

变更与管理分析报告

组员：











SY1506404 孟翰
SY1506409 苏若
SY1506425 李璇
SY1506406 孙敏芳

一、目的

在软件研发过程中，采用有效的方法进行软件变更控制和版本管理，使各项工作能够有条不紊的进行。

二、实验工具

在配置管理过程中，选用了 GitHub 版本控制系统作为管理工具。同时，为使项目文件具有较为合理的组织结构，在 Hadoop 上配置本组文档组织结构如下：

 1.会议记录	2016/5/6 9:11	文件夹
 2.问题清单	2016/5/6 12:59	文件夹
 3.项目PPT	2016/5/6 9:21	文件夹
 4.项目提交文档	2016/5/6 9:11	文件夹
 5.项目资料	2016/5/6 12:27	文件夹
 6.检查单	2016/5/6 9:11	文件夹
 7.工作日志	2016/5/6 9:11	文件夹
 8.工作量统计	2016/5/6 9:11	文件夹
 Team-A 评审材料	2016/5/6 9:11	文件夹
 软工实验最终版	2016/5/6 9:11	文件夹

三、配置管理过程

3.1 《需求分析规格说明书》版本变更

表 1- 《需求分析规格说明书》版本变更

版本	产出时间	主要编制人	版本说明
1.0	2016.03.24	孟翰、苏若、李璇、孙敏芳	初始版本
2.0	2016.04.04	孟翰、苏若、李璇、孙敏芳	V2.0
3.0	2016.04.12	孟翰、苏若、李璇、孙敏芳	V3.0
4.0	2016.05.03	孟翰、苏若、李璇、孙敏芳	V4.0

3.1.1 《需求分析规格说明书 V1.0》变更

输入版本：1.0

变更产出版本：2.0

变更管理依据：《问题清单——第二次整理》、交流互动。

变更内容如下表所示：

表 2-《需求分析规格说明书 V1.0》变更过程

变更对象	变更原因描述	变更策略	具体实施
数据字典模块	在对数据字典的描述上与名词解释的差异较小	数据字典用表格的形式来定义软件的数据流图中出现的元素。数据字典在作用上包含名词解释。	优化数据字典表现形式，修改 1.6 节数据字典模块
数据字典模块	宽依赖表述不全面	对宽依赖概念深度学习	数据字典针对宽依赖表述变更如下：子 RDD 的分区依赖于父 RDD 的所有分区。
数据字典模块	窄依赖表述不全面	对窄依赖概念深度学习	数据字典针对窄依赖表述变更如下：一个父 RDD 最多被一个子 RDD 用。

3.1.2 《需求分析规格说明书 V2.0》变更

输入版本：2.0

变更产出版本：3.0

变更管理依据：《问题清单——第三次整理》、交流互动。

变更内容如下表所示：

表 3-《需求分析规格说明书 V2.0》变更过程

变更对象	变更原因描述	变更策略	具体实施
需求规格说明书 V2.0	缺少用例图	接受	在功能性需求中，添加用例图，如图 1 所示
数据字典模块	文档中的一些专有名词未在数据字典中给出。	接受，整理文档中出现的专有名词，并深度理解其概念	在数据字典中添加了以下名词描述：Iterative Algorithm、Hadoop Mapreduce、Stream Processing、Spark Streaming、Checkpoint 及 Lineage 具体如图 2 所示
RUCM 截图	截图太小	接受	RUCM 重新截图
需求规格说明书 V2.0	缺少针对 Storage 模块需求的 RUCM 用例描述。	接受	在功能性需求中，添加 DiskStore、MemoryStore 的存取 Block 的过程和 RUCM 图，具体如图 3 所示。
需求规格说明书 V2.0	涉及到了一些设计细节	对需求分析应涉及的内容划分出清晰的界限	从用户角度描述需求，删除文档中涉及到的系统机制等描述

需求规格说明书 V2.0	参考文献放在了文档第一节	为更符合大众需求，将其放置于文档结束处	将 1.7 节参考文献移至最后一节
-----------------	--------------	---------------------	-------------------

