/**
 * 统计气温数据
 */
public class TempAggJava {
        public static void main(String[] args) {
                SparkConf conf = new SparkConf();
                conf.setAppName("tempAggJava");
                conf.setMaster("local") ;

                JavaSparkContext sc = new
JavaSparkContext(conf);

                //1.加载文件
                JavaRDD<String> rdd1 =
sc.textFile("d:/mr/temp.dat");

                //2.变换
                JavaPairRDD<Integer, Tuple4<Integer, Integer,
Double, Integer>> rdd2 = rdd1.mapToPair(new PairFunction<String, Integer,
Tuple4<Integer,Integer,Double,Integer>>() {
                        public Tuple2<Integer, Tuple4<Integer,
Integer, Double, Integer>> call(String s) throws Exception {
                                String[] arr = s.split(" ");
                                int year =
Integer.parseInt(arr[0]) ;
                                int temp =
Integer.parseInt(arr[1]) ;
                                return new Tuple2<Integer,
Tuple4<Integer, Integer, Double, Integer>>(year, new
Tuple4<Integer,Integer,Double,Integer>(temp , temp , new Double(temp) , 1)) ;
                        }
                }) ;

                //3.聚合
                JavaPairRDD<Integer, Tuple4<Integer, Integer,
Double, Integer>> rdd3 = rdd2.reduceByKey(
                                new Function2<Tuple4<Integer,
Integer, Double, Integer>, Tuple4<Integer, Integer, Double, Integer>,
Tuple4<Integer, Integer, Double, Integer>>() {
                                        public Tuple4<Integer,
Integer, Double, Integer> call(Tuple4<Integer, Integer, Double, Integer> v1,
Tuple4<Integer, Integer, Double, Integer> v2) throws Exception {
                                                int max =
Math.max(v1._1(),v2._1()) ;
                                                int min =
Math.min(v1._2(),v2._2()) ;
                                                int count =
v1._4() + v2._4() ;
                                                //计算平均值
                                                double avg =
(v1._3() * v1._4() + v2._3() * v2._4()) / count ;
```

```
                                                        return new
Tuple4<Integer, Integer, Double, Integer>(max, min, avg, count) ;
                                                    }
                                                }) ;

                                //收集
                                List<Tuple2<Integer, Tuple4<Integer, Integer,
Double, Integer>>> list = rdd3.collect();
                                for(Tuple2<Integer, Tuple4<Integer, Integer,
Double, Integer>> t : list){
                                        System.out.println(t);
                                }
                        }
                }
```

3.

查看job webui
---
http://192.168.231.101:4040

RDD
---
resilient distributed dataset,
弹性分布式数据集。
类似于java中集合.

idea下实现spark编程

1.常见模块
2.添加maven

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
                        <modelVersion>4.0.0</modelVersion>

                        <groupId>com.it18zhang</groupId>
                        <artifactId>my-spark</artifactId>
                        <version>1.0-SNAPSHOT</version>

                        <dependencies>
                                <dependency>
                                        <groupId>org.apache.spark</groupId>
                                        <artifactId>spark-core_2.11</artifactId>
                                        <version>2.1.0</version>
                                </dependency>
                        </dependencies>
                </project>
```

3.编程

```
import org.apache.spark.{SparkConf, SparkContext}

/**
  */
object WordCountScala {

        def main(args: Array[String]): Unit = {
                //常见spark配置对象
                val conf = new SparkConf()
                conf.setAppName("wcScala")
                conf.setMaster("local")

                //创建spark上下文
                val sc = new SparkContext(conf)

                //加载文件
                val rdd1 = sc.textFile("file:///d:/1.txt")
                //压扁
                val rdd2 = rdd1.flatMap(_.split(" "))
                //标1成对(word,1)
                val rdd3 = rdd2.map(e=>(e,1))
                //按key聚合
                val rdd4 = rdd3.reduceByKey(_+_)
                val arr = rdd4.collect()
                for(e <- arr){
                        println(e)
                }
        }
}
```

java版实现wc

```
import org.apache.spark.SparkConf;
import org.apache.spark.SparkContext;
import org.apache.spark.api.java.JavaPairRDD;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.api.java.function.FlatMapFunction;
import org.apache.spark.api.java.function.Function2;
import org.apache.spark.api.java.function.PairFunction;
import org.apache.spark.rdd.RDD;
import scala.Function1;
import scala.Tuple2;

import java.util.Arrays;
import java.util.Iterator;
import java.util.List;

/**
 * Created by Administrator on 2018/2/27.
 */
public class WordCountJava {
        public static void main(String[] args) {
                SparkConf conf  = new SparkConf();
                conf.setAppName("wcJava") ;
```

```
                conf.setMaster("local");

                //创建spark上下文
                JavaSparkContext sc = new JavaSparkContext(conf);

                //加载文件
                JavaRDD<String> rdd1 = sc.textFile("file:///d:/1.txt");

                //压扁
                JavaRDD<String> rdd2 = rdd1.flatMap(new
FlatMapFunction<String, String>() {
                        public Iterator<String> call(String s) throws
Exception {
                                String[] arr = s.split(" ");
                                return Arrays.asList(arr).iterator();
                        }
                }) ;

                //标1成对
                JavaPairRDD<String,Integer> rdd3 = rdd2.mapToPair(new
PairFunction<String, String, Integer>() {
                        public Tuple2<String, Integer> call(String s)
throws Exception {
                                return new Tuple2<String, Integer>(s,1);
                        }
                }) ;
                //聚合计算
                JavaPairRDD<String,Integer> rdd4 = rdd3.reduceByKey(new
Function2<Integer, Integer, Integer>() {
                        public Integer call(Integer v1, Integer v2)
throws Exception {
                                return v1 + v2;
                        }
                }) ;
                //
                List<Tuple2<String,Integer>> list = rdd4.collect();
                for (Tuple2<String,Integer> t : list) {
                        System.out.println(t._1 + " : " + t._2());
                }
        }
}
```

搭建spark集群

1.部署模式

1.local

没有任何spark进程，使用spark-shell交互终端，使用spark的api运行在jvm中。

调试测试该方式。

2.standalone

独立模式。

需要启动spark相应的进程，master + worker.

