Apache Spark

—RDD与Storage需求规格说明书

组员：

SY1506404 孟翰

SY1506409 苏若

SY1506425 李璇

SY1506406 孙敏芳

目录

[1 引言 4](#_Toc448254182)

[1.1 文档编写目的 4](#_Toc448254183)

[1.2 系统标识 4](#_Toc448254184)

[1.3 系统概述 5](#_Toc448254185)

[1.3.1 Spark 5](#_Toc448254186)

[1.3.2 RDD 5](#_Toc448254187)

[1.3.3 Storage 5](#_Toc448254188)

[1.3.4 RDD和Storage两模块的关系 6](#_Toc448254189)

[1.4 分析过程 6](#_Toc448254190)

[1.5 文档概述 6](#_Toc448254191)

[1.6 数字字典 7](#_Toc448254192)

[2 功能性需求 9](#_Toc448254193)

[2.1 用例图 9](#_Toc448254194)

[2.2 可以处理多种类型的数据 10](#_Toc448254195)

[2.3 可应用于各种大数据处理场景 10](#_Toc448254196)

[2.4 用户可自主选择容错方式 15](#_Toc448254197)

[2.5 用户可以命名、物化、控制中间结果的存储、分区。 16](#_Toc448254198)

[2.6 Storage用例图 16](#_Toc448254199)

[3 非功能性需求 20](#_Toc448254200)

[3.1 鲁棒性 20](#_Toc448254201)

[3.2 容错性 20](#_Toc448254202)

[3.3 安全性 21](#_Toc448254203)

[3.3.1 Web UI安全 21](#_Toc448254204)

[3.3.2 事件审计安全 21](#_Toc448254205)

[3.3.3 网络端口安全 21](#_Toc448254206)

[3.4 效率 22](#_Toc448254207)

[4.参考文献 23](#_Toc448254208)

**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 | 2016.03.24 | 孟翰、李璇、孙敏芳、苏若 |  | 初始版本 |
| 2.0 | 2016.04.04 | 孟翰、李璇、孙敏芳、苏若 |  | V2.0 |
| 3.0 | 2016.04.12 | 孟翰、李璇、孙敏芳、苏若 |  | V3.0 |
|  |  |  |  |  |

**修改记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档名称 | Apache Spark  —RDD与Storage需求规格说明书 V2.0 | 修改日期 | 2016.04.12 |
| 编号 | 修改内容 | | |
| 1 | 增加用例图 | | |
| 2 | 已有用例的UseCase的RUCM 重新截图（因为有评价说我们组RUCM截图太小） | | |
| 3 | 参考文献按照提议安排在需求规格说明书的最后一节 | | |
| 4 | 增加修改记录 | | |
| 5 | 增加数据字典的内容 | | |
| 6 | 增加Storage描述和UseCase；以及DiskStore和MemoryStore的存取Block的过程和RUCM图。 | | |

# 1 引言

## 文档编写目的

以Apache Spark开源软件及相关资料为输入依据，在对其进行可行性研究

后，对本课程实验相关问题进行进一步定位，确定获取和需求分析的方法，并对确定的软件需求进行定义和描述。之后将上述过程以可视的文档形式表达出来，输出为软件需求规格说明书。通过该文档的编写，可以保证以下目标的实现：

1. 便于开发人员进行理解和交流。
2. 反映用户问题的结构，可作为软件开发工作的基础和依据。
3. 作为确认测试和验收的依据。

## 1.2 系统标识

Spark版本号： spark1.6.1

Spark源码实现语言：Scala

可编程语言：Scala.+Scala shell、Java、Python+Scala shell

环境：jdk1.8

需求报告版本：V1.0

## 1.3 系统概述

### 1.3.1 Spark

Spark是UC Berkeley AMP lab所开源的类Hadoop MapReduce的通用的并行计算框架，分布式资源工作交由集群管理软件（如Mesos , Standalone ,YARN）.

从整体架构而言:

1. Spark提供了多种高级工具，如Shark SQL应用于查询、Spark Streaming应用于流式计算、MLib应用于机器学习、GraphX应用于图处理。
2. Spark可以访问存储在HDFS、Hbase、Cassandra、Amazon S3、本地文件系统等上的数据，并支持多种文件格式，如文本文件、序列文件及Hadoop的InputFormat。