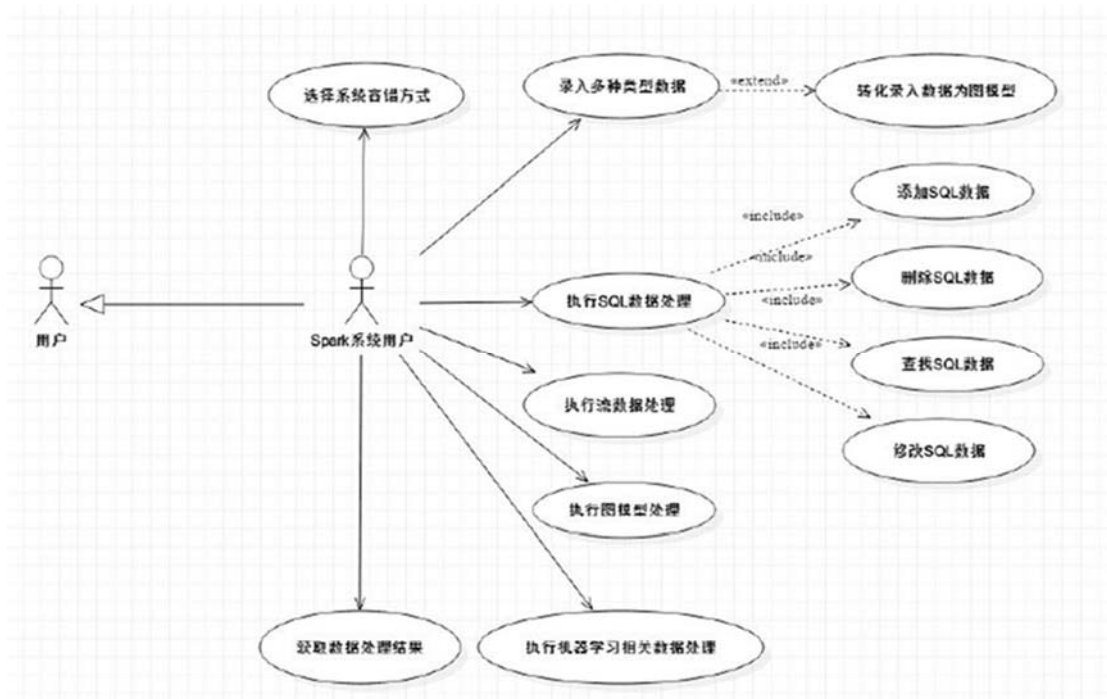


图 1 新增用例图

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

图 2 数据字典新增记录

Use Case Specification																							
Use Case Name	Storage模块存取数据																						
Brief Description	Spark对数据存取操作请求的执行																						
Precondition	Spark系统处于正常工作状态.																						
Primary Actor	用户																						
Secondary Actors	None																						
Dependency	None																						
Generalization	None																						
Basic Flow (Untitled) ▼	<table> <tr> <th>Steps</th><th></th></tr> <tr> <td>1</td><td>在SparkEnv中创建BlockManager</td></tr> <tr> <td>2</td><td>创建出MemoryStore和DiskStore对象.</td></tr> <tr> <td>3</td><td>使用initialize()函数创建BlockManagerWorker对象.</td></tr> <tr> <td>4</td><td>BlockManagerWorker监听远程的block存取请求来进行相应处理.</td></tr> <tr> <td>5</td><td>IF 请求为DiskStore存取 THEN</td></tr> <tr> <td>6</td><td>INCLUDE USE CASE DiskStore数据存取</td></tr> <tr> <td>7</td><td>ELSEIF 请求为MemoryStore存取 THEN</td></tr> <tr> <td>8</td><td>INCLUDE USE CASE MemoryStore数据存取</td></tr> <tr> <td>9</td><td>ENDIF</td></tr> <tr> <td>Postcondition</td><td>系统空闲.</td></tr> </table>	Steps		1	在SparkEnv中创建BlockManager	2	创建出MemoryStore和DiskStore对象.	3	使用initialize()函数创建BlockManagerWorker对象.	4	BlockManagerWorker监听远程的block存取请求来进行相应处理.	5	IF 请求为DiskStore存取 THEN	6	INCLUDE USE CASE DiskStore数据存取	7	ELSEIF 请求为MemoryStore存取 THEN	8	INCLUDE USE CASE MemoryStore数据存取	9	ENDIF	Postcondition	系统空闲.
Steps																							
1	在SparkEnv中创建BlockManager																						
2	创建出MemoryStore和DiskStore对象.																						
3	使用initialize()函数创建BlockManagerWorker对象.																						
4	BlockManagerWorker监听远程的block存取请求来进行相应处理.																						
5	IF 请求为DiskStore存取 THEN																						
6	INCLUDE USE CASE DiskStore数据存取																						
7	ELSEIF 请求为MemoryStore存取 THEN																						
8	INCLUDE USE CASE MemoryStore数据存取																						
9	ENDIF																						
Postcondition	系统空闲.																						

