

Hadoop生态系统



讲师简介

简介：张老师（Jackson）

邮箱：vzyuelel@126.com

关注的技术产品及工具：



➤ 毕业于中国科技大学，目前供职于某互联网金融公司任大数据基础平台架构师及测试负责人

➤ 关注大数据分析技术及业务实践

➤ 10余年一线数据业务及技术实践经验

- 制造行业
- 咨询服务
- 互联网金融

Hadoop生态系统

1. 大数据概述及和传统数据仓库的区别
2. 大数据应用场景
3. Hadoop生态系统概述
4. Hadoop核心模块
5. 从关系型数据到HDFS
6. HDFS数据处理方式
7. NoSQL及HBase概述
8. 组件协做方式及性能监控

急速增长的数据使传统的关系型数据库无所适从，如何存储大数据、如何处理大数据，如何挖掘大数据，这是大数据时代面临的一些挑战。



在大数据处理中，毫无疑问，Hadoop已成为当下的王者技术，经过开源社区无数贡献者的强大推动，Hadoop以其显著的低成本、高可靠性、高性能等特性，成功地征服了众多大数据处理需求的商业机构和科研团体。既有像Google、Facebook、Yahoo，阿里、百度、华为等这样的知名的企业，也有数以万计的中小企业。

由中国计算机学会 (CCF) 主办的“2014中国大数据技术大会”上发布了《中国大数据技术与产业发展白皮书(2014年)》，针对2015年度大数据发展做了十大预测，部分引用如下：

- 结合智能计算的大数据分析成为热点，包括大数据与神经计算、深度学习、语义计算以及人工智能其他相关技术结合，成为大数据分析领域的热点。
- 大数据将与物联网、移动互联、云计算、社会计算、等热点技术领域相互交叉融合，产生很多综合性应用。
- 大数据多样化处理模式与软硬件基础设施逐步夯实。内存计算将继续成为提高大数据处理性能的主要手段。
- 各种可视化技术和工具提升大数据分析。
- 大数据技术课程体系建设和人才培养是需要高度关注的问题。
- 开源系统将成为大数据领域的主流技术和系统选择。

(<http://www.ccf.org.cn/sites/ccf/xwdt.jsp>)

传统数据仓库分析大和数据分析之间的差别

传统数据仓库分析技术	大数据分析技术
传统分析对已知的数据范围中好理解的数据进行分析。大多数数据仓库都有一个精致的提取、转换和加载(ETL)流程和数据库限制。进入数仓的数据是经过清洗的并符合业务的元数据。	大数据最大的优点是针对传统手段捕捉到的数据之外的非结构化数据。
传统分析是建立在关系数据模型之上的，主题之间的关系在系统内就已经被创立，而分析也在此基础上进行。	主要处理非结构化数据。比如图片、视频、移动产生的信息、RFID等
面对数据量由TB级别升至PB级，并仍在持续爆炸式增长。传统数仓分析工具在处理上遇到很多瓶颈。	大数据技术在平台支撑和计算模式上，都可以很好的适应PB级别的数据量
硬件平台的区别。在传统数据仓库分析模式下，由于数据量的迅速增加，并行数据库的规模不得不随之增大，从而导致其成本的急剧上升。	出于成本的考虑，大数据分析从设计之初就着眼于大量中低端硬件构成的大规模机群平台，实现系统的平滑扩展。

Hadoop生态系统

1. 大数据概述及和传统数据仓库的区别
2. 大数据应用场景
3. Hadoop生态系统概述
4. Hadoop核心模块
5. 从关系型数据到HDFS
6. HDFS数据展示方式
7. NoSQL及HBase概述
8. 组件协做方式及性能监控

大数据应用场景(资料来源于《大数据价值49式》) <1>



Google针对大数据的三个应用

- ▶ 谷歌意图：谷歌不仅存储了搜索结果中出现的网络连接，还会储存用户搜索关键词的行为，它能够精准地记录下人们进行搜索行为的时间、内容和方式，这些数据能够让谷歌优化广告排序，并将搜索流量转化为盈利模式。换言之，谷歌能在你意识到自己要找什么之前预测出你的意图。这种抓取、存储并对海量人机数据进行分析，然后据此进行预测的能力，就是数据驱动的产品。
- ▶ 谷歌分析：谷歌在搜索之外还有更多获取数据的途径。企业安装“谷歌分析(Google Analytics)”之类的产品来追踪访问者在其站点的足迹，而谷歌也可获得这些数据。网站还使用“谷歌广告联盟(Google AdSense)”，将来自谷歌广告客户网的广告展示在其站点，因此，谷歌不仅可以洞察自己网站上广告的展示效果，同样还可以对其他广告发布站点的展示效果一览无余。
- ▶ 谷歌趋势：既然搜索本身是网民的“意图数据库”，当然可以根据某一专题搜索量的涨跌，预测下一步的走势。谷歌趋势可以预测旅游、地产、汽车的销售。此类预测最著名的就是谷歌流感趋势，跟踪全球范围的流感等病疫传播，依据网民搜索，分析全球范围内流感等病疫的传播状况。

大数据应用场景(资料来源于《大数据价值49式》) <2>

➤ 阿里小贷和聚石塔

虽然阿里系的余额宝如日中天，但其实阿里小贷才真正体现出了大数据的价值。早在2010年阿里就已经建立了“淘宝小贷”，通过对贷款客户下游订单、上游供应商、经营信用等全方位的评估，就可以在没有见面情况下，给客户放款，这当然是对阿里平台上大数据的挖掘。



数据来源于“聚石塔” 一个大型的数据分享平台，它通过共享阿里巴巴旗下各个子公司(淘宝、天猫、支付宝等)的数据资源来创造商业价值。这款产品就是大数据团队把淘宝交易流程各个环节的数据整合互联，然后基于商业理解对信息进行分类储存和分析加工，并与决策行为连接起来所产生的效果。

大数据应用场景(资料来源于《大数据价值49式》) <3>

► 百合网的婚恋匹配



作为一家婚恋网站，百合网不仅需要经常做一些研究报告，分析注册用户的年龄、地域、学历、经济收入等数据，即便是每名注册用户小小的头像照片，这背后也大有挖掘的价值。

百合网研究规划部李琦曾经对百合网上海量注册用户的头像信息进行分析，发现那些受欢迎头像照片不仅与照片主人的长相有关，同时照片上人物的表情、脸部比例、清晰度等因素也在很大程度上决定了照片主人受欢迎的程度。例如，对于女性会员，微笑的表情、直视前方的眼神和淡淡的妆容能增加自己受欢迎的概率，而那些脸部比例占照片1/2、穿着正式、眼神直视没有多余pose的男性则更可能成为婚恋网站上的宠儿。

大数据应用场景(资料来源于《大数据价值49式》) <4>

➤ 榨菜指数

负责起草《全国促进城镇化健康发展规划(2011-2020年)》(以下简称“城镇化规划”)的国家发改委规划司官员需要精确知道人口的流动,怎么统计出这些流动人口成为难题。

榨菜,属于低质易耗品,收入增长对于榨菜的消费几乎没有影响。一般情况下,城市常住人口对于方便面和榨菜等方便食品的消费量,基本上是恒定的。销量的变化,主要由流动人口造成。

据国家发改委官员的说法,涪陵榨菜这几年在全国各地区销售份额变化,能够反映人口流动趋势,一个被称为“榨菜指数”的宏观经济指标就诞生了。研究发现,涪陵榨菜在华南地区销售份额由2007年的49%、2008年的48%、2009年的47.58%、2010年的38.50%下滑到2011年的29.99%。这个数据表明,华南地区人口流出速度非常快。他们依据“榨菜指标”,将全国分为人口流入区和人口流出区两部分,针对两个区的不同人口结构,在政策制定上将会有所不同。



Hadoop生态系统

1. 大数据概述及和传统数据仓库的区别
2. 大数据应用场景
3. Hadoop生态系统概述
4. Hadoop核心模块
5. 从关系型数据到HDFS
6. HDFS数据处理方式
7. NoSQL及HBase概述
8. 组件协做方式及性能监控

➤ Hadoop起源



Hadoop是一个分布式文件系统和并行执行环境,目的是让用户便捷地处理海量数据。

由Doug Cutting和Mike Cafarella根据Google Lab 的三篇论文 Map/Reduce 、Google BigTable、 Google File System(GFS) 的启发研究而来。Doug Cutting给这个Project起了个名字叫Hadoop。

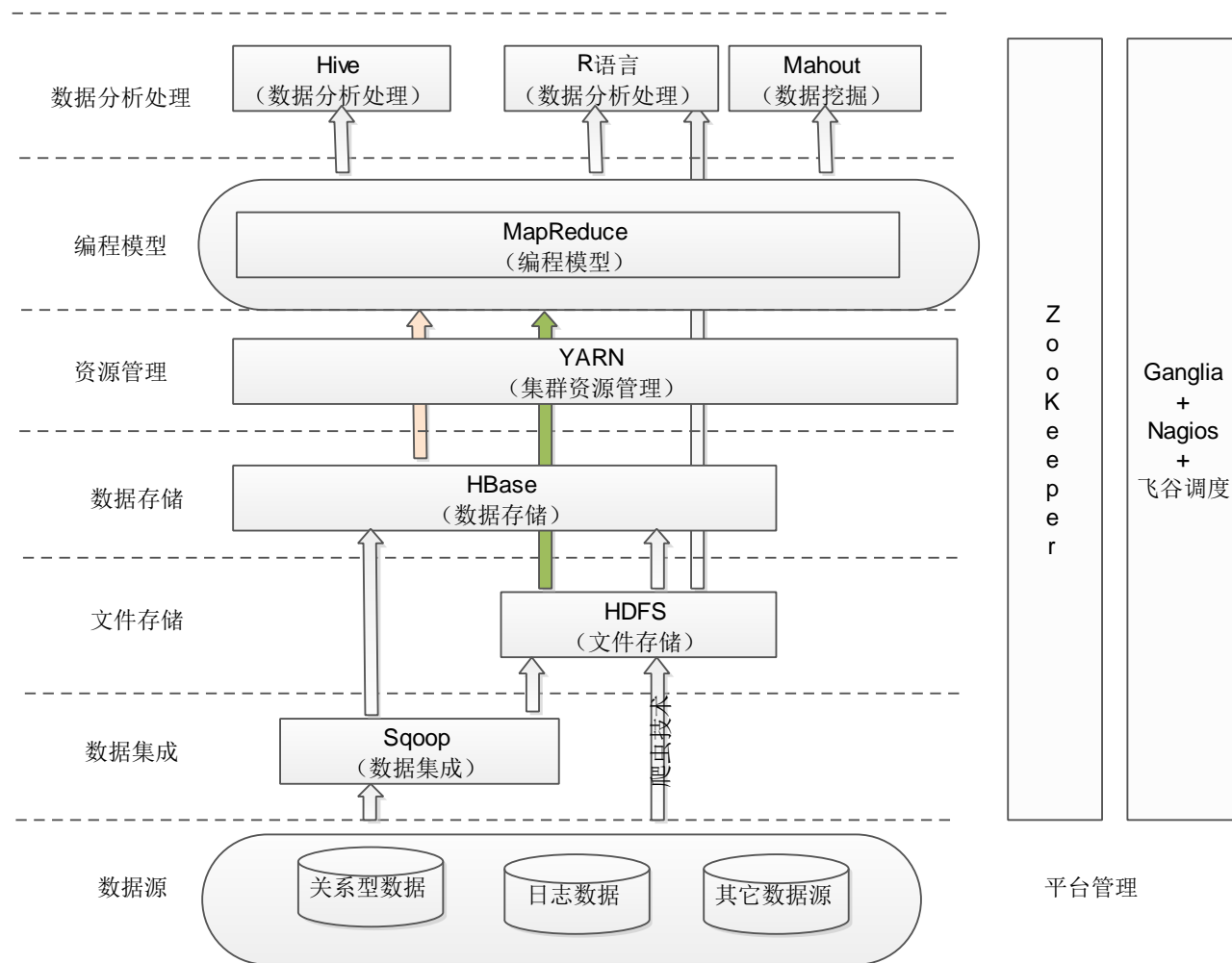
于2004年开始实施,2006年2月Apache Hadoop项目正式启动。至今为止是2.4.1, 稳定版本是1.2.1 和 yarn 的 2.4.0。