从核心模块而言：Spark核心组件包括RDD、Scheduler、Storage、Shuffle四

部分。在这里，主要针对需深入研究的RDD和Storage模块进行细节性概述。

### 1.3.2 RDD

a．数据处理模型——RDD混合了Iterative Algorithms，Relational Queries，

Hadoop MapReduce采用的MapReduces及Storm采用的Stream Processing四种模型，使得Spark可以应用于各种大数据处理场景。

b．5个属性—— 一组分片（Partition）、一个计算每个分片的函数、依赖（Dependencies）、preferredLocations、partitioner。

c．两个操作——转换（transformation）和动作（action）。

d． 11种存储级别——RDD根据useDisk、useMemory、deserialized、off\_heap、

replication五个参数的组合设置了11种存储级别，其中默认级别MEMORY\_ONLY。

e．两种依赖关系——窄依赖和宽依赖。

### 1.3.3 Storage

a．子模块1——通信层。

在该层，采用master-slave结构传输控制信息和状态信息。

b．子模块2——存储层。

该层为存储数据到disk或是memory，甚至replicate到远端提供了相应

接口。同时，提供了统一的操作类BlockManager，以完成其他模块和storage模块的交互。

### 1.3.4 RDD和Storage两模块的关系

RDD提供了的各种transformation和action接口以实现应用，提高了抽象层次，在接口和实现上进行有效地隔离，使用户无需关心底层的实现。

但对于所操作的数据究竟放在哪里，如何存取？这是由storage模块来实现和管理的。可以说，RDD实现了用户的逻辑，而Storage则管理了用户的数据。举例来说，用户在实际编程中，面对的是RDD，可以将RDD的数据通过调用org.apache.spark.rdd.RDD#cache将数据持久化，而持久化的动作都是由Storage模块完成的。

**1.4 分析过程**

在需求过程分析中，本小组首先选定Apache Spark开源软件作为分析对象。在对其特性、体系架构及源码进行综合分析后，决定对其主要的RDD和Storage模块进行分析研究，使用Spark对Gowalla签到数据集进行分析，研究SparkRDD模块和Storage模块使用方法，并且通过对Spark与Java程序进行数据分析的效率对比，研究Spark性能特性。

Gowalla 是一个基于位置的签到社交网站，用户通过Gowalla进行签到，并上传签到的相关数据（时间、地点），公共API将用户的签到数据进行收集。

我们分析的签到数据集中包含了Gowalla一共196591个用户在2009年2开始到2011年10月期间的6,442,890次签到信息的数据，包含签到用户的ID、签到时间、签到地点（经度、纬度、地点ID）。

签到数据集每条签到信息如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User | Check-in time | latitude | longitude] | location id |
| 196585 | 2010-10-07T17:39:18Z | 50.0278115273 | 8.7850981951 | 4555073 |

我们组研究内容包含两个部分，第一个确定每个UserID(用户)的签到总次数，并且使用Java编程实现相同的功能，通过统计对比两者不同的运行时间，研究Spark的数据分析效率；第二，研究用户行为相似度，此分为两个部分，首先以相同LocationID(签到地点)为分组依据，统计各分组下的签到信息；然后通过研究不同用户在相同地点签到时间的相近程度，研究用户行为的相似度。第二部分内容初步设想采取协同过滤算法进行数据分析。

## 1.5 文档概述

本文档对Apache Spark的RDD及Storage模块的需求分析进行说明，包括这些模块的功能性需求和非功能性需求。

## 1.6 数字字典

|  |  |
| --- | --- |
| 数据字典名称 | Scheduler |
| 简介 | Spark组件 |
| 数据定义 | Spark的调度机制，可分为DAGScheduler和TaskScheduler。 |

|  |  |
| --- | --- |
| 数据字典名称 | Shuffle |
| 简介 | Spark组件 |
| 数据定义 | shuffle是连接Map和Reduce之间的桥梁，Map的输出要用到Reduce中必须经过shuffle这个环节，shuffle的性能高低直接影响了整个程序的性能和吞吐量。 |

|  |  |
| --- | --- |
| 数据字典名称 | RDD |
| 简介 | Spark组件 |
| 数据定义 | Resilient Distributed Datasets，是一个容错的、并行的数  据结构，可以让用户显式地将数据存储到磁盘和内存中，并  能控制数据的分区。 |

|  |  |
| --- | --- |
| 数据字典名称 | Storage |
| 简介 | Spark组件 |
| 数据定义 | 负责在Spark计算过程中，包括基于Disk的和基于Memory  的数据存储模块。 |

|  |  |
| --- | --- |
| 数据字典名称 | 窄依赖 |
| 简介 | RDD间依赖关系 |
| 数据定义 | 一个父RDD最多被一个子RDD用。 |

|  |  |
| --- | --- |
| 数据字典名称 | 宽依赖 |
| 简介 | RDD间依赖关系 |
| 数据定义 | 子RDD的分区依赖于父RDD的所有分区。 |