图 3.1 新增 Storage 模块 RUCM-- DiskStore 存取 block

Use Case Specification																															
Use Case Name	MemoryStore数据存取																														
Brief Description	Spark将数据存储到内存以及从内存中取数据																														
Precondition	Spark系统处于正常工作状态.																														
Primary Actor	用户																														
Secondary Actors	None																														
Dependency	None																														
Generalization	None																														
Basic Flow (Untitled) ▼	<table> <tr> <th>Steps</th><th></th></tr> <tr> <td>1</td><td>IF 请求为MemoryStore数据存储 THEN</td></tr> <tr> <td>2</td><td>MemoryStore对象根据block id将block存放到hash map</td></tr> <tr> <td>3</td><td>MemoryStore对象计算block所需内存大小</td></tr> <tr> <td>4</td><td>MemoryStore对象调用ensureFreeSpace()</td></tr> <tr> <td>5</td><td>IF block所需内存大小小于可用内存大小 THEN</td></tr> <tr> <td>6</td><td>将block 存放到内存中</td></tr> <tr> <td>7</td><td>ELSEIF block所需内存大小大于可用内存大小 THEN</td></tr> <tr> <td>8</td><td>通过调用dropFromMemory()将block写入文件中</td></tr> <tr> <td>9</td><td>ENDIF</td></tr> <tr> <td>10</td><td>ELSEIF 请求为MemoryStore数据读取 THEN</td></tr> <tr> <td>11</td><td>MemoryStore对象利用block id计算对应的value</td></tr> <tr> <td>12</td><td>MemoryStore对象从hash map中读取对应的block</td></tr> <tr> <td>13</td><td>ENDIF</td></tr> <tr> <td>Postcondition</td><td>系统空闲.</td></tr> </table>	Steps		1	IF 请求为MemoryStore数据存储 THEN	2	MemoryStore对象根据block id将block存放到hash map	3	MemoryStore对象计算block所需内存大小	4	MemoryStore对象调用ensureFreeSpace()	5	IF block所需内存大小小于可用内存大小 THEN	6	将block 存放到内存中	7	ELSEIF block所需内存大小大于可用内存大小 THEN	8	通过调用dropFromMemory()将block写入文件中	9	ENDIF	10	ELSEIF 请求为MemoryStore数据读取 THEN	11	MemoryStore对象利用block id计算对应的value	12	MemoryStore对象从hash map中读取对应的block	13	ENDIF	Postcondition	系统空闲.
Steps																															
1	IF 请求为MemoryStore数据存储 THEN																														
2	MemoryStore对象根据block id将block存放到hash map																														
3	MemoryStore对象计算block所需内存大小																														
4	MemoryStore对象调用ensureFreeSpace()																														
5	IF block所需内存大小小于可用内存大小 THEN																														
6	将block 存放到内存中																														
7	ELSEIF block所需内存大小大于可用内存大小 THEN																														
8	通过调用dropFromMemory()将block写入文件中																														
9	ENDIF																														
10	ELSEIF 请求为MemoryStore数据读取 THEN																														
11	MemoryStore对象利用block id计算对应的value																														
12	MemoryStore对象从hash map中读取对应的block																														
13	ENDIF																														
Postcondition	系统空闲.																														

图 3.2 新增 Storage 模块 RUCM-- MemoryStore 存取 block

3.1.3 《需求分析规格说明书 V3.0》变更

输入版本：3.0

变更产出版本：4.0

变更管理依据：《问题清单——第四次整理》、交流互动。

变更内容如下表所示：

表 4：《需求分析规格说明书 V3.0》变更过程

变更对象	变更原因描述	变更策略	具体实施
需求规.。格说明书 V3.0	实验重点发生调整	接受	对 1.4 分析过程的修改，修改了分析和研究方向。

3.2 《软件开发计划书》版本变更

表 5- 《软件开发计划书》版本变更

版本	产出时间	主要编制人	版本说明
1.0	2016.03.24	孟翰、苏若、李璇、孙敏芳	初始版本
2.0	2016.04.04	孟翰、苏若、李璇、孙敏芳	V2.0

《软件开发计划书 V1.0》变更：

输入版本：1.0

变更产出版本：2.0

变更管理依据：《问题清单——第一次整理》、交流互动。

变更内容如下表所示：

表 6- 《软件开发计划书 V1.0》变更过程

变更对象	变更原因描述	变更策略	具体实施
软件开发计划书 V1.0-项目初衷	分析项目缺陷中存在前后矛盾的情况，可能是查阅资料不一致造成。	接受	统一规范，重新修正
软件开发计划书 V1.0-标准、条约与规定	文档涉及多余的标准	接受	删除计划书中不涉及的标准
软件开发计划书 V1.0-工作内容	工作内容中没有涉及到课程的八次实验	接受	加入实验内容作为工作内容

软件开发计划书 V1.0-项目模块	类似”分析 spark 的优势与不足“等内容不应出现在模块中。	接受	对模块重新定义，找出研究的主要模块内容
-------------------	---------------------------------	----	---------------------

