Apache Spark

软件测试报告

组员：

SY1506404 孟翰

SY1506409 苏若

SY1506425 李璇

SY1506406 孙敏芳

**版本变更历史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| 1.0 | 2016.05.25 | 孟翰、李璇、孙敏芳、苏若 | 孟翰、李璇、孙敏芳、苏若 | 初始版本V1.0 |

1. 概述

在本文档中，从目的上看，我们依据《测试规格说明书》中的测试用例完成相关的测试工作，以期获得Apache Spark系统与《需求规格说明书》中设定的需求的偏差值，并期按实际情况进行相关问题修复。

从文档内容上，主要包括了以下几方面：

1. 测试过程的策略选择，包括不同测试阶段的人员职责安排及回归测试时测试方法的选择。
2. 测试过程说明，包括各测试阶段的具体执行情况，如输入的数据、输入输出接口、产生的中间输出及测试结果描述。
3. 建立测试结果、软件问题、测试需求项、测试用例的追踪关系。
4. 总结整个测试并附上软件问题报告清单。
5. 测试过程说明
   1. 执行策略

在第一轮测试中，依据测试需求规格说明书中设计的3个针对功能性需求的测试用例和4个针对非功能性需求的测试用例进行人员分工和具体测试的实施，如下表所示：

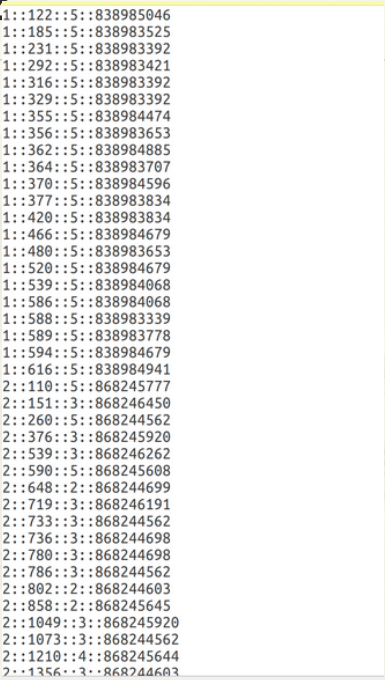
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 测试阶段 | 测试用例标识 | 具体测试活动 | 展开时间 | 执行人 | 监督人 | 备注 |
| 功能。性需求—Spark机器学习 | TC001-1 | 测试选取数据完整性 | 2016.5.18 | 孟翰 | 苏若 | Movielens数据集存在个别数据丢失值的情况，我们采取的的策略是舍去这些噪声点。 |
| 功能性需求—Spark机器学习 | TC001-2 | 数据类型转换与整理测试 | 2016.5.18 | 孟翰 | 苏若 | 采集到的数据默认为String类型且有多余的数据条目需要剔除，测试采用spark的map操作进行提取并检查生成结果。 |
| 功能性需求—Spark机器学习 | TC001-3 | Spark机器学习数据处理测试 | 2016.5.19 | 孟翰 | 苏若 | 利用Spark提供的机器学习中的推荐包对处理好的数据进行处理，对不同的参数进行测试以得到近似最优的准确率 |
| 功能性需求—Spark机器学习 | TC001-4 | 推荐系统准确率测试 | 2016.5.19 | 孟翰 | 苏若 | 选取测试数据对于机器学习生成的推荐系统进行准确率测试。 |
| 功能性需求- Spark执行SQL数据处理测试 | TC002-1  ~TC002-7 | 选择用于测试的数据，并针对SQL特有的增删改查功能进行分别测试 | 2016.5.20 | 孙敏芳 | 李璇 | 选取数据为Json格式数据，为Spark自带数据内容，测试围绕数据内容进行测试并输出 |
| 功能性需求- Spark Stream流计算测试 | TC003-1  、  TC003-2 | 选择用于测试的数据和测试的内容 | 2016.5.21 | 孙敏芳 | 李璇 | 选取数据服务器上的文本数据对过去1min内文本数据进行单词数统计 |
| 功能性需求- Spark Stream流计算测试 | TC003-3  、  TC003-4 | 更改batch interval的大小，测试耗时的差别 | 2016.5.21 | 孙敏芳 | 李璇 | 更改batch interval的大小测试结果中当值为1s的时候耗时最小；并且当时间设置为500ms的时候计算效率相差很大，时间为前者近9倍。 |
| 非功能性需求-鲁棒性 | TC011 | Spark鲁棒性测试 | 2016.5.23 | 孟翰 | 苏若 | 模拟Spark处理大量数据、错误输入、内存容量不足、节点失连等情况，测试其健壮性。 |
| 非功能性需求-容错性 | TC012 | Spark对于用户误操作的容错性测试 | 2016.5.23 | 孟翰 | 苏若 | 模拟用户错误操作，测试Spark能否根据其lineage（血统关系）回溯之前的操作与系统状态。 |
| 非功能性需求-安全性 | TC013 | 测试Spark是否能保证Web UI安全、事件审计安全、网络端口安全等安全性能要求 | 2016.5.23 | 孙敏芳 | 李璇 | 模拟对Spark Web　UI、事件日志文件夹设置权限，使用设置中允许和非允许用户对Web　UI进行查看、对事件日志文件夹进行读、写、移动、重命名操作。 |
| 非功能性需求-效率测试 | TC014 | Spark集群数据处理性能测试 | 2016.5.20 | 孟翰 | 苏若 | 选取Spark对movielens数据集进行机器学习的过程作为性能测试，测试十次取平均值。 |