|  |  |
| --- | --- |
| 数据字典名称 | BlockManager |
| 简介 | Storage模块内的类 |
| 数据定义 | Storage模块与其他模块交互最主要的类，它提供了读和  Block的接口。 |

|  |  |
| --- | --- |
| 数据字典名称 | Block |
| 简介 | Storage模块存取数据的最小单位 |
| 数据定义 | 每一个RDD partition都会对应一个Block，由唯一的标识，  格式是"rdd\_" + rddId + "\_" + partitionId。 |

|  |  |
| --- | --- |
| 数据字典名称 | Iterative Algorithm |
| 简介 | 迭代算法 |
| 数据定义 | Iterative Algorithm是指通过一个厨师估计值出发寻找一个  近似解来解决问题，最后通过不断重复来缩小与真实解之  间的差距；在图应用和机器学习领域很常见的一中算法。 |

|  |  |
| --- | --- |
| 数据字典名称 | Hadoop Mapreduce |
| 简介 | Hadoop MapReduce是一个使用简易的软件框架 |
| 数据定义 | 一个Map/Reduce 作业通常会把输入的数据集切分为若干  独立的数据块，由 map任务（task）以完全并行的方式处  理它们。Reducer任务接收Mapper任务的输出，归约处理  后写入到HDFS |

|  |  |
| --- | --- |
| 数据字典名称 | Stream Processing |
| 简介 | 流式数据处理 |
| 数据定义 | 以优秀的调度机制、快速的分布式计算能力实现对实时的  流式数据处理的 |

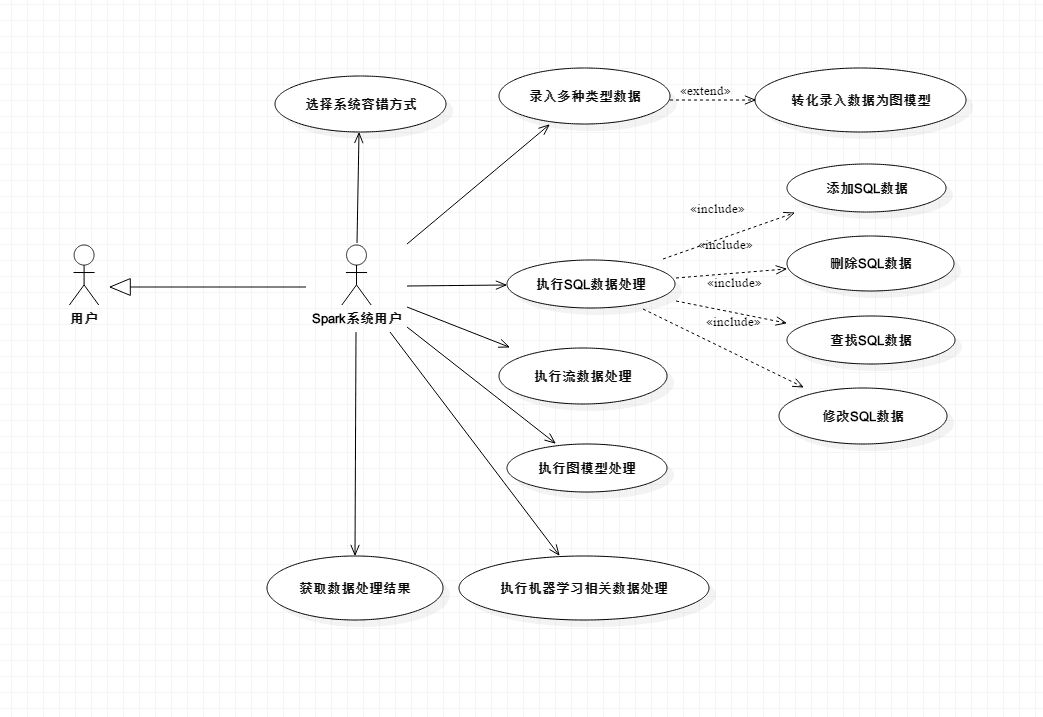
|  |  |
| --- | --- |
| 数据字典名称 | Spark Streaming |
| 简介 | Spark的流式框架 |
| 数据定义 | Spark Streaming用于流式数据的处理；具有高吞吐量和容  错能力强这两个特点。其支持的数据源包括Kafka、Flume、  Twitter、ZeroMQ 和简单的 TCP 套接字等；与MLlib（机  器学习）以及 Graphx 完美融合。 |

|  |  |
| --- | --- |
| 数据字典名称 | Checkpoint |
| 简介 | Spark容错机制的一种 |
| 数据定义 | Checkpoint将足够多的信息checkpoint到某些具备容错性  的存储系统如HDFS上，以便出错时能够迅速恢复；包括  Metadata checkpointing和Data checkpointing |