除了Apache的免费版本外,目前有很多公司开始提供基于Hadoop的商业软件、支持、服务以及培训。比较著名的有Cloudera、Hortonworks和MapR三家。

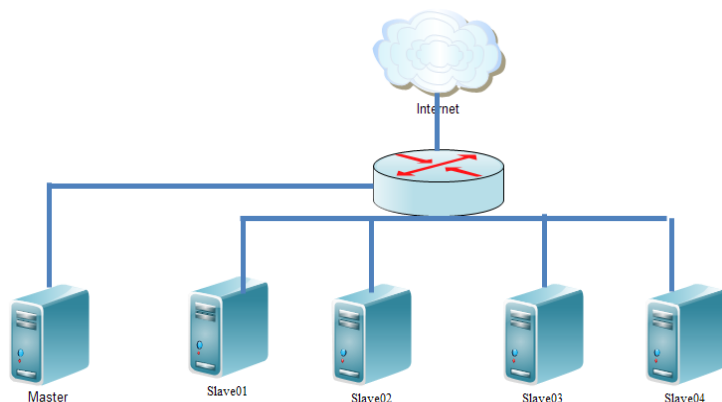
Hadoop主要有以下几个优点：

- 高可靠性。Hadoop按位存储和处理数据的能力值得人们信赖。
- 高扩展性。Hadoop是在可用的计算机集簇间分配数据并完成计算任务的，这些集簇可以方便地扩展到数以千计的节点中。
- 高效性。Hadoop能够在节点之间动态地移动数据，并保证各个节点的动态平衡，因此处理速度非常快。
- 高容错性。Hadoop能够自动保存数据的多个副本，并且能够自动将失败的任务重新分配。
- 低成本。与一体机、商用数据仓库以及数据集市相比，hadoop是开源的，项目的软件成本因此会大大降低。

➤ Hadoop生态系统组成部分



➤ Hadoop生态系统实施 -- 飞谷云(feiguyun.com)的系统架构



Master和Slave节点机器硬件配置基本相同。双CPU/8G内存/20G硬盘。
操作系统是Ubuntu。每台机器对外均有固定IP直接访问

A	B	C	D	E	F	G	H	I	J
机器名	hadoop	hbase	hive	hiveserver2	spark	ntp	zk	sqoop	ganglia
master	NameNode	HMaster	■		Master	server	■		gmond
slave01	DataNode/SecondaryNameNode	HRegionServer	■		Worker	crontab	■		gmetad
slave02	DataNode	HRegionServer	■	■	Worker	crontab	■	■	gmond
slave03	DataNode	HRegionServer	■		Worker	crontab	■		gmond
slave04	DataNode	HRegionServer			Worker	crontab	■		gmond

Hadoop生态系统

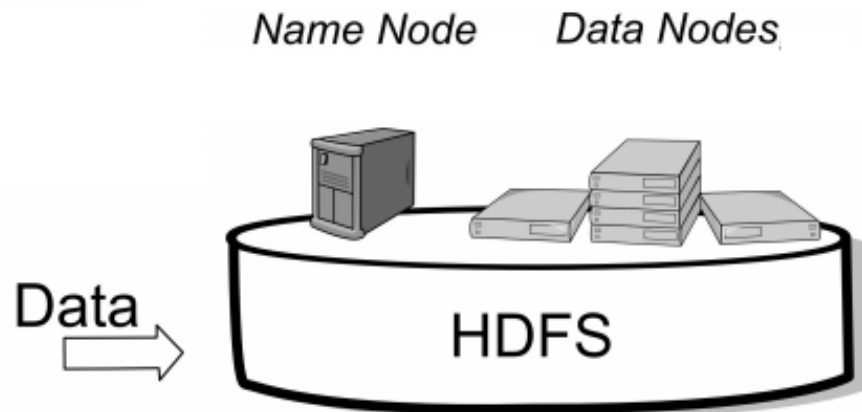
1. 大数据概述及和传统数据仓库的区别
2. 大数据应用场景
3. Hadoop生态系统概述
4. Hadoop核心模块
5. 从关系型数据到HDFS
6. HDFS数据处理方式
7. NoSQL及HBase概述
8. 组件协做方式及性能监控

- Hadoop的框架最核心的设计就是：HDFS和MapReduce。HDFS为海量的数据提供了存储，而MapReduce为海量的数据提供了计算框架。
- Hadoop Distributed File System (HDFS)

对外部客户访问而言，HDFS就像一个传统的分级文件系统。可以创建、删除、移动或重命名文件等等。但是 HDFS 的架构是基于一组特定的节点构建，这些节点包括一个NameNode和多个DataNode。其中NameNode在 HDFS 内部提供元数据服务,DataNode为 HDFS 提供存储块。

存储在 HDFS 中的文件首先被分成块，然后将这些块复制到多个计算机中(DataNode)，块的大小(通常为 64MB)和复制的块数量在创建文件时由客户机决定。NameNode 可以控制所有文件操作。HDFS 内部的所有通信都基于标准的TCP/IP 协议。

➤ Hadoop Distributed File System (HDFS)



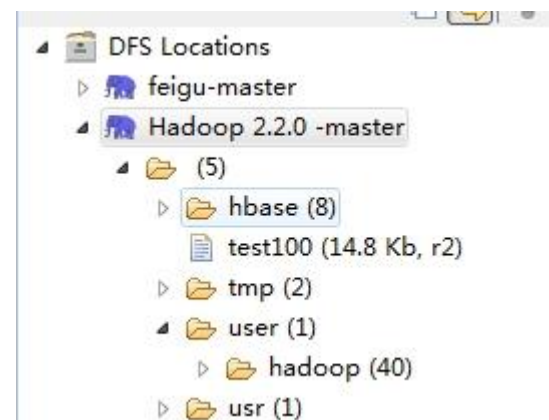
NameNode	DataNode
存储元数据	存储文件内容
元数据保存在内存中	文件内容保存在磁盘
保存文件, block , datanode之间的映射关系	维护了block id到datanode本地文件的映射关系

➤ Hadoop Distributed File System (HDFS)

演示如何使用命令行操作HDFS文件

```
feigu@slave01:~$ hadoop fs -ls
14/12/24 18:03:45 WARN util.NativeCodeLoader: Unable to load native-hadoop library
Found 11 items
drwxr-xr-x  - hadoop supergroup      0 2014-12-04 13:49 citylevel
drwxr-xr-x  - hadoop supergroup      0 2014-11-23 15:26 fund_data
drwxr-xr-x  - hadoop supergroup      0 2014-11-23 16:11 fund_data_output
drwxrwxr-x  - hadoop supergroup      0 2014-12-09 11:13 hbase
drwxrwxr-x  - hadoop supergroup      0 2014-11-02 09:52 history
drwxr-xr-x  - hadoop supergroup      0 2014-12-02 13:48 id-birthplace
drwxr-xr-x  - hadoop supergroup      0 2014-12-02 16:32 member_input
drwxr-xr-x  - hadoop supergroup      0 2014-12-08 15:08 member_output
drwxrwxr-x  - hadoop supergroup      0 2014-11-10 08:18 output
drwxr-xr-x  - hadoop supergroup      0 2014-11-15 12:11 roomingData
drwxr-xr-x  - hadoop supergroup      0 2014-11-15 12:36 roomingData-output
```

演示如何使用Eclipse插件操作HDFS文件



➤ MapReduce计算框架

MapReduce 是大规模数据（TB 级）计算的利器，Map 和Reduce 是它的主要思想，来源于函数式编程语言。其中，Map负责将数据打散，Reduce负责对数据进行聚集，用户只需要实现map 和reduce 两个接口，即可完成TB 级数据的计算，常见的应用包括：日志分析和数据挖掘等数据分析应用。另外，还可用于科学数据计算，如圆周率PI 的计算等。

Hadoop MapReduce的实现也采用了Master/Slave 结构。Master 叫做JobTracker，而Slave 叫做TaskTracker。用户提交的计算叫做Job，每一个Job会被划分成若干个Tasks。JobTracker负责Job 和Tasks 的调度，而TaskTracker负责执行Tasks。

➤ MapReduce计算框架

在hadoop安装目录下有一个hadoop-examples-x.jar文件，里面有计算文件中字符个数的示例程序，通过运行该例子，来分析MapReduce工作原理。

```
//在hadoop文件系统下创建目录wordcount_data
[hadoop@master bin]$ hadoop fs -mkdir wordcount_data

//把测试文件wordcount.txt拷贝到hdfs中
[hadoop@master bin]$ hadoop fs -copyFromLocal /mnt/hgfs/share-
folder/wordcount.txt wordcount_data

//查看该txt文件的位置
[hadoop@master bin]$ hadoop fs -ls wordcount_data
Found 1 items
-rw-r--r-- 2 hadoop supergroup      86 2014-06-08 16:42
/user/hadoop/wordcount_data/wordcount.txt
```

