数据共享平台

架构设计

贾中进

[www.abbkit.com](http://www.abbkit.com)

[www.ainobug.com](http://www.ainobug.com)

# 目录

[目录 1](#_Toc4057)

[1 整体架构 1](#_Toc4492)

[2 平台规划 2](#_Toc5724)

[3 数据存储中心 3](#_Toc5985)

[4 数据分析中心 3](#_Toc7782)

[5 数据交换平台 4](#_Toc455)

[6 数据服务平台（开放平台） 5](#_Toc27209)

[7 数据可视化 6](#_Toc17666)

[8 信息审计平台 6](#_Toc21384)

[9 全链路监控平台 6](#_Toc31128)

[10 LEMON数据存储平台 7](#_Toc18804)

[10.1 统一数据模型 7](#_Toc12007)

[10.1.1 插入数据结构 8](#_Toc28132)

[10.1.2 查询数据结构 8](#_Toc22431)

[10.1.3 文件的存储和查询 8](#_Toc2125)

[10.1.4 元数据Annotation 8](#_Toc18128)

[10.1.5 RDBMS数据源SQL查询结构 8](#_Toc2830)

[10.2 HBASE存储和查询 8](#_Toc10927)

[10.2.1 Batch存储 8](#_Toc942)

[10.2.1.1 HBasePut接口工厂类 8](#_Toc6719)

[10.2.1.2 HBase存储接口定义 8](#_Toc22096)

[10.2.1.3 HBase存储模型 8](#_Toc13695)

[10.2.2 查询 8](#_Toc26045)

[10.2.2.1 分页查询 8](#_Toc5071)

[10.2.2.2 Timeline查询 9](#_Toc5287)

[10.3 分布式数据同步 9](#_Toc23203)

[10.3.1 同步策略 9](#_Toc12819)

[10.3.1.1 Master同步任务产生策略 9](#_Toc26311)

[10.3.1.2 Worker执行同步任务策略 9](#_Toc12815)

[10.4 时序索引(Timeline) 9](#_Toc1211)

[10.5 RDBMS数据源统一管理 10](#_Toc23838)

[10.5.1 数据源上架 10](#_Toc22767)

[10.5.2 数据源连接/关闭 10](#_Toc13312)

[10.5.3 数据源下架 10](#_Toc24299)

[10.5.4 数据源管理UI 10](#_Toc29379)

[10.6 MR环境支持整合 10](#_Toc24227)

[10.6.1 MR On HBase 10](#_Toc13625)

[10.6.1.1 Mapper抽象 10](#_Toc29067)

[10.6.1.2 Job抽象 11](#_Toc4168)

[10.6.2 Hadoop Shell 12](#_Toc15653)

[10.6.3 LEMON MR Job UI 14](#_Toc20156)

[10.7 Hive元数据仓库整合 14](#_Toc30495)

[10.8 Hive On Spark 数据分析 15](#_Toc13682)

[10.8.1 分页 15](#_Toc10040)

[10.8.2 计数 15](#_Toc4634)

[10.9 系统性能监控 16](#_Toc32137)

[10.9.1 TPS监控 16](#_Toc12522)

[10.9.2 内存、CPU监控 16](#_Toc2026)

[10.9.3 表同步监控 16](#_Toc30960)

[10.9.4 表访问量监控 16](#_Toc10109)

[10.9.5 LEMON-模块内部监控 16](#_Toc11237)

[10.9.6 客户端连接监控 16](#_Toc28593)

[10.9.7 客户端访问量监控 17](#_Toc28082)

[10.10 LEMON Client 17](#_Toc8758)

[10.10.1 新增数据 17](#_Toc6404)

[10.10.2 更新数据 17](#_Toc22457)

[10.10.3 删除数据 17](#_Toc20903)

[10.10.4 查询数据 17](#_Toc9063)

[10.10.4.1 单实体查询 17](#_Toc2681)

[10.10.4.2 多条件查询 17](#_Toc12157)

[10.10.5 RDBMS统一数据源查询 17](#_Toc9269)

[10.10.5.1 即时查询 18](#_Toc885)

[10.10.5.2 延迟查询 18](#_Toc13164)

[10.10.6 LEMON元数据管理接口 18](#_Toc26773)

[10.10.6.1 创建表 18](#_Toc12127)

[10.10.6.2 删除表 18](#_Toc18403)

[10.10.6.3 新增/删除表字段 18](#_Toc6301)

[10.10.6.4 表信息获取 18](#_Toc29902)

[10.10.6.5 表字段信息获取 18](#_Toc23948)

[10.10.6.6 Timeline信息获取 18](#_Toc12814)

[10.10.7 File操作 19](#_Toc13386)

[10.10.7.1 文件新增 19](#_Toc27264)

[10.10.7.2 文件下载 19](#_Toc7306)

[10.10.7.3 文件删除 19](#_Toc9754)