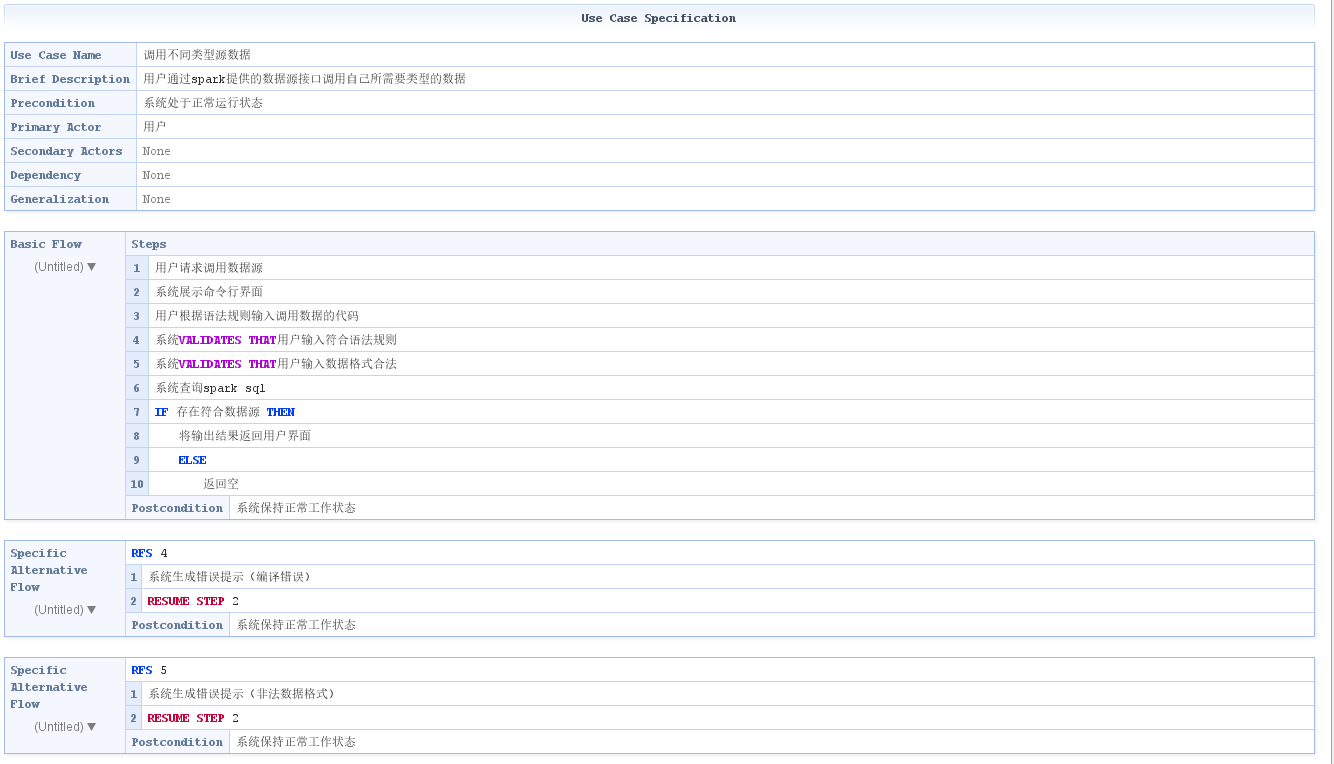
|  |  |
| --- | --- |
| 数据字典名称 | Lineage |
| 简介 | 血统关系—Spark容错机制的一种 |
| 数据定义 | 血统关系描述了RDD之间的演变关系，记录RDD的粗颗  粒度的特定数据Transformation操作（如filter、map、join  等）行为；当这个RDD的部分分区数据丢失时，它可以  通过Lineage获取足够的信息来重新运算和恢复丢失的数  据分区 |