回归测试主要包括再测试全部用例、基于风险选择测试、基于操作剖面选择测试、再测试修改的部分这四种方法，这里，若进行多次回归测试，则在前三次选择再测试全部用例的方法进行回归测试，随着回归测试次数的增加，选择再测试修改的部分的方法更能综合体现效率和有效性。

* 1. 测试过程
     1. 功能性测试
        1. TC001 : Spark机器学习结果测试

功能性需求：Spark机器学习

TC001-1：<http://grouplens.org/datasets/movielens/下载100M数据，选取ratings.dat作为数据处理文件>



经统计共有数据项10008074条，有数据缺失的数据项8020条，占比很小可忽略不计，因此将此8020条视作噪声点剔除，经测试剩余数据都为完整数据，测试通过。



TC001-2：原始数据项为String类型且有四列条目：userid、movieid、scores、time，本数据处理测试中time为多余数据需要剔除，另外需将userid、movieid转换为int类型，score转换为float类型，spark提供数据转换操作：



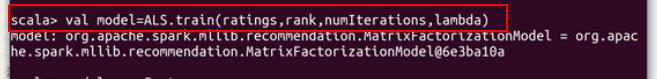
转换完毕，通过生成中间文件的方式测试结果与原数据是否相符，经检测数据与原数据相符，测试通过。

TC001-3：spark的机器学习推荐模型提供三项参数对模型进行训练，分别为rank，iterations，lambda：

Rank：对应ALS模型中的因子个数，即在低阶近似矩阵中的隐含特征个数。因子个数一般越多越好，提高因子个数的同时也会提高模型训练和保存时所需的内存开销，通常合理取值在10-200.

Iterations：运行时迭代次数，一般经少数次迭代后ALS模型就能收敛为一个比较合理的好模型。一般取值为10左右、

Lambda：控制模型正则化过程，从而控制模型过度拟合的情况、其值越高，正则化越严厉。该参数与实际数据的大小、特征和稀疏程度有关、



模型的训练结果好坏通常可以通过余弦方差进行计算，方差越接近于0代表训练效果越好，前两个参数根据集群性能越大越好，lambda参数需根据具体数据情况进行设置，经测试，在集群能够负载的条件下，rank值150，iterations值10，lambda值0.03时余弦方差最小，此时为0.00467，可看作为当前集群下的近似最优解，测试通过。