[10.11 LEMON Streaming 19](#_Toc4435)

为适应大数据时代，数据量庞大、数据业务处理繁杂，整体方案以多库融合、数据分析、数据服务为导向，建设能够进行高效的数据管理，数据分析的平台。支撑引导数据价值的发掘、应用。数据种类繁多，一般分为结构化数据和非结构化数据。两种数据结构还是存在比较大的差异的。

首先，从数据价值上来看，结构化数据有着更明显的价值，而且开发利用便捷、成本低。非结构化数据由于数据结构差异大、数据价值隐藏深，需要特殊的数据清洗ETL之后才能利用起来，开发利用成本高。

其次，两种数据的存储，分析接口不同，需要数据分析人员熟悉多种数据分析工具，人员要求高。

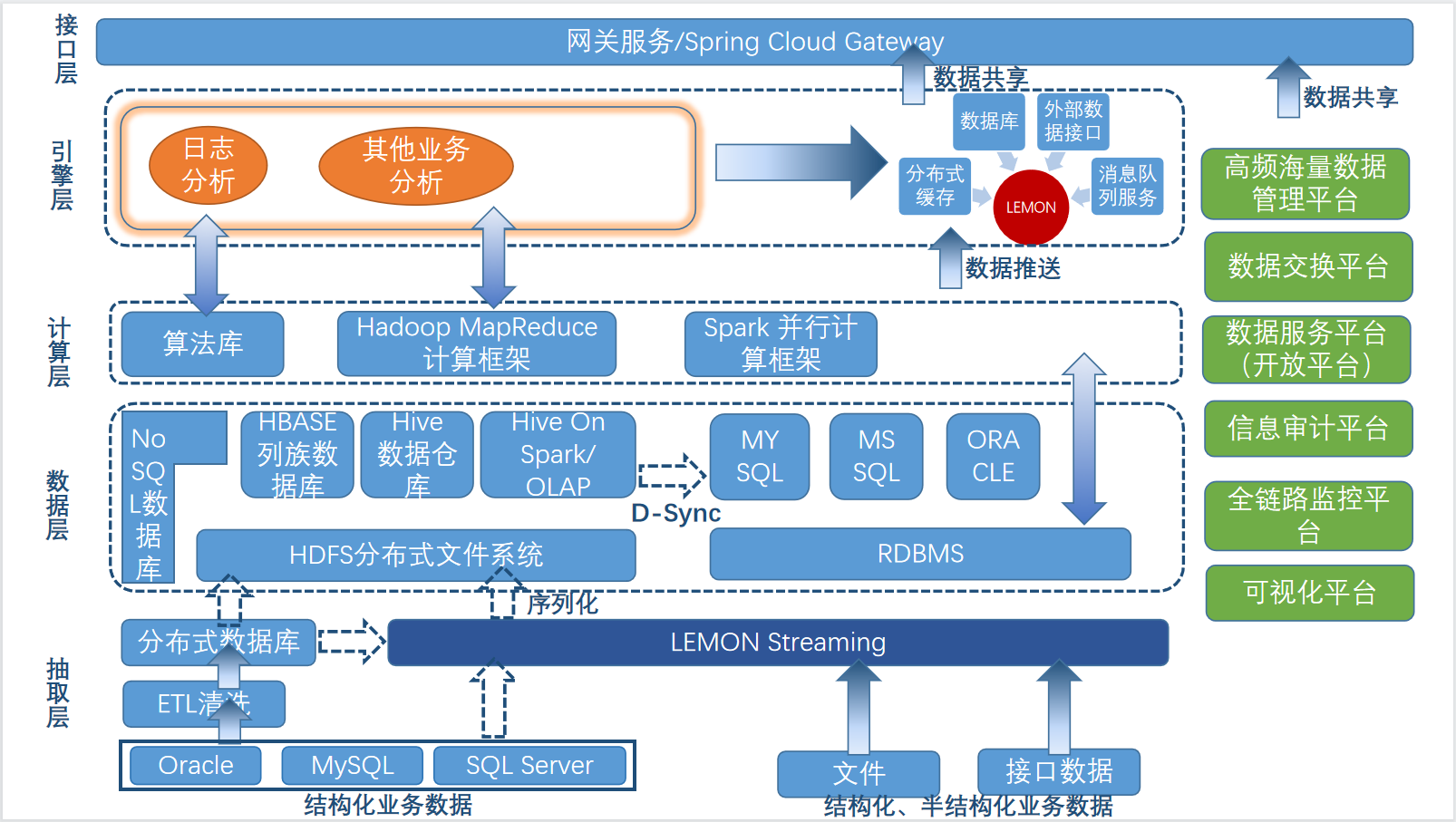
第三，为支撑多种存储、分析技术，导致软件环境复杂，运维困难。为此，通过透明化处理结构化和非结构化数据结构，建设统一的数据存储、分析平台尤为重要。为此数据共享平台通过构建于Hadoop大数据、关系型数据库（RDBMS）和NOSQL之上，对两种数据结构提供统一的数据存储、数据访问、数据分析。