# 2 功能性需求

## 2.1 用例图



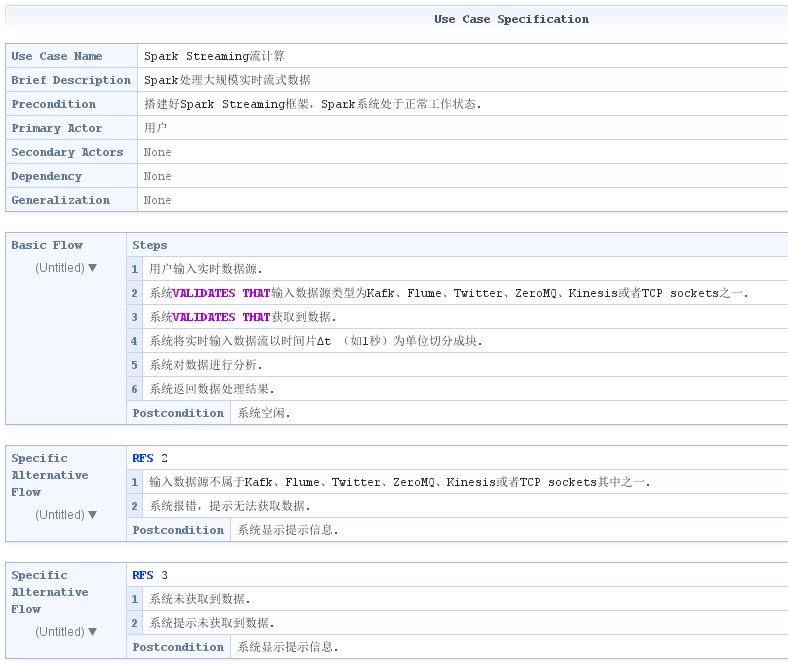
## 2.2 可以处理多种类型的数据



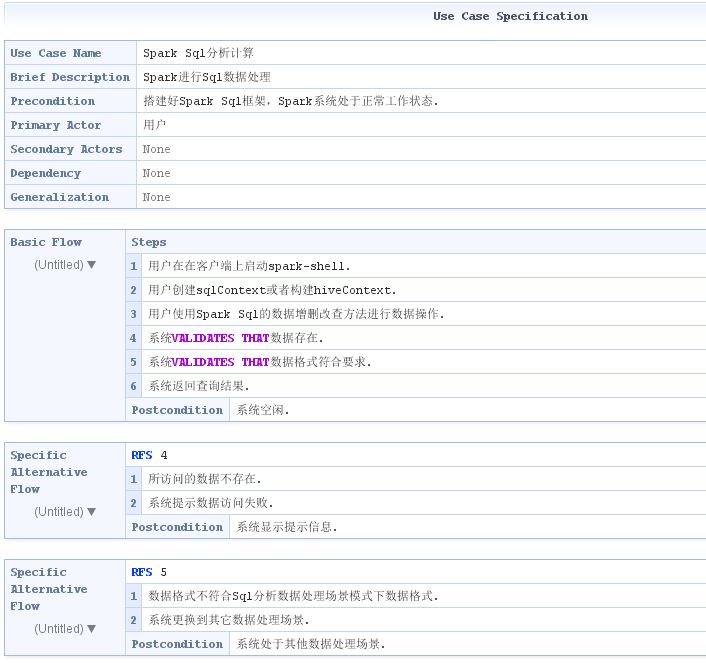
## 2.3 可应用于各种大数据处理场景

数据处理模型——RDD混合了Iterative Algorithms，Relational Queries， Hadoop MapReduce采用的MapReduces及Storm采用的Stream Processing四种模型，使得Spark可以应用于各种大数据处理场景。