This is a hadoop test line.
And this is the second line.
And this is the third line.

wordcount.txt

➤ MapReduce计算框架

**hadoop jar hadoop-examples-1.1.2.jar wordcount wordcount_data
wordcount_output**

参数	含义
hadoop	Hadoop命令关键字
jar	要运行的命令类型。比如dsf代表查看hadoop文件系统；jar代表要运行jar包
hadoop-examples-1.1.2.jar	要运行的jar文件名。文件名要包含路径。Hadoop-examples-x.jar文件在hadoop安装目录的根目录下。文件内容是运行和mapreduce有关的各种sample
wordcount	要运行的是统计文件中单词个数
wordcount_data	要统计的文件在HDFS中的目录
wordcount_output	生成的结果文件在HDSF中的目录

➤ MapReduce计算框架

运行example例子中提供的wordcount计数程序并查看结果

```
[hadoop@master bin]$ hadoop fs -ls wordcount_output
```

Found 3 items

```
-rw-r--r--  2 hadoop supergroup      0 2014-06-08 17:01
/user/hadoop/wordcount_output/_SUCCESS
```

```
drwxr-xr-x  - hadoop supergroup      0 2014-06-08 17:01
/user/hadoop/wordcount_output/_logs
```

```
-rw-r--r--  2 hadoop supergroup    76 2014-06-08 17:01
/user/hadoop/wordcount_output/part-r-00000
```

```
[hadoop@master bin]$ hadoop fs -cat wordcount_output/part-r-00000
```

And 2

This 1

a 1

hadoop 1

is 3

line. 3

second 1

test 1

the 2

third 1

this 2

part-r-00000

➤ MapReduce计算框架

在Map阶段，原始数据被输入mapper进行过滤和转换，获得的中间数据在Reduce阶段作为reducer的输入，经过reducer的聚合处理，获得最终的处理结果。

为了适应多样化的数据环境，MapReduce中把键/值数据对(Key-Value pair)作为基础数据单元。Key和Value可以是简单数据类型，比如整数、浮点数、字符串等；也可以是复杂数据结构，比如列表、数组、自定义结构等。

Map阶段和Reduce阶段都将键/值作为输入和输出，其公式表达如下：

$$\text{Map:} \langle k1, v1 \rangle \rightarrow \{ \langle k2, v2 \rangle \}$$

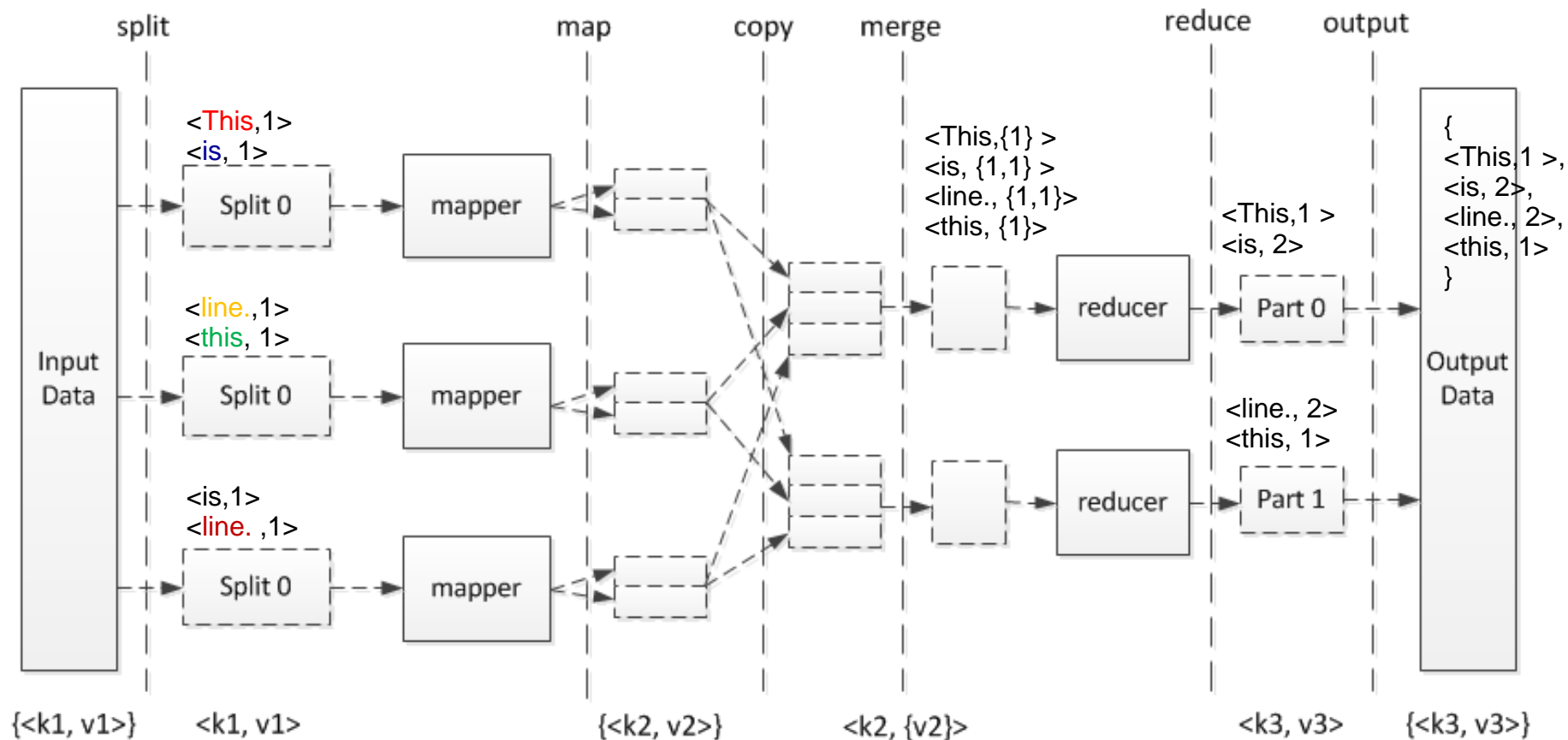
$$\text{Reduce:} \langle k2, \{v2\} \rangle \rightarrow \{ \langle k3, v3 \rangle \}$$

其中， $\langle \dots \rangle$ 代表键/值数据对， $\{ \dots \}$ 代表列表(List)

➤ MapReduce计算框架

Wordcount计数过程中，MapReduce的处理过程如下示意：

This is a hadoop test line.
And this is the second line.



➤ MapReduce计算框架

使用MapReduce算法找出外汇报价信息中每个货币对的最高价。从中国货币网

(<http://www.chinamoney.com.cn/fe/Channel/21478>) 下载一年来人民币外汇即期月报，从月报中找出每个货币对的各种统计指标。

www.chinamoney.com.cn			
同业拆借月报	质押式回购月报	买断式回购月报	现券买卖月报
人民币外汇即期月报	人民币外汇掉期月报	人民币外汇远期月报	外币对即期月报
人民币外汇即期月报			
2014-05			导出EXCEL
币种	成交金额(亿美元)	加权价	成交笔数
USD.CNY	2976.64	6.2382	30482
HKD.CNY	14.16	0.80458	744
100JPY.CNY	66.69	6.1304	1942
EUR.CNY	28.26	8.5686	1486
GBP.CNY	1.12	10.5404	186
AUD.CNY	21.85	5.8052	629
NZD.CNY	6.01	5.3628	131
CAD.CNY	0.07	5.7134	53
CNY.MYR	0.14	0.51736	23
CNY.RUB	1.07	5.5564	140
注：本表于每月月初更新上月统计数据			

```
[hadoop@master data]$ ls
201301.txt 201304.txt 201307.txt 201310.txt 201401.txt 201404.txt
201302.txt 201305.txt 201308.txt 201311.txt 201402.txt
201303.txt 201306.txt 201309.txt 201312.txt 201403.txt
```

```
USD.CNY,3380.34,6.1341,27595
HKD.CNY,17.67,0.79088,868
100JPY.CNY,184.73,6.1496,2554
EUR.CNY,32.17,8.0307,1938
GBP.CNY,0.88,9.2989,186
AUD.CNY,30.41,5.6276,1111
CAD.CNY,0.03,5.8734,29
CNY.MYR,0.16,0.51727,34
CNY.RUB,0.78,5.3329,139
```

➤ MapReduce计算框架

使用MapReduce算法找出外汇报价信息中每个货币对的最高价。

查看结果文件

100JPY.CNY	9.0768
AUD.CNY	6.5227
CAD.CNY	6.2315
CNY.MYR	0.5414
CNY.RUB	5.8325
EUR.CNY	8.6032
GBP.CNY	10.3748
HKD.CNY	0.80359
NZD.CNY	9.3564
USD.CNY	6.2331
Part-00000	

Hadoop生态系统

1. 大数据概述及和传统数据仓库的区别
2. 大数据应用场景
3. Hadoop生态系统概述
4. Hadoop核心模块
5. 从关系型数据到HDFS
6. HDFS数据处理方式
7. NoSQL及HBase概述
8. 组件协做方式及性能监控

➤ Sqoop和Flume

Hadoop HDFS文件存储方式推出后，随之而来的出现了很多从HDFS收格式到其他数据格式的导入导出工具。其中用的比较多的有Sqoop和Flume。



sqoop最早于2009年5月成为Hadoop的一个贡献模块，2012年3月升级成为Apache顶级项目。它主要用来在

Hadoop和关系数据库中传递数据。通过sqoop，我们可以方便的将数据从关系数据库导入到HDFS，或者将数据从HDFS导出到关系数据库。

Flume最早是Cloudera提供的日志收集系统，目前是Apache的一个顶级项目，Flume支持在日志系统中定制各类数据发送方，用于收集数据。并对数据进行简单处理，写到各种数据接受方(可定制)。



➤ 利用Sqoop工具把Mysql中的数据导入到HDFS

1) 列出mysql数据库中的所有数据库

```
sqoop list-databases --connect jdbc:mysql://slave03:3306/ --username feigu --P
```

2) 连接mysql并列出数据库feigu中的表

```
sqoop list-tables --connect jdbc:mysql://slave03:3306/feigu --username feigu --P
```

3) 把feigu数据库中sqoop_test表中的所有数据导出到Hive

```
sqoop import --connect jdbc:mysql://slave03:3306/feigu --username feigu --P --  
table sqoop_test --hive-import
```