# 整体架构

随着微服务、大数据处理等技术的成熟，为应对在横向扩展和大数据处理方面面临的挑战，平台着重关注如下两方面

* 计算能力
* 存储能力

计算能力定义为数据流的计算处理，通过能够横向高扩展的微服务架构和大数据分布式处理来快速的在计算能力上按需线性扩展；存储能力定义为海量数据的存储，利用分布式多备份基础存储架构，为海量的数据提供高伸缩、高响应的存储。



通过建立多层次的数据流转架构，应对数据在不同生命周期阶段在技术、业务上面对的挑战。层次化架构中每一层都聚焦于特定阶段的处理，在处理能力上都能够线性扩展。

1. 抽取层

从数据源中抽取数据经过ETL完成数据的抽取。由于数据源种类繁多，通过一些数据采集工具，比如Flume，可以规范化采集的过程，提高采集效率。

1. 数据层

提供数据存储，统一大数据分布式存储和关系型数据库存储，通过线性扩展的方式支撑海量数据的存储。

1. 计算层

通过建立Map Reduce和Spark并行计算框架，提供线性伸缩的计算环境。多种算法库也能够优化计算，为计算能力的提升增速。

1. 引擎层

提供面向多种主题的分析能力，比如日志分析、其他业务分析等。

1. 接口层

在安全网关下提供统一数据服务接口。

# 平台规划

平台建设以数据存储，数据分析为基础，整合多方资源，提供标准数据分析，提供统一标准数据服务。为此，建设二个中心、六大平台。

1. 数据存储中心

通过引入数据主题的概念，建设多个主题库，

1. 数据分析中心

基于数据存储中心的数据，提供特定领域的业务服务

1. 六大平台

* 高频海量数据管理平台

提供对LEMON整体的管理，包括数据、计算环境，比如Hadoop、HBase、Hive、Mysql等

* 数据交换平台

在大数据分析技术之上，合并传统的小批量数据分析，提供统一的数据转换。

* 数据服务平台（开放平台）

通过提供平台认证，提供第三方数据服务。

* 信息审计平台

对数据服务平台、数据交换平台处理的数据提供信息审计，保证整个平台中的数据的安全性，比如公民身份证涉及哪些服务的统计等。

* 全链路监控平台

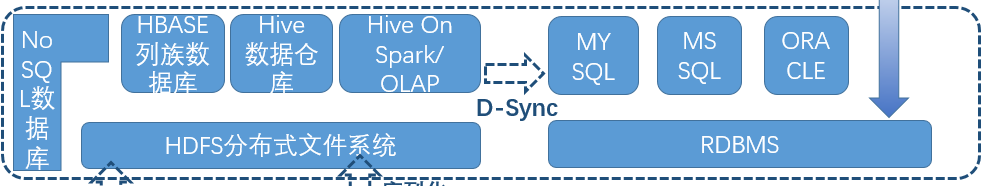
包括平台硬件、软件资源监控，平台接口监控、数据流转链路监控等。

* 可视化平台

通过一些可视化技术，展示平台数据信息，比如数据量、数据来源，数据主题库信息等。

# 数据存储中心

以大数据存储***Hadoop+HBase***为主，关系型数据库存储为辅。通过内部的***分布式数据同步***同步热数据到关系型数据库中。



# 数据分析中心

存储中心的数据是静态的，没有生命力的，通过数据分析技术，能让数据插上翅膀，变的更有表现力和生命力。数据分析就是数据开放的窗口，能够一窥数据的价值。由于数据量大，传统的数据分析已不能应对，因此需要搭建基础的大数据分析平台。引入Hadoop生态以及流数据处理技术，在以下两方面支撑数据分析。

* 在线处理

通过引入流处理技术，在线处理需要的数据分析，需要近实时的分析。

* 离线处理

通过Hadoop生态技术，能够在大数据量下分布式高效率的处理某个分析任务。



# 数据交换平台

平台包含标准数据定义与数据转换（适配）服务。标准数据定义指的是数据平台上面的所有标准服务需要的数据结构类型，数据转换服务定义了平台标准数据定义与二级平台或者第三方服务商产生的数据的映射关系。平台要能支持此关系的动态增减，系统以插件化、模块化开发。



# 数据服务平台（开放平台）

数据平台中心的数据可以通过提供数据服务来产生更多的价值。一方面这些数据能够为平台自己的各种分析提供服务，另一方面平台也能通过一些授权提供给第三方。数据服务平台登记统一的数据服务，第三方通过授权，能够接入平台提供的数据服务。平台提供一些功能，比如数据服务统计、数据服务管理、数据服务授权等。