TC001-4：根据生成的结果即可对用户行为进行预测，本测试的训练数据与测试数据比重为8比2，读取测试数据进行测试，验证训练模型的正确率：

总测试数据项：2000015

验证正确数据项：1969867

准确率：98.493%

经测试，预测模型准确率较高，符合要求，测试通过。

* + - 1. TC002 : Spark执行SQL数据处理测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 测试数据选择 | 选择Spark自带Json文件进行测试（在./examples/src/main/resources文件夹中有一个名为people.json的文件） | | |
| 文件内容 | {"name":"Michael"}  {"name":"Andy", "age":30}  {"name":"Justin", "age":19} | | |
| 2 | 启动Spark shell |  | | |
| 3 |  | 主要命令 | | |
| 创建SQLContext | val sqlContext = new org.apache.spark.sql.SQLContext(sc) | | |
| 导入数据源 | val df = sqlContext.read.json("examples/src/main/resources/people.json") | | |
|  | 测试要求 | 执行操作 | 中间结果 | 测试结果说明 |
| 测试过程 | TC002-1：查询并显示文件内容 | df.show() | 输出数据源的内容  // age name  // null Michael  // 30 Andy  // 19 Justin | 输出内容与数据源内容一致，**与预期结果一致，**执行查询语句成功 |
| TC002-2：选择查询（查询“name”） | df.select("name").show() | 将文件中name内容显示为一列  // name  // Michael  // Andy  // Justin | 输出为数据源全部name信息，**与预期结果一致，**执行选择查询语句成功 |
| TC002-3：统计查询（按照age统计、按照name统计） | df.groupBy("age").count().show()  df.groupBy("name").count().show() | 输出统计结果  按照age统计：  // age count  // null 1  // 19 1  // 30 1  按照name统计  // name count  // Michael 1  // Andy 1  //Justin 1 | 输出结果为数据源中age、name的统计结果，**与预期结果一致，**  执行统计查询语句成功 |
| TC002-4：修改（将所有人年龄加1）并显示修改结果 | df.select(df("name"), df("age") + 1).show() | 输出修改后的文件内容name和age+1列  // name (age + 1)  // Michael null  // Andy 31  // Justin 20 | 输出结果显示数据源中所有人的年龄增加一（年龄为null的除外），**与预期结果一致，**执行修改语句成功。 |
| TC002-5：增加 | 将DataFrame注册为临时表（person），然后执行val insertSql = sqlContext.sql (insert into person(“name”,“age”) values(“Helen”,“19”))  df.show() | 输出增加后的文件内容  // name age  // Michael null  // Andy 31  // Justin 20  // Helen 19 | 输出结果为在源文件原有内容的基础之上增加了一组数据，**与预期结果一致，**执行添加语句成功。 |
| TC002-6：删除 | 将DataFrame注册为临时表（person），然后执行val deleteSql = sqlContext.sql (delete from person where name =”Helen”)  df.show() | 输出删除后的文件内容  // name age  // Michael null  // Andy 31  // Justin 20 | 输出结果为在原有内容的基础之上删除了一组数据，**与预期结果一致，**执行删除语句成功。 |
| TC002-7：执行SQL语句查询 | 首先将DataFrame注册为临时表，然后执行  val df = sqlContext.sql("SELECT \* FROM table")  df.show() | 输出数据源的内容  // age name  // null Michael  // 30 Andy  // 19 Justin | 输出内容与数据源内容一致，**与预期结果一致，**执行SQL语句进行查询成功 |

* + - 1. TC003 : Spark Stream流计算测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 创建StreamingContext对象并且设置batch interval为1秒,统计的时间周期无60s | val ssc = new StreamingContext(conf, Seconds(1)) | | |
|  | 确定测试数据 | 通过TCP套接字获取数据服务器上的文本数据val lines = ssc.socketTextStream("localhost", 8888) | | |
| 确定测试内容 | 文本数据中单词的总数 | | |
|  | TC003-1：统计 | 编程实现count（统计每一个batch中的单词数目），使用saveAsTextFiles将中间结果以文本的形式保存为文本文件 | 输出文件中为每个batch的单词统计结果 | 中间结果被保存，与预期结果一致 |
| TC003-2：输出结果 | 使用print()在Driver中打印出DStream中数据的部分元素统计结果。 | 每秒打印一些生成的“和”，以及最终统计结果和运行时间T1=0.567s。 | 打印出最终统计结果 |
| TC003-3：修改batch interval为2秒，并重复以上过程 | 打印出统计结果 | 每秒打印一些生成的“和”，以及最终统计结果和运行时间T=1.061s | 时间较T1中长 |
| TC003-4： 修改batch interval为500ms，并重复以上过程 | 打印出统计结果 | 每秒打印一些生成的“和”，以及最终统计结果和运行时间T=5.010s | 时间非常长 |

* + 1. 非功能性测试
       1. TC011 : Spark鲁棒性测试

鲁棒性测试主要模拟以下几种情景：

1. 用户非法输入：对于语法错误，编译器会直接报告异常，对于数据操作的错误，spark只有在真正对数据进行操作时才会发现并回溯操作过程，运行时程序未出现异常，测试通过。
2. 内存容量不足：在模拟大量数据处理时，spark出现内存不足的情况，此时spark会自动将临时数据存储至硬盘中，并未出现异常，测试通过。
3. 节点失连：在节点突然失连的情况下，其他节点依然正常工作，但此节点的数据处理任务没有成功完成，在回归测试中，更改任务调度策略为yarn，此时当节点失连时，会将任务重新分配至其他节点，测试表明spark默认任务调度策略有时会无法分配丢失节点的任务，yarn调度不会出现这种问题，测试未通过。
   * + 1. TC012 : Spark容错性测试