3.3 《项目计划》版本变更

表 7- 《项目计划》版本变更

版本	产出时间	主要编制人	版本说明
1.0	2016.03.24	孟翰、苏若、李璇、孙敏芳	初始版本
2.0	2016.04.04	孟翰、苏若、李璇、孙敏芳	V2.0
3.0	2016.04.12	孟翰、苏若、李璇、孙敏芳	V3.0
4.0	2016.05.03	孟翰、苏若、李璇、孙敏芳	V4.0

1、《项目计划》变更链：《项目计划 V1.0》——>《项目计划 V2.0》——>《项目计划 V3.0》——>《项目计划 V4.0》。

2、影响变更的因素：

a、对 Microsoft Project 工具的使用不够熟练，导致计划中存在“资源”分配出错的问题。

b、对实验过程认识不够透彻，随着时间的进行，会对后续子任务的分配计划进行不断调整和优化。

3.4 《测试需求规格说明书》版本变更

3.4.1 《测试需求规格说明书 V1.0》变更

输入版本：1.0

变更产出版本：2.0

变更管理依据：《测试问题报告——第 11 周》、交流互动。

变更内容如下表所示：

表 8- 《测试需求规格说明书 V1.0》变更过程

变更对象	变更原因描述	变更策略	具体实施
文档格式——目录	文档没有目录	接受	在第三页正文前添加目录
文档格式——引用文档	参考资料没有需求规格说明	接受	在“1.5 引用文档”一节添加需求规格说

	书且格式不够规范		明书，并对格式加以更正。
文档格式——表命名	图表名应该比正文字体小一号	接受	更正“2 测试方法概述”中涉及的图的命名文字为宋体 5 号；更正“4 测试用例”中涉及的表格的命名文字为宋体 5 号。
文档内容——测试目的模块	测试目的那一部分现在写的是测试文档的目的，而应该写测试的目的	部分调整，介绍编写测试文档的目的是有必要的，可以阐述我们编写该文档的初衷，并以此目标来规范文档编写过程。	将“1.4 目的”中的内容改为“1.4.1 文档目的”和“1.4.2 测试目的”两部分内容。
测试用例	功能需求测试用例较少	接受	在“4.1 节功能性需求测试”一节添加两个功能性测试用例
测试用例	测试准备的部分，建议简要说明一下测试数据的具体信息，以及为什么选择这些数据。	接受	在“3.2 测试数据”一节的添加对 Movielens 数据集及选择该数据集的理由的描述。
测试用例	“软硬件环境的准备配置情况”相关表格信息应该补充完整	接受	对“3.1 软硬件环境的准备配置情况”中 Linux 平台下相关信息补充完整。
测试用例	需求和测试用例之间的追踪关系不明确 测试用例的追踪可以更丰富一些 建议与需求文档相对应	适当调整	a. 在“4 测试用例”中对表格 1-1，表格 2-1~表格 2-4 中添加被测对象在需求规格说明书中对应的模块说明； b. 在“5 测试用例追踪”中添加测试需求项、测试用例项和需求规格说明书三者的追踪关系列表。
测试用例	测试完成的标准中，“结合某个测试阶段中单位时间查	修改并回应，这一完成标准属于理想测试状态下，测试周期较长的情况。通过记录测试阶段	在“3.3 测试策略”中对该判断标准进行阐述，并标明属于理想测试状态下的参考依

	出错误的数量的曲线信息，确定是否进入下一测试阶段。”表意模糊，望详悉。	A在固定时间间隔下发现问题的数量来判断测试阶段A已没有发现新问题的潜力，此时可以选择进入下一测试阶段	据，适合于测试周期较长的情况。
--	-------------------------------------	--	-----------------

3.4.2 《测试需求规格说明书 V2.0》变更

输入版本：2.0

变更产出版本：3.0

变更管理依据：《测试问题报告——第12周》、交流互动。

变更内容如下表所示：

表 9-《测试需求规格说明书 V2.0》变更过程

变更对象	变更原因描述	变更策略	具体实施
测试用例	既然提到了“性能测试（负载测试+压力测试）也是本文选用的测试方法之一”，还是建议增加单独的负载测试和压力测试测试用例，逻辑上比较说得通（或者可以修改上述说法）。	部分调整和存疑。比较疑惑的是，拿黑盒测试举例，黑盒测试亦有多种测试方法组成，若同性能测试，那么在测试用例描述时，是否应细化到黑盒测试的某个方法？在本组文档中，按理解，只能将“性能测试（负载测试+压力测试）也是本次选用的测试方法之一”改为“性能测试也是本次选用的测试方法之一”。	“2 测试方法概述”中，将“性能测试（负载+压力测试）也是本次选用的测试方法之一”改为“性能测试也是本次选用的测试方法之一”。
测试用例	在一些测试用例中只包含了正常分支，没有异常分支。	接受	在“4 测试用例”中对存在该问题的用例进行修改调整