4) 在HDFS中查看导出的结果

```
hadoop fs -ls /user/hive/warehouse/sqoop_test
```

Hadoop生态系统

1. 大数据概述及和传统数据仓库的区别
2. 大数据应用场景
3. Hadoop生态系统概述
4. Hadoop核心模块
5. 从关系型数据到HDFS
6. HDFS数据处理方式
7. NoSQL及HBase概述
8. 组件协做方式及性能监控

HDFS解决了大数据的存储问题，而对数据的读写和运算主要利用MapReduce计算框架。由于Hadoop本身是用Java语言编写的，它的MapReduce对Java提供了详细的API, 但对于更多的数据仓库数据处理人员而言，把各种业务逻辑都用MapReduce表达出来，显然起点太高。并且每次Java编写MapReduce代码后，都要编译打包发布测试，流程太繁琐。

Hadoop生态系统中，除了Java之外还提供了三类自动化/半自动化的工具，来协助完成MapReduce数据处理：

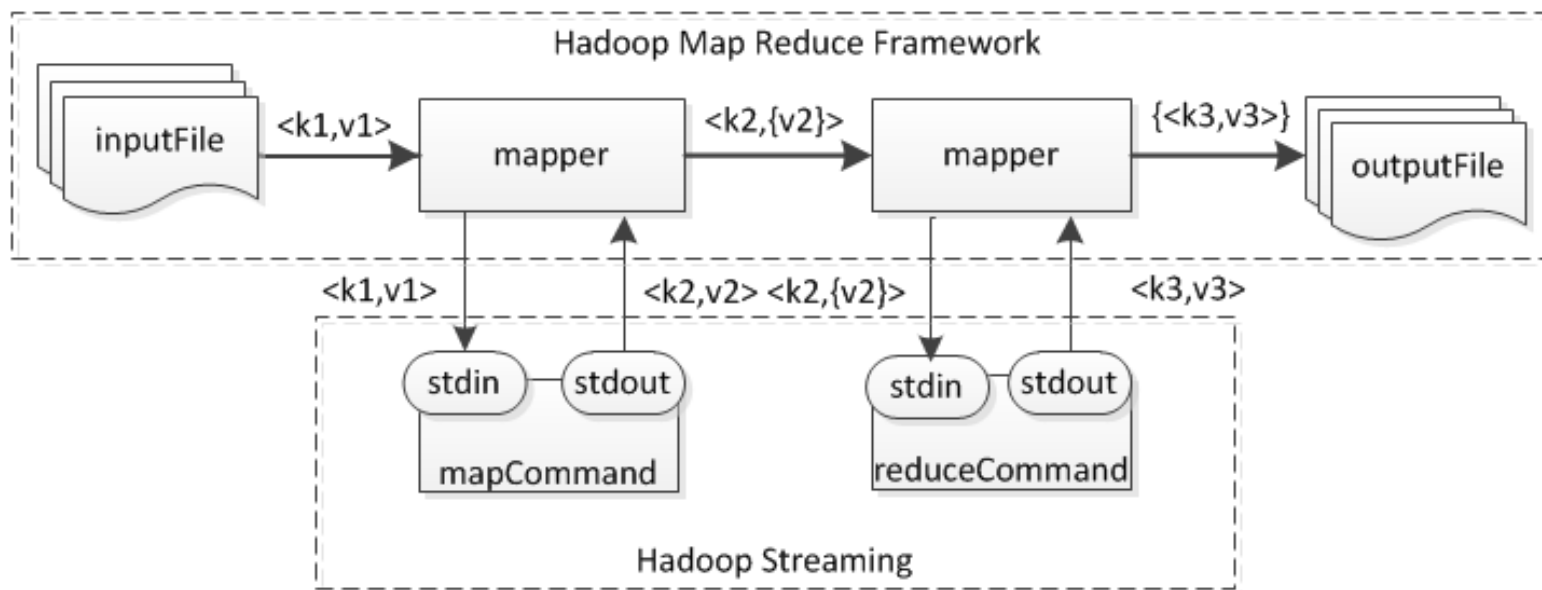
- Hadoop Streaming
- Pig
- Hive

➤ Hadoop Streaming及其使用

Hadoop环境中MapReduce应用的默认开发语言是Java，但在一些应用开发中，可能会使用到其他开发语言(例如C/C++、Python、Ruby, Shell Script等)编写的模块担任MapReduce任务的执行单元。Hadoop提供了Streaming工具，以支持使用除Java以外的语言来编写map和reduce函数。

Streaming采用UNIX标准输入输出机制(stdin和stdout)作为Hadoop运行环境和程序之间的接口。这种方式很好地实现了语言无关性，只要适配标准输入输出接口，开发人员可以使用任何语言编写map/reduce单元。

➤ Hadoop Streaming及其使用



Streaming模式的流程以及MapReduce运行框架图

➤ Apache Pig

Apache Pig是一种编程语言，它简化了Hadoop常见的工作任务。Pig可加载数据、表达转换数据以及存储最终结果。同时Pig可扩展使用Java中添加用户自定义函数(UDF)并支持数据转换。

它的优点是可以处理TB级的数据，提供一个控制台(Grunt)来接收和执行脚本命令行 -- Pig Latin。

Pig最早是在Yahoo! 里创建的，后来贡献给Apache。目的是为了让工程师们能更容易的掌握处理那里的大量的数据。

Pig & Pig Latin



➤ Apache Hive



Hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供类SQL查询功能。

本质是将SQL转换为MapReduce程序。与Pig一样，Hive的

核心功能是可扩展的。最早在Facebook内部使用，后来贡献给Apache。

Hive与传统数据库对比如下表：

	Hive	RDBMS
查询语言	HQL	SQL
数据存储	HDFS	Raw Device or Local File System
执行	MapReduce	Executor
执行延迟	高	低
数据处理规模	大	小
索引	0.8版本以后加入位图索引	有复杂的索引

➤ Apache Hive

SQL是一个可以有效、合理、直观地组织和使用的数据库模型，所有的开发和管理人员都会使用。如果要从传统的关系型数据库迁移到Hadoop上，Hive是一个很好的选择。它提供了Hive查询语言(HiveQL)的SQL方言，来查询存储在Hadoop集群中的数据。HQL不符合ANSI 92标准，语法更加类似于MySQL方言。

Hive最适合于数据仓库应用程序，可以维护海量数据，提供面向主题的查询分析功能进行数据挖掘。

Hive不是一个完整的数据库，它是依附于HDFS文件系统的。不支持记录级别的更新、插入和删除，即不支持OLTP(联机事务处理)，也不支持事务。它把每一个HQL都转换为MapReduce任务，而MapReduce任务的启动过程需要消耗较长的时间，故查询延时比较严重。在Hive中，即使数据集相对较小，往往也需要执行更长的时间。本章节主要介绍一下HQL数据操作, Streaming和HiveThrift服务。

➤ Apache Hive -- 命令行界面功能介绍(Command Line Interface)

hive命令行界面，也就是CLI，是和Hive交互最常见的方式。使用CLI，用户可以创建表、检查模式及查询表，等等。

```
hive> show databases;  
hive> use default;  
hive> show tables;  
hive> desc formatted sqoop_test;
```

```
hive> desc formatted sqoop_test;  
OK  
# col_name          data_type          comment  
  
id                  int  
s1                  string  
s2                  string  
  
# Detailed Table Information  
Database:           default  
Owner:               hadoop  
CreateTime:          Sun Nov 23 18:07:11 CST 2014  
LastAccessTime:      UNKNOWN  
Protect Mode:        None  
Retention:           0  
Location:             hdfs://master:49100/user/hive/warehouse/sqoop_test  
Table Type:          MANAGED_TABLE
```

➤ Apache Hive -- Streaming功能介绍

Hive是通过利用或扩展Hadoop的组件功能来运行的。Streaming提供的是另一种处理数据的方式。在streaming job中，Hadoop Streaming API会为外包进程开启一个I/O管道。然后数据会被传输给这个进程，其会从标准输入中读取数据，然后通过标准输出来写结果数据，最后返回到Streaming API job。

Streaming方式对于快速原型设计非常有用，其缺点是执行效率较低，并且要求每个slave都有对应的运行环境。以下代码演示如何利用python脚本对Hive表的结果数据变换大小写：

```
hive> select * from sqoop_test;
```

```
1          Java      Oracle
```

```
2          C#        Micro$oft
```

```
3          Hadoop    Apache
```

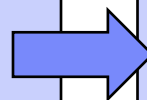
```
4          PHP       Zend
```

```
Time taken: 0.179 seconds, Fetched: 4 row(s)
```


➤ Apache Hive -- Streaming功能介绍

在hive中使用add file命令，把python脚本加入hive运行环境, 利用transform函数进行转换

```
hive> ADD FILE toUpperCase.py;  
Added resource: toUpperCase.py  
hive> select  
    transform(s1,s2)  
    using 'python toUpperCase.py'  
    as str1  
from  
    sqoop_test;
```



```
Total MapReduce CPU Time Spent: 3 seconds  
510 msec  
OK  
JAVA;ORACLE;  
C#;MICRO$OFT;  
HADOOP;APACHE;  
PHP;ZEND;  
Time taken: 698.862 seconds, Fetched: 4 row(s)
```

➤ Apache Hive -- Streaming功能介绍

```
# toUpperCase.py脚本代码
import sys
for line in sys.stdin:
    words = line.strip().split('\t')
    for word in words:
        sys.stdout.write(word.upper() + ';')
        sys.stdout.flush()
sys.stdout.write('\n')
```

Python, 是一种面向对象、解释型计算机程序设计语言,



由Guido van Rossum于1989年底发明，第一个公开发行人版发行于1991年。

Python语法简洁而清晰，具有丰富和

强大的类库。它常被昵称为胶水语言，它能够把用其他语言制作的各种模块（尤其是C/C++）很轻松地联结在一起。常见的一种应用情形是，使用Python快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分，用更合适的语言改写，比如3D游戏中的图形渲染模块，性能要求特别高，就可以用C++重写。

➤ Apache Hive -- Hive的Thrift服务

Hive具有一个可选的组件叫做HiveServer或者HiveThrift, 其允许通过指定端口访问Hive。Thrift (<http://thrift.apache.org>) 是一个软件框架, 用于跨语言的服务开发。只通过CLI方式访问Hive是胖客户端方式, Thrift允许客户端使用Java、C++、Ruby等方式远程访问Hive。