1. Stream流计算



1. Sql分析计算

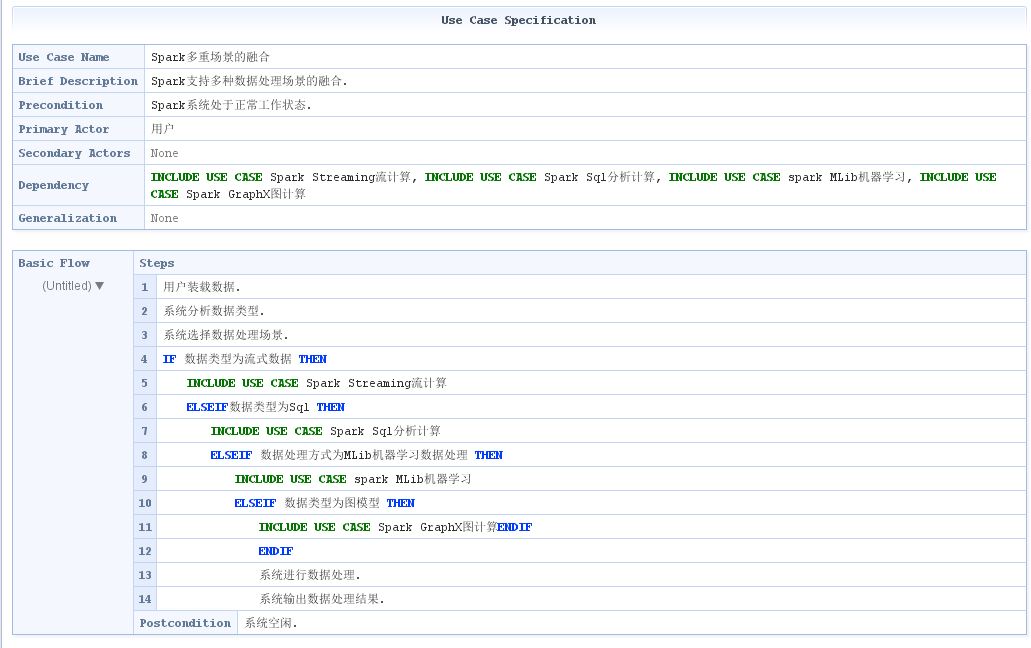


1. MLib机器学习



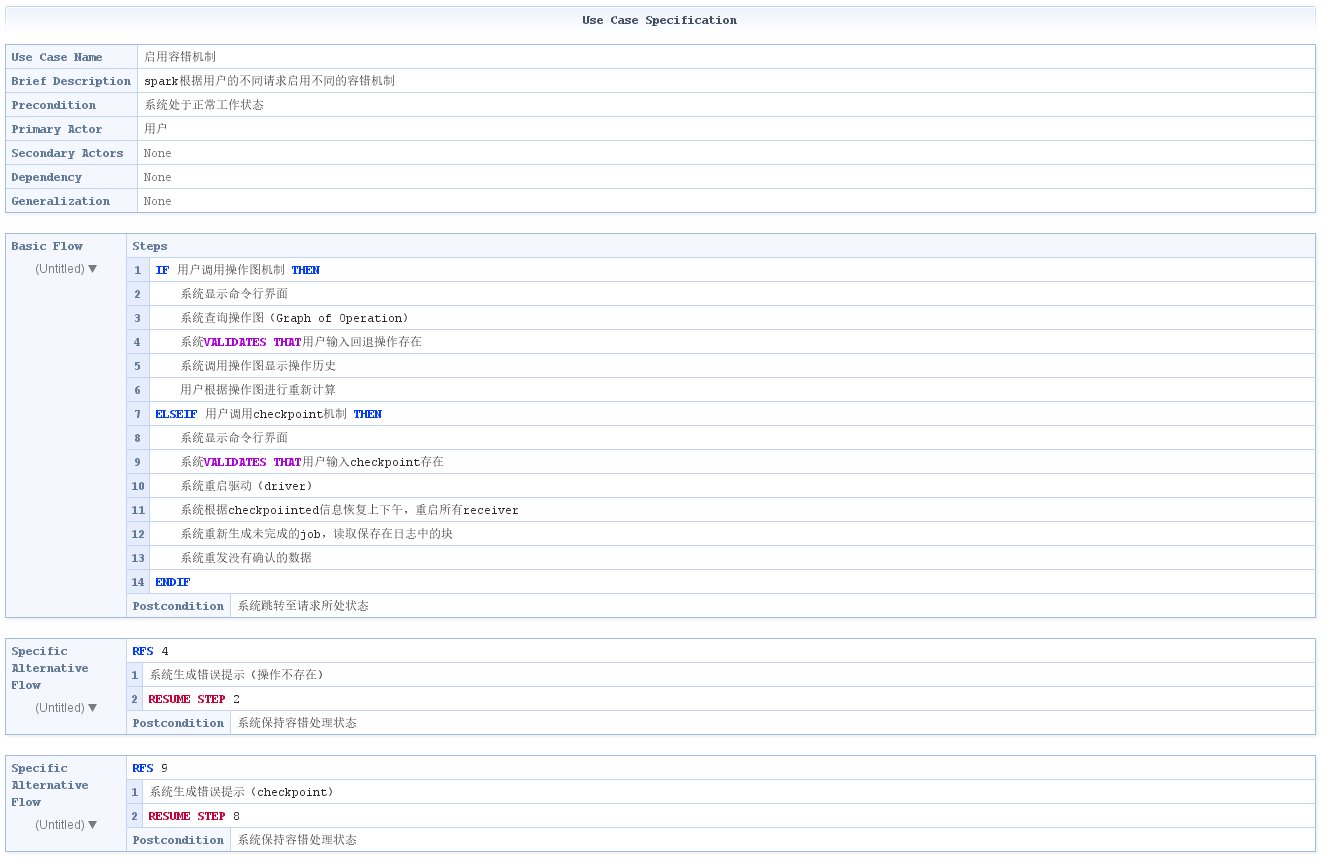
1. GraphX图计算





## 2.4 用户可自主选择容错方式

在RDD计算，通过checkpint进行容错，做checkpoint有两种方式，一个是checkpoint data，一个是logging the updates。用户可以控制采用哪种方式来实现容错，默认是logging the updates方式，通过记录跟踪所有生成RDD的转换（transformations）也就是记录每个RDD的lineage（血统）来重新计算生成丢失的分区数据。