Spark提供血统机制来保证用户错误转换rdd后进行恢复操作，经测试，20次模拟错误操作再进行恢复，spark均能恢复到错误之前的系统状态，数据也未出现丢失、改变等异常情况，测试通过。

* + - 1. TC013 : Spark安全性测试

Spark的安全性测试主要是对Spark的Web UI安全性、事件审计安全以及网络端口安全进行测试，分别如下：

Web UI安全性:

TC013-1: 使用A用户账户登录，并设置spark.ui.filters，设置spark.ui.view.acls列表为空，启用javax.servlet.filters；使用用户账户B登录并试图方位Spark web UI，测试结果为访问失败；

TC013-2:将B用户加入到spark.ui.view.acls用户列表中，使用用户账户B登录并试图方位Spark web UI，测试结果为访问成功；

证明可以通过设置，启动javax.servlet.filters实现Spark web UI安全性。

事件审计安全:

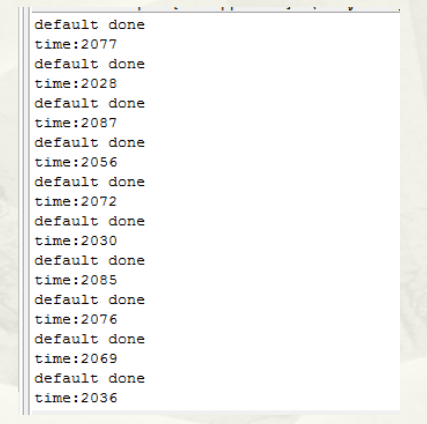
TC013-3: 使用A用户账户登录,创建存放事件日志的文件夹，并且设置这个文件夹为drwxrwxrwxt权限，使用用户账户B登录，查看事件日志文件夹，并试图进行写操作、移动文件夹、重命名文件夹——结果为B可以对文件进行读和写操作，但是不能移动和重命名文件夹。

网络端口安全:

TC013-4：参照本文档2.2.1.3节 TC003 : Spark Stream流计算测试内容，使用Stream对多媒体数据进行处理，测试中系统要求进行身份认证，证明Spark对网络通信安全有很高要求。

* + - 1. TC014 : Spark效率测试

效率测试采用简单的java时间戳方法，即System.currentMills设置于数据处理前后，两次时间相减为spark的数据处理运行时间（单位：毫秒），我们处理三十次相同数据（spark没有记忆功能），得到三十次时间取平均值



得到平均时间为2048毫秒，本次测试数据容量大小为108MB左右，即在双节点集群下Spark的处理速度为50MB/s左右，相比hadoop性能有较大提升，测试通过。

1. 测试结果

1、第一次测试

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试需求项** | **测试需求项标识** | **测试用例** | **测试用例标识** | **测试结果** | **软件问题** |
| 功能性需求 | TR01 | Spark机器学习结果测试 | TC001 | 通过 | / |
| Spark执行SQL数据处理 | TC002 | 通过 | / |
| Spark Stream流计算测试 | TC003 | 通过 | / |
| 非功能性需求 | TR11 | Spark鲁棒性测试 | TC011 | 不通过 | SPR02 |
| Spark容错性测试 | TC012 | 通过 | / |
| Spark安全性测试 | TC012 | 通过 | / |
| Spark效率测试 | TC014 | 通过 | / |

2、回归测试

|  |  |  |  |
| --- | --- | --- | --- |
| **测试需求项标识** | **测试用例标识** | **测试结果** | **软件问题** |
| TR01 | TC001 | 通过 | / |
| TC002 | 通过 | / |
| TC003 | 通过 | / |
| TR11 | TC011 | 通过 | / |
| TC012 | 通过 | / |
| TC013 | 通过 | / |
| TC014 | 通过 | / |
| … |  |  |  |

1. 测试结论

在本次实验过程中，共发现3个软件问题，即数据量过大导致Java堆溢出、RDD转换操作链过长导致线程栈溢出和节点失连导致丢失部分处理数据，其中严重问题2个，较严重问题1个（节点失连导致丢失部分处理数据），除了节点失连导致丢失部分处理数据这个问题之外全部解决。

附1：软件问题报告清单

|  |  |  |  |
| --- | --- | --- | --- |
| **软件问题标识** | **软件问题简述** | **严重程度** | **状态** |
| SPR01-1 | 数据量过大导致Java堆溢出 | 严重 | 修复 |
| SPR01-2 | RDD转换操作链过长导致线程栈溢出 | 严重 | 修复 |
| SPR01-3 | 节点失连导致丢失部分处理数据 | 较严重 | 未修复 |