# 数据可视化

通过一些可视化技术，展示平台数据信息，比如数据量、数据来源，数据主题库信息等。模块化设计各可视化系统，各个模块子系统通过服务自治的方式参与系统整合，选择微服务架构下的强内聚的服务接入方式。



# 信息审计平台

暂无设计

# 全链路监控平台

暂无设计

# LEMON数据存储平台

建立统一的数据管理平台，解决跨HBASE和RDBMS数据存储系统时访问不一致，开发接口多样的问题。引入数据分层的理念，在冷热数据处理上采取不同的策略，综合利用大数据存储和RDBMS数据存储的优势。通过Master Slave的架构设计，平台在高可用、线性扩展上继承了Hadoop生态的优点。

参考**<http://abbkit.com/lemon.html>**

## 统一数据模型

由于分布式存储和RDBMS存储的数据结构差异大，查询模型不一致，往往导致复杂的接口调用；分布式存储一般基于Key-Value存储模型，没有数据的范式定义，在主题库建设中需要元数据来表现数据。因此，通过在分布式存储的基础上提供动态伸缩的元数据定义，提供了与RDBMS相似的范式数据定义，同时兼顾了大数据存储的高伸缩性。

**定义如下**

|  |
| --- |
| 数据结构：JSON  数据类型：   * + - 1. string 字符类型       2. int 整型       3. long 长整型       4. double 浮点型       5. boolean布尔型       6. date 日期类型，值为长整型（long）   网络协议：HTTP/TCP |

### 插入数据结构

### 查询数据结构

### 文件的存储和查询

### 元数据Annotation

### RDBMS数据源SQL查询结构

## HBASE存储和查询

|  |
| --- |
| *Tips：*  *如果想保证数据的原子性，可以一条一条的插入。* |

HBase是一个列式存储系统，不满足传统数据处理上的ACID特性，一般不能提供多行数据处理的原子性、事务性。在操作性能上，平台提供了Batch操作，能够大大降低网络流量，但是在事务上不能保证批量插入的原子性、一致性。

### Batch存储

#### HBasePut接口工厂类

#### HBase存储接口定义

#### HBase存储模型

### 查询

|  |
| --- |
| *Tips：*  *如果一张表的数据分布在多个region服务器上，分页查询是合并多个region的分页结果返回给客户端。* |

#### 分页查询

在查询模型里面设置参数“***LEMON\_page\_count***”的值为每页查询的条数，类***DefaultHBaseQuery***解析参数并执行查询。

#### Timeline查询

## 分布式数据同步

平台会把最近的数据同步到RDBMS数据库中，实现数据的分层。而且可以利用RDBMS数据库提供的计算能力，比如聚合、计数、排序等，对小量的热数据做数据上的在线分析。平台通过Master分发表同步任务给相应的Worker来执行数据的同步。

### 同步策略

#### Master同步任务产生策略

Master产生同步任务时，有下面多种策略

1. **滚动同步策略**

采取小片段的增量同步策略，能够分散数据同步的压力，避免大量数据同步时数据积压的问题，能够稳定系统的运行，时间片段可以通过配置文件配置（列出部分）

1. **手动同步策略**

支持用户在LEMON管理页面上自定义时间段的同步需求。

#### Worker执行同步任务策略

Worker执行同步任务时，有下面两种策略。

1. **异步策略**

异步策略使用X-STREAM模块通过PUB-SUB模式异步同步数据，有着高吞吐量，但是不能保证数据同步一致性，因为同步完成通知事件是在加载完数据之后就产生了。

1. **强制同步策略**

强制同步策略利用SQL**批插入**时性能比较高的特性，在一个事务中执行全删全增操作，在同步到RDBMS数据库之后才产生同步完成事件，保证了数据同步一致性。

## 时序索引(Timeline)

通过HBase的协处理器（Coprocessor），记录数据的PUT操作记录，生成面向时间线的日志。

## RDBMS数据源统一管理

|  |
| --- |
| *Tips：*  *因为每一个Worker节点都会建立到数据库的连接，想想如果Worker节点很多，会有如下问题*   1. *连接数过多* 2. *如何释放连接* 3. *如何管理连接*   *AHA，当然要用连接池技术啦* |

***lemon-meta-datasource***表存储数据源信息，包含数据库名称、地址、驱动、用户名、密码等。数据源连接的建立是分散在所有的Worker中，每一个Worker节点都会建立到新建数据源的连接，Worker节点越多，相应数据源的连接数也越多，需要通过**连接池技术**动态关闭闲置的连接。