验证服务器端是否启用HiveServer服务

```
netstat -nl | grep 10000
```

使用Java的JDBC API可以像访问其他数据库一样来访问Hive中表数据。

- 1) 把hive\lib\hive-jdbc-0.13.0.jar加入类路径
- 2) 使用org.apache.hive.jdbc.HiveDriver字符串加载该驱动
- 3) jdbc:hive2://slave02:10000/default是连接hiveServer2的jdbc连接字符串

➤ Apache Hive

使用JDBC连接HiveServer

检索表数据的示例代码

```
public boolean connHive() throws ClassNotFoundException, SQLException {
    boolean bRet = false;
    Class.forName(DriverName);
    if (m_conn == null) {
        m_conn = DriverManager.getConnection(url, userName, password);
        if (m_stmt == null) {
            m_stmt = m_conn.createStatement();
        }
        bRet = true;
    }
    return bRet;
}
```

```
String url = "jdbc:hive2://slave02:10000/default";
```

```
String uname = "hadoop";
```

```
String pwd = "";
```

```
try {
    HiveJdbcUtil dbUtil = new HiveJdbcUtil(url, uname, pwd);
    if (dbUtil.connHive()) {
        System.out.println("连接Hive Thrift Server2成功....");
        String hql = "select id ,s1, s2 from sqoop_test";
        ResultSet rs = dbUtil.ExecQuery(hql);
        while (rs.next()) {
            System.out.println(String.format("编号=%d, 产品=%s, 所属=%s", rs.getInt("id"), rs.getString("s1"), rs.getString("s2")));
        }
        rs.close();
        dbUtil.Close();
    }
}
```

➤ Apache Hive

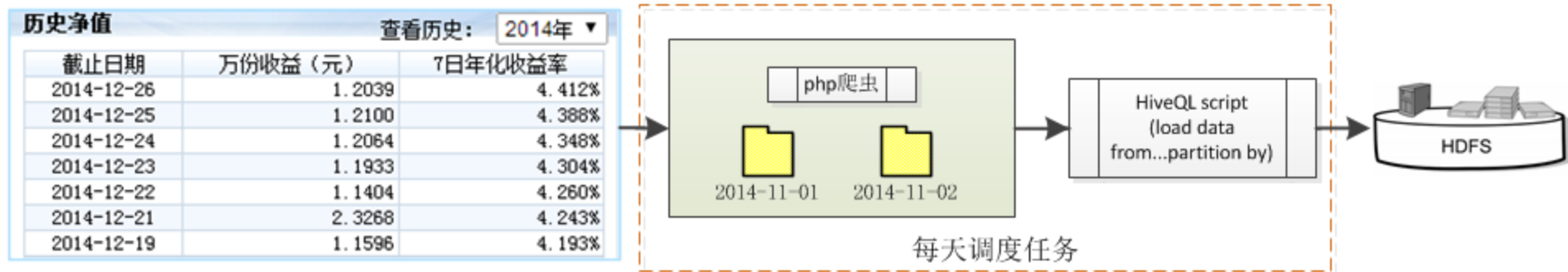
把代码编译打包成jar文件，shell脚本调用执行，结果如下

```
exportHadoopClasspath
exportJarClasspath
java -cp ${CLASSPATH}:/home/feigu/training/ReadHive.jar com.feigu.ReadHiveData
```

```
feigu@slave03:~/training$ sh readHive.sh
log4j:WARN No appenders could be found for logger (org.apache.thrift.transport.TSaslTransport).
log4j:WARN Please initialize the log4j system properly.
连接Hive Thrift Server2成功....
编号=1, 产品=Java, 所属=Oracle
编号=2, 产品=C#, 所属=Microsoft
编号=3, 产品=Hadoop, 所属=Apache
编号=4, 产品=PHP, 所属=Zend
feigu@slave03:~/training$
```

➤ Apache Hive -- 示例项目演示(1)

演示如何从网站上抓取每天基金净值信息，利用hive按照日期分区导入至HDFS



- ① 使用php做爬虫，每天收盘后定时从金融网站上抓取基金编号、7日收益率、年化收益率信息。以日期为文件名保存成文本文件
- ② 用HQL脚本导入指定日期的文本文件到hive中的基金表，以日期为分区标准
- ③ shell script中调度该HQL脚本，并传入基金日期

➤ Apache Hive -- 示例项目演示(2)

从<http://fund.jrj.com.cn/archives,000569,jjjz.shtml>网站上抓取每个基金的每日报价信息。使用的工具是php的simpleHtmlDom工具类

```
function getDailyFundDataByUrl($url,$fund_no){
    >> try {
    >> >> $fundObj = new FundModel();
    >> >> $fundObj->fundNo = $fund_no;
    >> >> $response = send_curl_request($url);

    >> >> if($response){
    >> >> >> $html = str_get_html($response);
    >> >> >> $table = $html->find('table[class="table"];0);
    >> >> >> $tr = $table->find("tr",1); //只取第一行最新的记录
    >> >> >> $tds = $tr->find("td");
    >> >> >> if(count($tds)==3 && $tds[0] && $tds[1] && $tds[2]){
    >> >> >> >> $dt = $tds[0]->plaintext;
    >> >> >> >> $yearval = $tds[1]->plaintext;
    >> >> >> >> $roi7day = $tds[2]->plaintext;
    >> >> >> >> //如果页面上的日期和当前日期不一样,就说明当日无基金数据,退出循环
    >> >> >> >> $yestoday = date("Y-m-d",strtotime('-1 day'));
    >> >> >> >> if($dt == $yestoday){
    >> >> >> >> >> if(strlen($dt)>0 && strlen($yearval)>0 && strlen($roi7day)>0){
    >> >> >> >> >> >> $fundObj->fundDate = $dt;
    >> >> >> >> >> >> $fundObj->ROI7days = fmtROI7($roi7day);
    >> >> >> >> >> >> $fundObj->ROIyear = $yearval;
    >> >> >> >> >> >> echo "curl-$fund_no-OK.\n";
    }
```

```
feigu@slave03:~/php_curl/service/feigu/money/temp$ ll
total 108
-rw-rw-r-- 1 feigu feigu 550 Nov 25 17:13 2014-11-24.txt
-rw-rw-r-- 1 feigu feigu 549 Nov 26 17:38 2014-11-25.txt
-rw-rw-r-- 1 feigu feigu 551 Nov 27 15:50 2014-11-26.txt
-rw-rw-r-- 1 feigu feigu 550 Nov 28 15:51 2014-11-27.txt
```

➤ Apache Hive -- 示例项目演示(3)

抓取后的基金日报价信息是用逗号分隔的，按基金编号/7日收益/年化收益保存

```
000198,0.04633,1.3361  
000581,0.04777,2.1862  
000569,0.04304,1.1933
```

在Hive中创建基金表时使用了日期分区。Hive中分区功能是非常有用的，因为Hive通常要对输入进行全盘扫描来满足查询条件，虽然Hive有索引功能，通过创建很多分区确实可以优化一些查询。用来分区的字段可以并列多个，组合使用。

```
create table if not exists tbl_fund_info (  
fundno      string comment 'fundNo',  
roidays     float comment '7days ROI',  
roiyear     float comment 'year ROI'  
) partitioned by (  
  funddate string comment 'fund date(format yyyy-mm-dd)')  
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```


➤ Apache Hive -- 示例项目演示(4)

hive不支持用insert语句一条一条的进行插入操作，也不支持update操作。数据是以load的方式加载到建立好的表中。数据一旦导入就不可以修改。

```
LOAD DATA LOCAL INPATH '2014-12-23.txt' INTO TABLE tbl_fund_info  
PARTITION (funddate='2014-12-23');
```

参数值	参数含义
LOCAL	如果指定LOCAL, 是从hive当前运行目录下去获取文件; 如不指定, 则是按照schema中指定的方式去获取
INTO TABLE	前面可以加上OVERWRITE关键字, 来区分是覆盖导入还是追加导入
PARTITION	分区信息必须要指定

➤ Apache Hive -- 示例项目演示(5)

使用load data命令把文本文件加载到HDFS

```
hive> LOAD DATA LOCAL INPATH '2014-12-23.txt' INTO TABLE tbl_fund_info PARTITION (funddate='2014-12-23');
Copying data from file:/home/feigu/php_curl/service/feigu/money/temp/2014-12-23.txt
Copying file: file:/home/feigu/php_curl/service/feigu/money/temp/2014-12-23.txt
Loading data to table feigu.tbl_fund_info partition (funddate=2014-12-23)
Partition feigu.tbl_fund_info{funddate=2014-12-23} stats: [numFiles=1, numRows=0, totalSize=549, rawDataSize=0]
OK
Time taken: 2.164 seconds
```

查看Hive表中的数据

```
hive> select * from tbl_fund_info;
OK
000198 0.04633 1.3361 2014-12-23
000581 0.04777 2.1862 2014-12-23
000569 0.04304 1.1933 2014-12-23
530002 0.0376 1.033 2014-12-23
```

查看导入至HDFS中的文件

```
hadoop fs -ls hdfs://master:49100/user/hive/warehouse/feigu.db/tbl_fund_info/funddate=2014-12-23
```

把load data命令整合进入shell脚本，方便自动调度