## 2.5 用户可以命名、物化、控制中间结果的存储、分区。

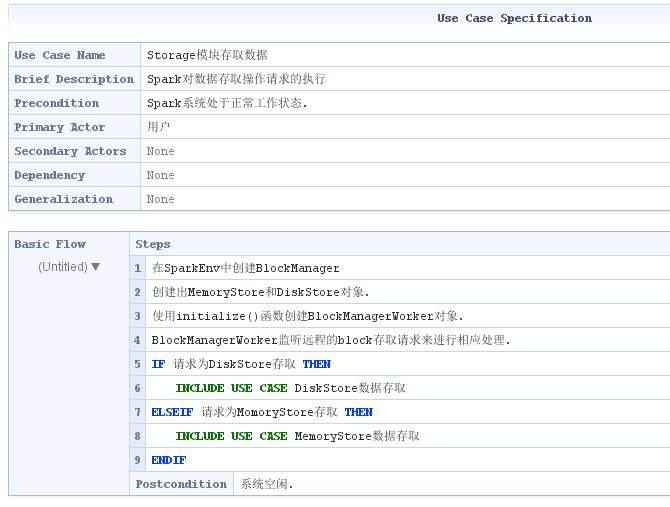
C:\Users\L.X\AppData\Roaming\Tencent\Users\1668856072\QQ\WinTemp\RichOle\]]H$(`N$X7D~(81XH`LJZ}1.png

## 2.6 Storage功能分析

RDD是由不同的partition组成的，我们所进行的transformation和action是在partition上面进行的；而在storage模块内部，RDD又被视为由不同的block组成，对于RDD的存取是以block为单位进行的，本质上partition和block是等价的，只是看待的角度不同。在Spark storage模块中存取数据的最小单位是block，所有的操作都是以block为单位进行的。

1. Storage模块存取数据

首先在SparkEnv中创建BlockManager对象，同时创建出MemoryStore和DiskStore对象用以存取block；在initialize()函数中创建BlockManagerWorker对象用以监听远程的block存取请求来进行相应处理。然后根据不同的情况对于DiskStore和MemoryStore，采取不同的block的存取操作。



1. DiskStore存取block

DiskStore会在对象被创建时调用createLocalDirs()来创建文件夹，所有的block都会存储在所创建的folder里面。在DiskStore里面，每一个block都被存储为一个file，通过计算block id的hash值将block映射到文件中；根据hash值将block id映射成相应的文件路径；block id为文件名创建file handler，DiskStore使用此file handler将block写入文件内；

而获取block则非常简单，将block id映射成相应的文件路径，然后读取Block即可。



1. MemoryStore存取block

MemoryStore内部维护了一个hash map来管理所有的block，以block id为key将block存放到hash map中。在MemoryStore中存放block必须确保内存足够容纳下该block，若内存不足则会将block写到文件中，因此需要先计算存储的block所需的内存大小，判断是否大于可用内存大小，如果大于则通过调用dropFromMemory()将block写入文件；

从MemoryStore中取得block只需从hash map中取出block id对应的value即可。用例图如下：



# 3 非功能性需求

## 3.1 鲁棒性

鲁棒性就是系统的健壮性，指系统或组件在面对非法输入、相连接的系统或组件故障、或非预期的运行条件下能够持续正确运行的程度。

为了保证RDD中数据的鲁棒性，RDD数据集通过所谓的血统关系(Lineage)记住了它是如何从其它RDD中演变过来的。相比其它系统的细颗粒度的内存数据更新级别的备份或者LOG机制，RDD的Lineage记录的是粗颗粒度的特定数据变换（Transformation）操作（filter, map, join etc.)行为。当这个RDD的部分分区数据丢失时，它可以通过Lineage获取足够的信息来重新运算和恢复丢失的数据分区。这种粗颗粒的数据模型，限制了Spark的运用场合，但同时相比细颗粒度的数据模型，也带来了性能的提升。