### 数据源上架

### 数据源连接/关闭

Worker通过***连接池技术***同步打开或者关闭表中的数据库连接，默认的连接数为10。数据源可连接/关闭检测周期为1分钟。

### 数据源下架

下架集群中指定的数据源，Worker通过定时任务周期性检测连接是否下架，如果下架了就关闭连接。

### 数据源管理UI

## MR环境支持整合

基于MapReduce计算架构，可以高效处理HBase中的数据，通过在LEMON环境中整合MR，可以方便地管理基于HBase的MR应用。

### MR On HBase

#### Mapper抽象

继承***org.apache.hadoop.hbase.mapreduce.TableMapper***，提供便捷的访问LEMON中的表数据。

|  |
| --- |
| 1. /\*\* 2. \* Created by J on 2019/9/4. 3. \*/ 4. **public** **abstract** **class** LEMONTableMapper<KEYOUT, VALUEOUT>  **extends** TableMapper<KEYOUT, VALUEOUT> { 6. **private** String tableName; 8. **private** RowValueTransform rowValueTransform= bytes -> Value.of(bytes,String.**class**); 10. **private** ColumnValueTransform columnValueTransform; 12. @Override 13. **protected** **void** setup(Context context) **throws** IOException, InterruptedException { 14. **super**.setup(context); 15. tableName=context.getConfiguration().get(TableInputFormat.INPUT\_TABLE,""); 16. columnValueTransform=**new** ColumnValueTransformFromRemoteTable(tableName, DefaultClients.option()).build(); 18. }  21. **protected** **final** String tableName(){ 22. **return** tableName; 23. } |

#### Job抽象

规范化Hadoop Shell指令中参数，连接LEMON MR管理模块。

|  |
| --- |
| 1. /\*\* 2. \* Created by J on 2019/9/5. 3. \*/ 4. **public** **abstract** **class** LEMONJob **extends** Configured **implements** Tool ,Closeable { 5. **protected** **abstract** **int** run0(String[] args) **throws** Exception;  8. **protected** **final** **void** log(String desc){ 9. LOGGER.info(desc); 10. } 12. **private** **void** submit0(Job job,String[] args) **throws** IOException, ClassNotFoundException, InterruptedException{ 13. LEMONJobContext LEMONJobContext=LEMONJobContext.option(); 15. addFileToClassPath(job,LEMONJobContext.getLEMONMapReduceConfigPathFile()); 16. addFileToClassPath(job,LEMONJobContext.getLEMONMapReduceClassPathFile()); 18. addFileToClassPath(job,LEMONJobContext.getLEMONCustomMapReduceConfigPathFile()); 19. addFileToClassPath(job,LEMONJobContext.getLEMONCustomMapReduceClassPathFile()); 21. job.getConfiguration().set(MRJobConfig.MAPREDUCE\_JOB\_USER\_CLASSPATH\_FIRST,"true"); 22. job.getConfiguration().set("mapreduce.task.classpath.user.precedence","true"); 24. job.submit(); 26. String hadoopJobIdentifier=job.getJobID().toString(); 27. LEMONJobContext.setHadoopJobIdentifier(hadoopJobIdentifier); 28. }  31. **protected** **final** **void** submitLocal(Job job,String[] args) **throws** IOException, ClassNotFoundException, InterruptedException { 33. job.getConfiguration().set(MRConfig.FRAMEWORK\_NAME,MRConfig.LOCAL\_FRAMEWORK\_NAME); 34. job.waitForCompletion(**true**); 36. } 37. } |

### Hadoop Shell

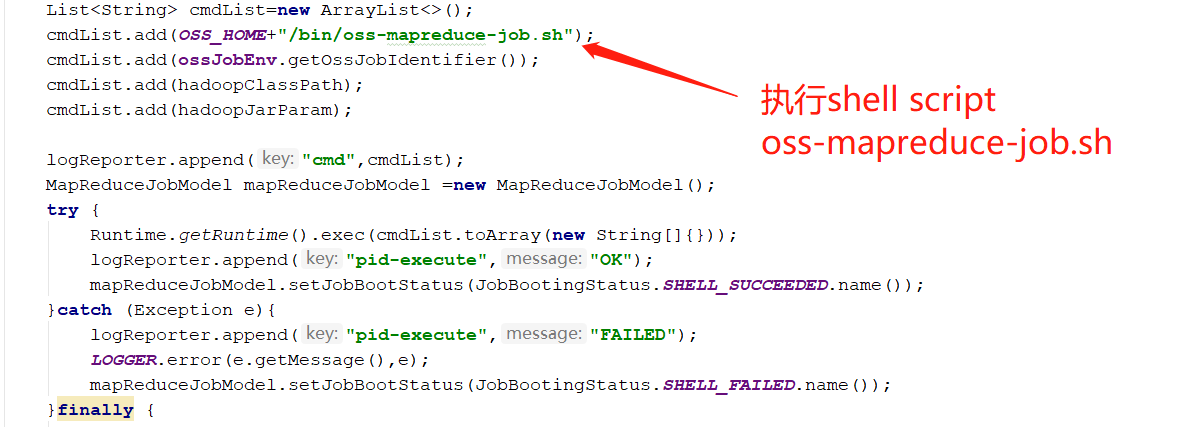
调用Hadoop Shell发起Hadoop Job任务，规范shell调用接口数据结构。