3.yarn

运行hadoop的yarn之上。

4.mesos
-

2.部署spark成standalone
2.1)规划
s101 ~ s104
s101 //master
s102 //worker
s103 //worker
s104 //worker

2.2)分发s101 spark安装目录到所有节点
$>su centos
$>xsync.sh /soft/spark*
$>xsync.sh /soft/spark

$>su root
$>xsync.sh /etc/profile

2.3)在spark的conf目录下创建到hadoop的配置文件的软连接
xcall.sh "ln -s /soft/hadoop/etc/hadoop/hdfs-site.xml /soft/spark/conf/hdfs-site.xml"
xcall.sh "ln -s /soft/hadoop/etc/hadoop/core-site.xml /soft/spark/conf/core-site.xml"

2.4)修改slaves文件
[spark/conf/slaves]
s102
s103
s104

2.4')配置/spark/conf/spark-env.sh并分发
export JAVA_HOME=/soft/jdk

2.5)先启动hadoop的hdfs
2.5.1)启动zk
[s101]
$>xzk.sh start

2.5.2)启动hdfs
[s101]
start-dfs.sh
2.6)启动spark集群
$>spark/sbin/start-all.sh

2.7)验证webui
http://s101:8080

启动spark-shell，连接到spark集群,实现wordcount

$>spark-shell --master spark://s101:7077

```
$scala>sc.textFile("hdfs://mycluster/user/centos/1.txt").flatMap(_.split(" ")).map((_,1)).reduceByKey(_+_).collect
```

使用nc方式，将各节点运行的信息发送到s101进行输出查看

1.在spark-shell中定义函数，发送消息给远程服务器
```
def sendInfo(str:String) = {
        val localIp =
java.net.InetAddress.getLocalHost().getHostAddress()
        val socket = new java.net.Socket("192.168.231.101" ,
8888) ;
        val out = socket.getOutputStream()
        out.write((localIp + " ==> " + str + "\r\n").getBytes())
        out.flush()
        socket.close()
}
```
2.在s101启动nc服务器
```
nc -lk 8888
```
3.编写程序
```
val rdd1 = sc.textFile("hdfs://mycluster/user/centos/1.txt")
val rdd2 = rdd1.flatMap(line=>{
        sendInfo(" flatMap() : " + line)
        line.split(" ")
})
val rdd3 = rdd2.map(word=>{
        sendInfo(" map() : " + word)
        (word , 1)
})
val rdd4 = rdd3.reduceByKey((a,b)=>{
        sendInfo(" reduceByKey() : " + a + " & " + b)
        a + b
})
rdd4.collect()
```

导出程序jar包，丢到spark集群上运行

1.修改master地址
```
conf.setMaster("spark://s101:7077")
...
```
2.导出jar包
略
3.传递jar到centos

4.执行一下命令，实现程序在spark集群上运行
```
spark-submit --master spark://s101:7077 --class WordCountScala
my-spark.jar
spark-submit --master spark://s101:7077 --class WordCountJava
my-spark.jar
```

在spark中处理数据倾斜

1.以local方式启动spark-shell
```
$>spark-shell --master local[4]
```
2.wordcount

```
$>sc.textFile("file:///home/centos/1.txt").flatMap(_.split(" ")).map(e=>(e + "_" + scala.util.Random.nextInt(10),1)).reduceByKey(_+_).map(t=>(t._1.substring(0,t._1.lastIndexOf("_")),t._2)).reduceByKey(_+_).collect
```

部署spark程序在集群运行

1.修改程序代码，从hdfs加载文件。

```
conf.setMaster("spark://s101:7077") ;
...
sc.textFile("hdfs://mycluster/user/centos/1.txt");
```

2.导出程序，生成jar包。

```
project structure ->artifact -> + -> jar -> 删除自带jar包
```

3.build -> artifacts -> myspark

4.定位到到处目录，复制jar到centos

```
D:\big10\out\artifacts\myspark_jar
```

5.在centos上执行spark-submit命令运行程序

```
[scala版]
spark-submit --master spark://s101:7077 --class WCAppScala myspark.jar
[java版]
spark-submit --master spark://s101:7077 --class com.oldboy.spark.java.WCAppJava myspark.jar
```

spark集群管理

```
[启动]
start-all.sh                                    //启动所有spark进程
start-master.sh                                 //启动master节点
start-slaves.sh                                 //master节点启动所有worker节点
start-slave.sh spark://s101:7077        //单独登录单个worker节点，启动worker进程

[停止]
stop-all.sh                                     //停止所有进程
stop-master.sh                                  //停止master进程
stop-slaves.sh                                  //停止所有worker节点
stop-slave.sh                                   //登录每个worker节点，停止worker进程
```

作业

---

taggen使用spark实现(scala + java)。