总之，Spark的核心思路就是将数据集缓存在内存中加快读取速度，同时用lineage关联的RDD以较小的性能代价保证数据的鲁棒性。

## 3.2 容错性

对于大规模数据处理来讲，容错性是其中比较重要的一环。与在软件故障或违反指定接口的情况下维持规定的性能水平的能力有关的软件属性（如离线录入支持等）叫做容错性。

一般来说，分布式数据集的容错性有两种方式：即数据检查点和记录数据的更新。我们面向的是大规模数据分析，数据检查点操作成本很高：需要通过数据中心的网络连接在机器之间复制庞大的数据集，而网络带宽往往比内存带宽低得多，同时还需要消耗更多的存储资源（在内存中复制数据可以减少需要缓存的数据量，而存储到磁盘则会拖慢应用程序）。所以，我们选择记录更新的方式。但是，如果更新太多，那么记录更新成本也不低。因此，RDD只支持粗粒度转换，即在大量记录上执行的单个操作。将创建RDD的一系列转换记录下来（即Lineage），以便恢复丢失的分区。

同时，RDD独有的宽/窄依赖的概念，在RDD Transformation操作中间发生计算失败时，对容错性会有很大帮助。具体描述如下：

（1）运算是窄依赖。只需把丢失的父RDD分区重算即可，这样可以大大加快场景恢复的开销。

（2）增加检查点。当Lineage特别长时或者有宽依赖时，主动调用checkpoint把当前的数据写入稳定存储，作为检查点。

## 3.3 安全性

### 3.3.1 Web UI安全

a．Spark提供的javax.servlet.filters可以提高Web UI安全性,支持用户自主选择允许其他人访问自己数据的权限。具体来看，用户通过使用javax.servlet.filters对其他用户进行验证，一旦其他用户登录进入Spark，系统此时会在该用户与视图访问控制列表之间进行比较分析，以确保该用户有权查看所有者用户的UI界面。

b．Spark支持通过修改访问控制列表的方式来控制哪个用户可以访问、修改正在运行的Spark应用，其中包括终止一个应用或任务。

c．Spark允许管理员在访问控制列表中指定哪个用户对所有应用程序总都具有查看、修改权限。

### 3.3.2 事件审计安全

Spark支持事件审计安全设置。通过进行相应设置，可以设置非所有者的其他用户的使用权限。如：允许非所有者的其他用户可以在该文件夹下写，但是不能移动或者重命名文件。这样事件日志只会被root用户和Spark系统生成和修改，从而保证其安全性。

### 3.3.3 网络端口安全

Spark Streaming支持大规模流式数据处理，其安全威胁与多媒体云安全有共通之处。在利用Spark Streaming进行多媒体数据处理时，需要进行身份认证，同时利用安全协议如RTMP将多媒体数据进行加密传输。

## 3.4 效率

Spark的RDD模块的高效率主要体现在以下几方面：

1、RDD只能从持久存储或通过Transformations操作产生，相比于分布式共享内存（DSM）可以更高效实现容错，对于丢失部分数据分区只需根据它的lineage就可重新计算出来，而不需要做特定的Checkpoint。

2、RDD的数据分区特性，可以通过数据的本地性来提高性能。

3、RDD在需要进行分区把数据分布于集群中时会根据每条记录Key进行分区（如Hash 分区），保证了两个数据集在Join时能高效。

4、RDD操作分为transformation和action。所有transformation都是“惰性”的，只有在执行action操作之后，所有的operation才会被真正执行，大大提高类系统的性能。

# 4.参考文献

[1] <http://spark.apache.org/>

[2] <https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html>

[3] <http://jerryshao.me/architecture/2013/10/08/spark-storage-module-analysis/>

[4] <http://jingyan.baidu.com/article/90bc8fc80960f1f653640ce0.html>

[5] <http://www.lxway.com/66982556.htm>

[6] <http://blog.csdn.net/colorant/article/details/8255958>

[7] <http://tech.uc.cn/?p=2116>

[8] <http://shiyanjun.cn/archives/744.html>

[9] <https://www.zhihu.com/question/36532979>

[10] <http://blog.csdn.net/hsluoyc/article/details/43977779>

[11] <http://www.chinacloud.cn/show.aspx?id=23013&cid=14>

[12] 张安站. Spark技术内幕. 机械工业出版社[M]. 2015-09.

[13] 许鹏.《Apache Spark源码剖析》. [电子工业出版社](http://book.jd.com/publish/%E7%94%B5%E5%AD%90%E5%B7%A5%E4%B8%9A%E5%87%BA%E7%89%88%E7%A4%BE_1.html): 2015-03-01.

[14] Ji, Changqing, et al. "Big data processing: Big challengesand opportunities." Journal of Interconnection Networks 13.03n04 (2012).

[15] Rahimian, Fatemeh. "Gossip-Based Algorithms for InformationDissemination and Graph Clustering." (2014).