|  |  |  |
| --- | --- | --- |
| 参数 | 备注 | 默认值 |
| --LEMONJobIdentifier | LEMON MR Job标识 |  |
| --LEMONMapReduceClassPathFile | 全局配置Job需要的Jar | lemon.mapreduce.mapReduceClassPathFile.lib |
| --LEMONMapReduceConfigPathFile | 全局配置Job需要的配置文件 | lemon.mapreduce.mapReduceClassPathFile.config |
| --LEMONCustomMapReduceClassPathFile | 自定义配置Job需要的Jar |  |
| --LEMONCustomMapReduceConfigPathFile | 自定义配置Job需要的配置文件 |  |
| --LEMONMapReduceLogFile | 配置shell日志文件 |  |
| --LEMONTableName | 配置需要处理的LEMON表 |  |

**相关类**

* com.abbkit.lemon.server.mapreduce.LEMONJobBoot： lemon-mapreduce-job.sh shell脚本引导类
* com.abbkit.lemon.server.mapreduce.LEMONJobEnv：配置shell脚本的入口参数

调用shell脚本的代码片段参考如下



**lemon-mapreduce-job.sh**

|  |
| --- |
| #!/usr/bin/env bash  # Licensed to the Apache Software Foundation (ASF) under one or more  # contributor license agreements. See the NOTICE file distributed with  # this work for additional information regarding copyright ownership.  # The ASF licenses this file to You under the Apache License, Version 2.0  # (the "License"); you may not use this file except in compliance with  # the License. You may obtain a copy of the License at  #  # http://www.apache.org/licenses/LICENSE-2.0  #  # Unless required by applicable law or agreed to in writing, software  # distributed under the License is distributed on an "AS IS" BASIS,  # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  # See the License for the specific language governing permissions and  # limitations under the License.  #  #ps -ef| grep RunJar | grep -v grep | awk '{print $2}' | xargs kill -9  #  # export HADOOP\_HOME=/work/hadoop-2.7.5  # export LEMON\_HOME=/work/LEMON/cpp-LEMON-server  #  # (  # 1.检查shell文件是否有执行权限, chmod 777 \*.sh  # 2.检查HADOOP\_HOME和LEMON\_HOME是否配置  # 3.MASTER\WORKER 必须在环境变量配置完之后启动  # )  #  LEMONJobIdentifier=$1  if [ "$LEMON\_MR\_LOG\_TEMP" = "" ]; then  export LEMON\_MR\_LOG\_TEMP=/tmp  fi  [ -w "$LEMON\_MR\_LOG\_TEMP" ] || mkdir -p "$LEMON\_MR\_LOG\_TEMP"  shellExeMarkLog=$LEMON\_MR\_LOG\_TEMP/LEMON-mapreduce-shell-mark.log  if [ ! -f "$shellExeMarkLog" ] ;then  echo 'a' > $shellExeMarkLog  fi  echo `date` $@ >> $shellExeMarkLog  shellLogFile=${LEMON\_MR\_LOG\_TEMP}/${LEMONJobIdentifier}.log  echo > $shellLogFile  export HADOOP\_USER\_CLASSPATH\_FIRST=true  #export HADOOP\_CLASSPATH=/work/LEMON/LEMON-mr/config:/work/LEMON/LEMON-mr/lib/\*:/work/LEMON/LEMON-mr/custom//cpp-LEMON/mr-rowcount-lib/\*  export HADOOP\_CLASSPATH=$2  echo "hadoop-class-path contains: $HADOOP\_CLASSPATH" >> $shellLogFile  #/work/LEMON/LEMON-mr/custom///cpp-LEMON/mr-rowcount-lib/cpp-LEMON-mr-rowcount-3.0.1.jar me.libme.LEMON.mapreduce.rowcount.RowCounterJob --LEMONMapReduceClassPathFile /cpp-LEMON/mr-lib --LEMONTableName LEMON-meta-sync-table --LEMONJobIdentifier LEMON-meta-sync-table-6e23b91c --LEMONCustomMapReduceClassPathFile /cpp-LEMON/mr-rowcount-lib/  if [ "$HADOOP\_HOME" = "" ]; then  echo "hadoop home is missing : $HADOOP\_HOME" >> $shellLogFile  echo "exit!!!" >> $shellLogFile  exit 1  fi  HADOOP\_EXE=$HADOOP\_HOME/bin/hadoop  echo "${HADOOP\_HOME}/bin/hadoop jar $3 " >> $shellLogFile  NUM=`ps aux | grep -w RunJar | grep -v grep |wc -l`  #echo $NUM  #少于1，重启进程  if [ "${NUM}" -lt "1" ];then  echo "RunJar count:"$NUM",ok,execute hadoop jar" >> $shellLogFile  **${HADOOP\_HOME}/bin/hadoop jar $3 --LEMONMapReduceLogFile "file://"$shellLogFile >> $shellLogFile 2>&1 &**  **else**  echo "RunJar count:"$NUM",ok,drop executing hadoop jar ,avoid resource leak" >> $shellLogFile  fi  echo OK >> $shellLogFile |