```
#`hive<<EOF
    echo "load data local inpath '$funddate_file' overwrite into table tbl_fund_info partition
(funddate='$funddate');" | hive
#EOF`
```

Hadoop生态系统

1. 大数据概述及和传统数据仓库的区别
2. 大数据应用场景
3. Hadoop生态系统概述
4. Hadoop核心模块
5. 从关系型数据到HDFS
6. HDFS数据处理方式
7. NoSQL及HBase概述
8. 组件协做方式及性能监控

➤ NoSQL简介

NoSQL，泛指非关系型的数据库。随着互联网web2.0网站的兴起，传统的关系数据库在应付web2.0网站，特别是超大规模和高并发的SNS类型的web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。

和RDBMS相比，NoSQL不使用SQL作为查询语言。其存储可以不需要固定的表模式，通常也会避免使用RDBMS的JOIN操作，一般都具备水平可扩展的特性。NoSQL的实现具有两个特征：使用硬盘和把随机存储器作为存储载体。

按照存储格式来分可以分为4类：键值(Key-Value)存储数据库、列存储数据库、文档型数据库和图形(Graph)数据库。目前比较流行的NoSQL数据库有Cassandra、Lucene、Neo4J、MongoDB和Hbase。

➤ NoSQL简介 -- 传统RDBMS和NoSQL的优缺点比较

RDBMS缺点	NoSQL优点
高并发读写的瓶颈。Web2.0模式下要实时生成动态页面而无法使用静态化技术，对于每秒上万次的写入DB操作，硬盘I/O存在明显瓶颈	扩展性强。每种NoSQL产品都去掉关系型数据库的关系特性，弱关联的数据更容易扩展，使得很容易实现支撑数据从TB级别到PB级别的过度。
可扩展性的限制。DB无法像Web Server或App Server那样依靠简单增加节点来平滑扩展性能，往往要停机维护和数据迁移。	并发性能好。NoSQL数据库具有良好的读写性能，其得益于它的弱关系性特点，数据库的结构简单。
事务一致性的负面影响。保证数据完整性的唯一方法是使用事务，这会消耗数据库资源，而很多Web系统并不需要严格的数据一致性。	数据模型灵活。NoSQL无需事先为要存储的数据建立字段，随时可以存储自定义的数据格式。NoSQL允许用户随时添加字段。而对传统RDBMS，增删字段是非常麻烦的事情，尤其是对数据量非常大的表。

► HBase - 和Hadoop之间的关系

HBase (Hadoop Database) 是一个高可靠、高性能、面向列、可伸缩的分布式数据库系统。它

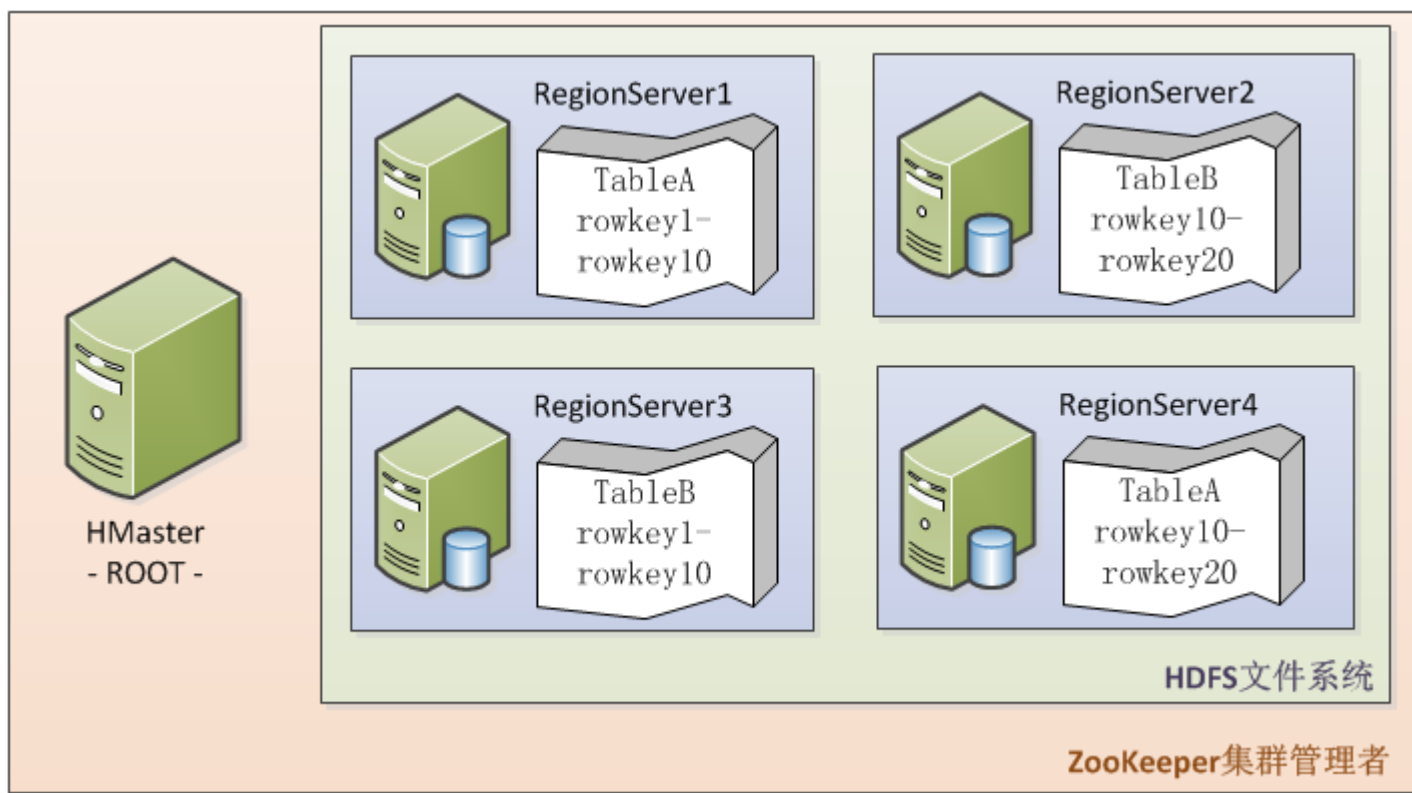


使用类似GFS的HDFS作为底层文件存储系统，在其上运行MapReduce批量处理数据，使用ZooKeeper作为协同服务组件。

HBase项目使用Java语言实现，最初是由Google Bigtable原型演化而来，2007年第一个简单可用的HBase版本发布，2008年1月，Hadoop升级为Apache的顶级项目时，HBase作为Hadoop的子项目存在。后来随着hadoop版本的提升而不断更新，2010年5月HBase成为Apache顶级项目。截止14年底HBase稳定版本是0.96。HBase的运行严重依赖于Hadoop，且二者版本存在协调关系。

➤ HBase - 架构及数据存储模型(1)

HBase是一个分布式的数据库，使用Zookeeper来管理集群。在架构层面上分为Master（Zookeeper中的leader）和多个RegionServer，基本架构如图：



➤ HBase - 架构及数据存储模型 (2)

在HBase的概念中，HMaster主服务器主要负责利用ZooKeeper为Region服务器分配或移除region, 起负载均衡作用。RegionServer对应于集群中的一个节点，而一个RegionServer负责管理多个Region。一个Region代表一张表的一部分数据，所以在Hbase中的一张表可能会需要很多个Region来存储其数据，但是每个Region中的数据并不是杂乱无章的，Hbase在管理Region的时候会给每个Region定义一个Rowkey的范围，落在特定范围内的数据将交给特定的Region，从而将负载分摊到多个节点上，充分利用分布式的优点。

另外，为了防止集群中master的单点故障，Hbase中启动多个HMaster实例，通过ZooKeeper的投票机制来保证总有一个HMaster在运行。

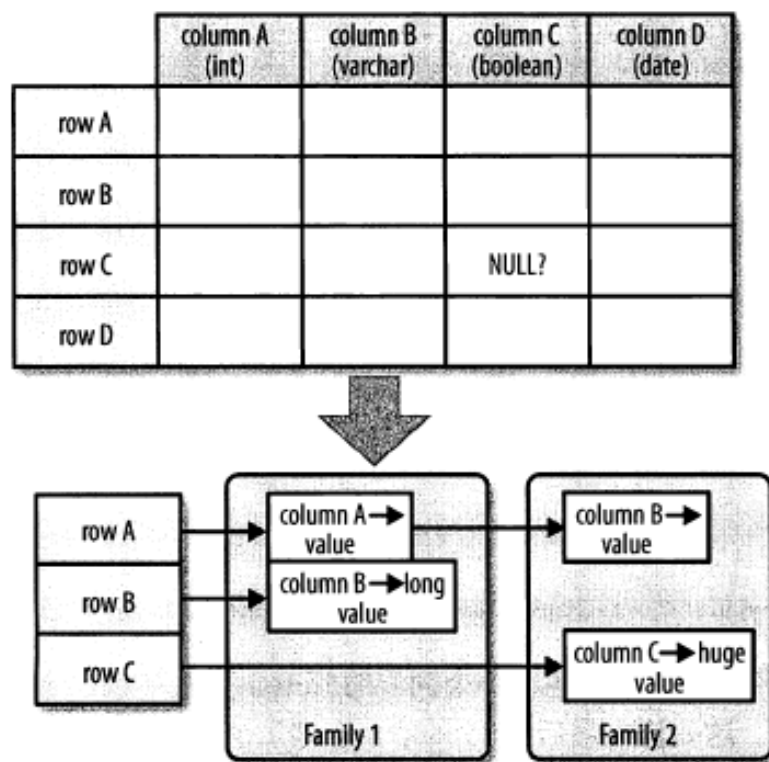
➤ HBase - 数据模型(1)

HBase是一种列式存储的分布式数据库，其核心概念是表(Table)。和传统的数据库一样，表由行和列组成，但HBase把不同的列组成一个列族(Column Family)，同一列的值又可以有多个版本(version)，这种组织形式主要是为了存取性能的考虑。

表：在HBase中数据以表的形式存储。表名使用Java String类型或byte[]字节数组表示。每个表名对应HDFS中的一个目录结构。创建表时需要指定表名和至少一个列族。列族影响表的物理存储结构。

列族：是一些列的集合。创建好后不能频繁修改，数量不能太多。列族名由可见字符组成。列族中包含列的数量没有限制，可以有数百万个列。列值没有类型和长度的限定。一个列族中的所有列存储在同一个底层文件(HFile)中。常见的引用列的格式是family:qualifier，qualifier是任意的字节数组。

➤ HBase - 数据模型 (2)



左图用可视化方式展示了普通数据库与列式HBase在行设计上的不同。行和列没有用二维表的模型排列，而是采用了标签描述 (tag metaphor) 的方式，信息保存在一个特定的标签下。

➤ HBase - 数据模型(3)

行键 (Rowkey) 是HBase中最重要的概念之一，它在表中用来唯一的标示一行，其值保存为二进制字节数组。表中的行是按照行键的字典序排序存储的。

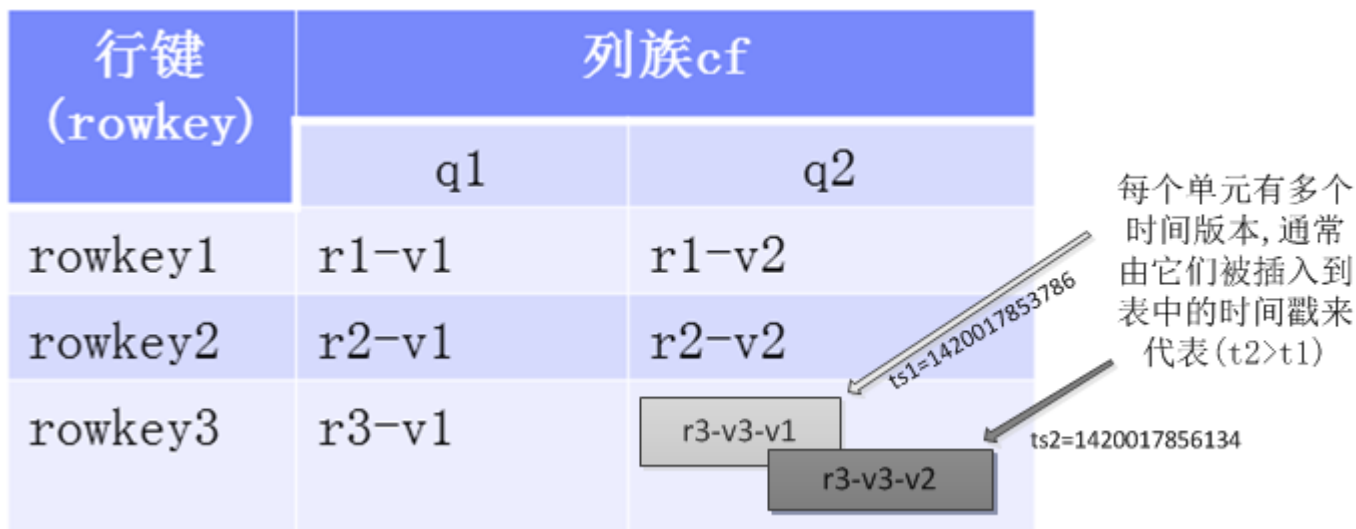
```
hbase(main):001:0> scan 'table1'
ROW                                COLUMN+CELL
row-1                             column=cf:val, timestamp=1420013446529, value=value-1
row-10                             column=cf:val, timestamp=1420013561396, value=value-10
row-11                             column=cf:val, timestamp=1420013570461, value=value-11
row-1a                             column=cf:val, timestamp=1420013605692, value=value-1a
row-2                             column=cf:val, timestamp=1420013577217, value=value-2
row-22                             column=cf:val, timestamp=1420013584822, value=value-22
row-3                             column=cf:val, timestamp=1420013592073, value=value-3
row-abc                           column=cf:val, timestamp=1420013599133, value=value-abc
8 row(s) in 1.2410 seconds
```

在HBase中行键是唯一的索引(新版本HBase中增加了辅助索引)，所以在设计行键时必须考虑检索数据的效率，比如把常用的检索字段组合作为rowkey，并且要保证其唯一性，但rowkey长度又不能太长。太长的rowkey会增加存储开销，降低内存利用率，从而降低索引命中率。另外rowkey的设计同时要考虑如何使其值能够尽量散列，这样会保证所有的数据不都是在一个region上，避免做读写时负载集中在个别region上面。

➤ HBase - 数据模型(4)

单元 (Cell) 是行键、列族和列名一起确定了一个单元。存储在单元里的数据称为单元值 (Value)。单元值没有数据类型，存储为字节数组 `byte[]`。

时间版本 (version)，单元值有时间版本，时间版本用时间戳标识，是一个 `long` 型数字，默认使用当前时间戳。HBase 保留单元值时间版本的数量基于列族进行配置，默认数量是 3 个。下图展示如何保存两个不同版本的数据：



➤ HBase - HBase客户端(1)

使用HBase客户端可以方便的操作HBase表中的数据。其中，HBase Shell工具、原生Java API客户端和Thrift客户端是三种最常见工具。此外还有REST(Representational State Transfer, 表述性状态转移)和Avro。

HBase Shell工具是安装HBase时就已经自带的客户端工具，它可以接受用户输入的DDL和DML语句来操作HBase表中的数据，其具体实现是用Ruby语言编写，并使用JRuby解释器。包括交互模式和批量模式两种使用方式。