### LEMON MR Job UI

## Hive元数据仓库整合

通过建立外部表的方式，整合LEMON HBASE中的数据，参考如下

|  |
| --- |
| CREATE **EXTERNAL TABLE** `hive\_t\_test\_user`(  `rowkey` string COMMENT '',  `age` int COMMENT '',  `birthday` bigint COMMENT '',  `leader` boolean COMMENT '',  `money` double COMMENT '',  `name` string COMMENT '',  `phone` bigint COMMENT '',  `LEMONrowtime` bigint COMMENT '')  ROW FORMAT SERDE  'org.apache.hadoop.hive.hbase.HBaseSerDe'  STORED BY  'org.apache.hadoop.hive.hbase.HBaseStorageHandler'  WITH SERDEPROPERTIES (  'hbase.columns.mapping'=':key,cf:age#b,cf:birthday#b,cf:leader#b,cf:money#b,cf:name,cf:phone#b,cf:LEMONRowTime#b',  'serialization.format'='1')  TBLPROPERTIES ('hbase.table.name'='t\_test\_user') |

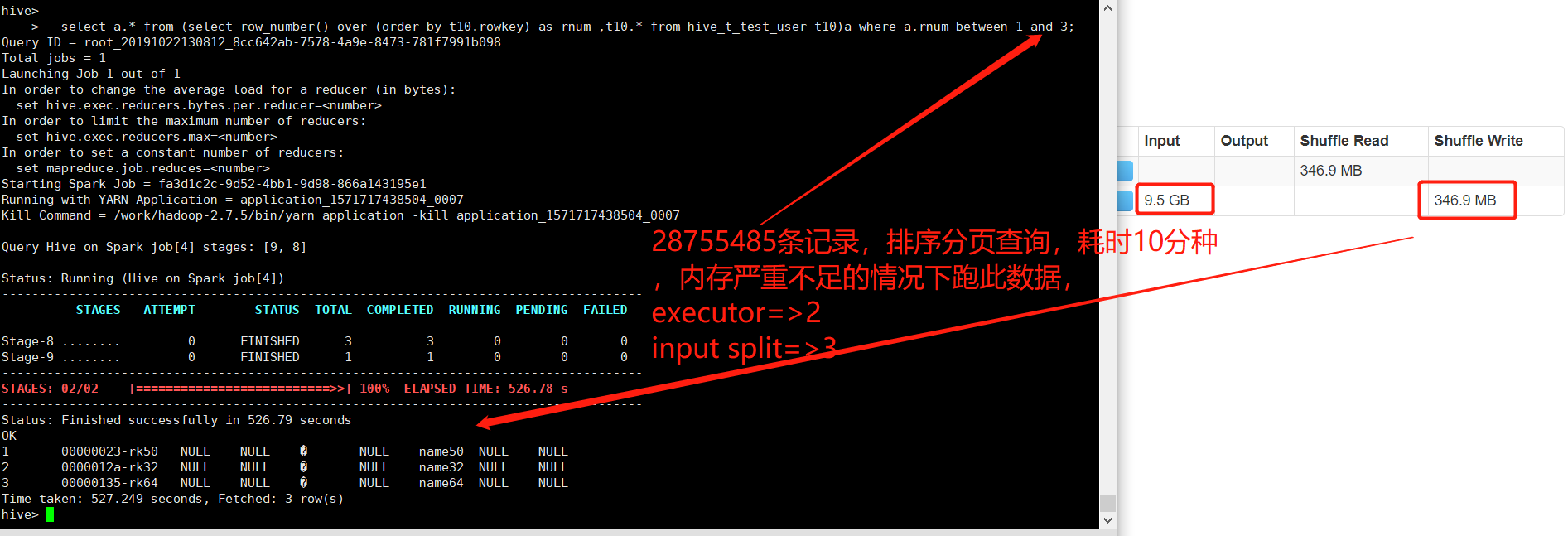
表连接建立完成之后可以执行SQL统计分析

|  |
| --- |
| 1. **SELECT** 2. a.\*, a.age 3. **FROM** 4. ( 5. **SELECT** 6. row\_number () over (**ORDER** **BY** tmp.rowkey) **AS** rnum, 7. tmp.\* 8. **FROM** 9. hive\_t\_test\_user tmp 10. **WHERE** 11. tmp. **NAME** = "name33" AND tmp.LEMONrowtime > 1571156142185 12. ) a 13. **WHERE** a.rnum BETWEEN 3000  AND 3100; |

## Hive On Spark 数据分析

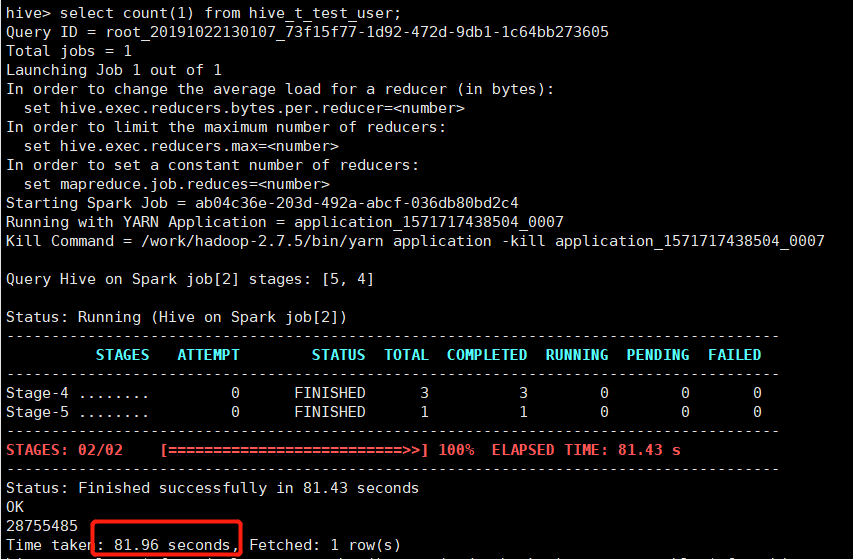
### 分页

|  |
| --- |
| 1. **SELECT** 2. a.\*, a.age 3. **FROM** 4. ( 5. **SELECT** 6. row\_number () over (**ORDER** **BY** tmp.rowkey) **AS** rnum,tmp.\* 7. **FROM** 8. hive\_t\_test\_user tmp 9. **WHERE** 10. tmp. **NAME** = "name33" AND tmp.LEMONrowtime > 1571156142185 11. ) a 12. **WHERE** a.rnum BETWEEN 3000000  AND 3000100; |



### 计数

|  |
| --- |
| 1. **select** count(1) **from** hive\_t\_test\_user; |



## 系统性能监控

### TPS监控

### 内存、CPU监控

### 表同步监控

LEMON同步HBase中的数据到RDBMS数据库中。这个页面列出同步相关的信息，上半部分列出Zookeeper上所有正在同步的表，下半部分可以根据查询条件查询历史同步信息，默认按照同步开始时间升序排序，例如查询特定的表，可以清晰看到表的同步信息。

### 表访问量监控

列出LEMON中所有表的历史访问量，Put为插入、更新、删除量，Query为查询量。

### LEMON-模块内部监控

展示LEMON JVM内部所有模块的运行时状态，包含集群中所有的节点数据。***ResourceCollector***类接收所有信息，并把信息作为JSON字节流输出。包括但不仅限于如下

* JVM环境信息
* 数据库连接池信息
* HTTP连接信息
* 缓存使用信息
* LEMON各种配置信息

### 客户端连接监控

列出当前正在连接的客户端，高亮标识那些掉线的客户端（目前心跳时间为3分种），可以有效跟踪客户端的连接情况，特别在网络复杂的环境中，可以在客户端掉线的时候发出预警通知。

### 客户端访问量监控

列出当前连接的客户的访问量，辨别访问量大的客户端。

## LEMON Client

Client通过连接到LEMON的前置NGINX进行通信，通过HTTP的方式发送统一的数据结构。客户端需要access key建立和服务端的可信任连接。

### 新增数据

### 更新数据

### 删除数据

### 查询数据

#### 单实体查询

#### 多条件查询

根据**统一查询模型**查询单表数据，数据集通过迭代器的方式返回。

1. 一般表数据查询

根据查询条件返回单表的数据集，使用方法

1. Timeline数据查询

根据查询条件返回单表的数据集，使用方法

### RDBMS统一数据源查询

由于外部数据源的查询效率差别大，有的比较快，有的比较慢，LEMON提供两种查询方式。第一种为即时查询，查询结果立即返回给客户端；第二种为延迟查询，通过提交查询任务，客户端等待一段时间之后可以收到服务端查询完成的通知。

#### 即时查询

#### 延迟查询

1. **同步等待**
2. **异步回调**

### LEMON元数据管理接口

客户端也提供了部分LEMON元数据管理接口，包含创建表、删除表、查询表元数据、查询表Timeline信息等。

#### 创建表

#### 删除表

#### 新增/删除表字段

#### 表信息获取

#### 表字段信息获取

#### Timeline信息获取

### File操作

#### 文件新增

#### 文件下载

#### 文件删除

## LEMON Streaming

[http://abbkit.com/lemon.html#item-3](http://abbkit.com/lemon.html" \l "item-3)