```
feigu@slave02:~$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 0.96.2-hadoop2, r1581096, Mon Mar 24 16:03:18 PDT 2014

hbase(main):001:0>
```

➤ HBase - HBase客户端(2)

查看HBase集群状态命令status

```
hbase(main):001:0> status  
4 servers, 0 dead, 1.2500 average load
```

查看HBase版本命令version

```
hbase(main):002:0> version  
0.96.2-hadoop2, r1581096, Mon Mar 24 16:03:18 PDT 2014
```

查看HBase所有表命令list

```
hbase(main):003:0> list  
TABLE  
t_kaifang_log  
tbl_fund_info  
test.app.time  
3 row(s) in 0.1510 seconds  
=> ["t_kaifang_log", "tbl_fund_info", "test.app.time"]
```

➤ HBase - HBase客户端(3)

在Shell工具中操作表的命令包括DDL和DML。DDL是数据定义语言，包括创建表、修改表、上线和下线表、删除表等。创建表create命令使用如下

```
hbase(main):026:0> create 'test1', {NAME => 'cf', VERSIONS => 3, COMPRESSION => 'GZ' }  
0 row(s) in 1.7700 seconds  
=> Hbase::Table - test1
```

参数值	参数含义
test1	表名为test1
NAME => 'cf'	列族名为cf
VERSIONS => 3	Cell单元保存3个时间版本
COMPRESSION => 'GZ'	表的HFile采用gz压缩方式保存

➤ HBase - HBase客户端(4)

DML是数据操作语言，有数据写入、删除、查询和清空。不包括更新操作。

put命令是数据写入

```
put 'test1','rowkey1', 'cf:q1', 'r1-v1'  
put 'test1','rowkey1', 'cf:q2', 'r1-v2'  
put 'test1','rowkey2', 'cf:q1', 'r2-v1'  
put 'test1','rowkey2', 'cf:q2', 'r2-v2'
```

参数值	参数含义
test1	表名为test1
rowkey1	指定行键值为rowkey1
cf:q1	在列族cf中插入一个名字叫做q1的列
r1-v1	列族cf中cf:q1列的值是r1-v1

➤ HBase - HBase客户端(5)

scan命令是检索表数据的命令,可以指定列族中的特定列、显示某个列的所有时间戳版本或是限定返回的记录行数。

```
hbase(main):018:0> scan 'test1'
ROW                COLUMN+CELL
rowkey1            column=cf:q1, timestamp=1420018037109, value=r1-v1(input3)
rowkey1            column=cf:q2, timestamp=1420017853906, value=r1-v2
rowkey2            column=cf:q1, timestamp=1420017853994, value=r2-v1
rowkey2            column=cf:q2, timestamp=1420017854117, value=r2-v2
rowkey3            column=cf:q1, timestamp=1420017854197, value=r3-v1
rowkey3            column=cf:q2, timestamp=1420017856134, value=r3-v2
3 row(s) in 2.6400 seconds
```

```
hbase(main):019:0> scan 'test1', {COLUMNS=>['cf:q1'], LIMIT=>1, VERSIONS=>3}
ROW                COLUMN+CELL
rowkey1            column=cf:q1, timestamp=1420018037109, value=r1-v1(input3)
rowkey1            column=cf:q1, timestamp=1420017968281, value=r1-v1(input2)
rowkey1            column=cf:q1, timestamp=1420017853786, value=r1-v1
1 row(s) in 0.8980 seconds
```

```
hbase(main):020:0> scan 'test1', {COLUMNS=>['cf:q2']}
ROW                COLUMN+CELL
rowkey1            column=cf:q2, timestamp=1420017853906, value=r1-v2
rowkey2            column=cf:q2, timestamp=1420017854117, value=r2-v2
rowkey3            column=cf:q2, timestamp=1420017856134, value=r3-v2
3 row(s) in 0.2730 seconds
```

➤ HBase - HBase客户端(6)

scan命令的各种筛选条件可以组合使用。STARTROW参数是指定从哪一行开始取数据。

```
hbase(main):021:0> scan 'test1', {COLUMNS=>['cf:q2'], LIMIT=>2}
ROW          COLUMN+CELL
  rowkey1     column=cf:q2, timestamp=1420017853906, value=r1-v2
  rowkey2     column=cf:q2, timestamp=1420017854117, value=r2-v2
2 row(s) in 0.0550 seconds
hbase(main):023:0> scan 'test1', {COLUMNS=>['cf:q2'], LIMIT=>2, STARTROW=>'rowkey2,q2,r2-v2'}
ROW          COLUMN+CELL
  rowkey3     column=cf:q2, timestamp=1420017856134, value=r3-v2
1 row(s) in 0.0830 seconds

hbase(main):024:0> scan 'test1', {COLUMNS=>['cf:q2'], LIMIT=>2, STARTROW=>'rowkey1'}
ROW          COLUMN+CELL
  rowkey1     column=cf:q2, timestamp=1420017853906, value=r1-v2
  rowkey2     column=cf:q2, timestamp=1420017854117, value=r2-v2
2 row(s) in 0.0960 seconds
```

➤ HBase - HBase客户端(6)

scan命令的各种筛选条件可以组合使用。TIMERANGE参数是针对某个列值取指定时间戳区间的版本。TIMERANGE取值是左闭右开的。

```
hbase(main):055:0> scan 'test1', {COLUMNS=>['cf:q1'], TIMERANGE=>[0,1420017853787]}
```

```
ROW                COLUMN+CELL
  rowkey1          column=cf:q1, timestamp=1420017853786, value=r1-v1
1 row(s) in 0.0610 seconds
```

```
hbase(main):056:0> scan 'test1', {COLUMNS=>['cf:q1'], LIMIT=>1,TIMERANGE=>[1420017853786,1420017968282]}
```

```
ROW                COLUMN+CELL
  rowkey1          column=cf:q1, timestamp=1420017968281, value=r1-v1(input2)
1 row(s) in 0.1050 seconds
```

```
hbase(main):057:0> scan 'test1', {COLUMNS=>['cf:q1'], LIMIT=>1,TIMERANGE=>[1420018037109,2000000000000]}
```

```
ROW                COLUMN+CELL
  rowkey1          column=cf:q1, timestamp=1420018037109, value=r1-v1(input3)
1 row(s) in 0.0730 seconds
```

Shell客户端还可以引入JavaAPI类，在命令行中调用其中的方法。

```
hbase(main):058:0> import java.util.Date
```

```
file:/usr/hadoop/hbase-0.96.2/lib/jruby-complete-1.6.8.jar!/builtin/javasupport/core_ext/object.
```

```
rb:99 warning: already initialized constant Date
```

```
=> Java::JavaUtil::Date
```

```
hbase(main):059:0> Date.new(2000000000000000).toString()
```

```
=> "Wed May 18 11:33:20 GMT+08:00 2033"
```

➤ HBase - JavaAPI客户端(1)

HBase官方代码中包含原生访问客户端，是由Java语言实现，这也是最直接、最高效的客户端。相关类都在包org.apache.hadoop.hbase.client中，涵盖增，删，查等API。主要类包括HTable、HBaseAdmin、Put、Get、Scan、Delete等。

使用原生客户端之前首先要配置客户端，从HBaseConfiguration类中创建一个配置实例。设置的参数hbase.master、hbase.zookeeper.quorum和hbase.zookeeper.property.clientPort分别表示HMaster地址、Zookeeper队列名和Zookeeper端口。

```
public static void main(String[] args) {  
    Configuration conf = HBaseConfiguration.create();  
    try {  
        conf.set("hbase.master", "master");  
        conf.set("hbase.zookeeper.quorum", "master");  
        conf.set("hbase.zookeeper.property.clientPort", "2181");  
    }
```

➤ HBase - JavaAPI客户端(2)

使用HTable类中的方法包含4类：增删查方法、获取表元数据、获取状态信息方法和设置属性方法。HTable中的put方法可以向表中逐条或批量插入数据。

```

/****
 * 循环插入数据
 * @param conf
 */
public static void insertTableTest1(Configuration conf) {
    try {
        String tableName = "test1";
        HTable table = new HTable(conf, tableName);
        for(int i=1; i<=3; i++) {
            Put put = new Put(Bytes.toBytes("rowkey"+i));
            put.add(Bytes.toBytes("cf"), Bytes.toBytes("q1"), Bytes.toBytes("r"+i+"v1"));
            put.add(Bytes.toBytes("cf"), Bytes.toBytes("q2"), Bytes.toBytes("r"+i+"v2"));
            table.put(put);
        }
        System.out.println("insert recored ok.");
        table.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

➤ HBase - JavaAPI客户端(3)

使用HBaseAdmin类中的getTableDescriptor方法可以获取表的元数据信息。

```
/**
 * 读取表结构
 *
 * @param admin
 */
public static void readTableDesc(Configuration conf) {
    HBaseAdmin admin = null;
    try {
        admin = new HBaseAdmin(conf);
        HTableDescriptor tbdesc = admin.getTableDescriptor("test1".getBytes());
        System.out.println(tbdesc);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
'test1', {NAME => 'cf', BLOOMFILTER => 'ROW', VERSIONS => '3', IN_MEMORY => 'false',
KEEP_DELETED_CELLS => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL =>
'2147483647', COMPRESSION => 'GZ', MIN_VERSIONS => '0', BLOCKCACHE => 'true',
BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
```

➤ HBase - JavaAPI客户端(4)

使用HTable类中的get方法和scan方法可以检索表中的数据。Get是指定rowkey检索，scan和FilterList结合可以实现各种组合条件检索。

```
public static void getResultByColumn(Configuration conf,String tableName, String rowKey) throws IOException {
    HTable table = new HTable(conf, Bytes.toBytes(tableName));
    Get get = new Get(Bytes.toBytes(rowKey));
    Result result = table.get(get);
    for (KeyValue kv : result.list()) {
        System.out.println("family:" + Bytes.toString(kv.getFamily()));
        System.out.println("qualifier:" + Bytes.toString(kv.getQualifier()));
        System.out.println("value:" + Bytes.toString(kv.getValue()));
        System.out.println("Timestamp:" + kv.getTimestamp());
        System.out.println("-----");
    }
    table.close();
}
```

```
family:cf
qualifier:q1
value:r2v1
Timestamp:1420180810221
-----
family:cf
qualifier:q2
value:r2v2
Timestamp:1420180810221
```

➤ HBase - JavaAPI客户端(5)

假如多线程访问HBase，需要创建多个HTable对象，并且需要单独对每个HTable对象进行维护其创建、使用、消亡的整个过程，这样不仅增加开发成本，也不利于资源的合理使用，同时HTable写入时不是线程安全的。HBase官方提供了通过连接池的方式使用HTable，即HTablePool类。该类支持多线程使用HTable，并且是线程安全的。类似于MySQL数据库连接池。

➤ HBase - Hive和HBase整合(1)

Hive和HBase整合有两种方案，分别是利用hive_hbase-handler.jar工具类和MapReduce算法把Hive中的数据导入HBase中。

在hive安装目录lib下有hive_hbase-handler.jar文件，利用该jar包提供的功能可以实现hive和HBase之间的数据互通。数据是保存在HBase中,而hive作为一个连接客户端可以读写HBase表中数据。首先要在hive-site.xml中配置属性hive.aux.jars.path，值指定为hive-hbase-handler.jar。在hive中创建表：

```
CREATE TABLE hbase_table_city(level int, city string)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,cf1:val")
TBLPROPERTIES ("hbase.table.name" = "t_city");
```

Hive表中的level做为HBase中的rowkey, city作为列族之一

➤ HBase - Hive和HBase整合(2)

在HBase表中插入测试数据

```
hbase(main):016:0> put 't_city', 3, 'cf1:val', 'city3'  
hbase(main):017:0> put 't_city', 4, 'cf1:val', 'city4'  
hbase(main):018:0> put 't_city', 5, 'cf1:val', 'city5'
```

在Hive和HBase中分别查看数据

```
hbase(main):019:0> scan 't_city'  
ROW                COLUMN+CELL  
3                   column=cf1:val, timestamp=1418119690475, value=city3  
4                   column=cf1:val, timestamp=1418119700825, value=city4  
5                   column=cf1:val, timestamp=1418119709304, value=city5
```

```
hive> select * from hbase_table_city;  
3          city3  
4          city4  
5          city5
```

➤ HBase - Hive和HBase整合(3)

从Hive导入数据到HBase中还有另外一种方法，利用HBase所提供的工具类TableMapReduceUtil来实现。该类在hadoop.hbase.mapreduce包下，主要实现使用其中包含的TableMapper和TableReducer类来读写表中数据。

该Util类把从Hive表中读数据作为Mapper计算模型，写入数据到HBase作为Reduce模型。首先指定要读取的hive表中的列名和HBase中的rowkey，作为Mapper的输出。

在TableReducer类的重写方法reduce中，Text类型的key作为HBase表的行键rowkey，values枚举列表中存放的是每个列族中的值，把这两者构造成一个Put对象实例，放入reduce方法的输出value中，而reduce方法的输出key是HBase中的表名。

➤ HBase - Hive和HBase整合(4)

```
public static class MapperFromHive extends Mapper<Text, Text, Text, Text> {  
    @Override  
    public void map(Text key, Text values, Context context)  
        throws IOException, InterruptedException {  
        String[] valueArray = key.toString().split(",", -1);  
        // 把日期+基金编号作为rowkey  
        String rowkey = "";  
        if (valueArray.length == 3) {  
            rowkey = fund_date + "_" + valueArray[0];  
        }  
        //把整行的内容直接放入value中  
        context.write(new Text(rowkey), key);  
    }  
}
```

map方法的输入key对应hive表中的一行数据，字段之间在hive中以逗号分隔。从这些字段中找出要做行键的值。

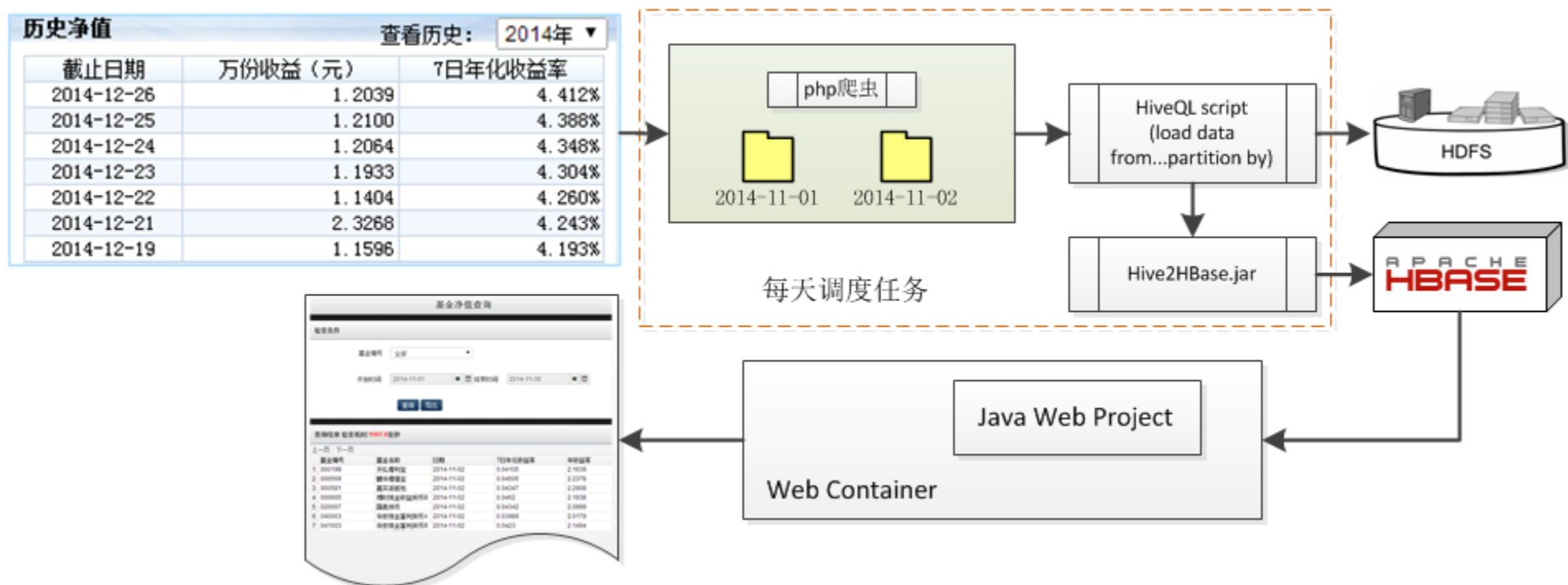
➤ HBase - Hive和HBase整合 (5)

```
public void reduce(Text key, Iterable<Text> values, Context context)
    throws IOException, InterruptedException {
    for (Text val : values) {
        String[] valArray = val.toString().split(",");
        if (valArray.length == 3) {
            for (int x=0; x<valArray.length; x++) {
                byte[] b_rowkey = Bytes.toBytes(key.toString());
                Put put = new Put(b_rowkey); // rowkey
                String each_val = valArray[x];
                byte[] b_family = Bytes.toBytes(family);
                byte[] b_qualifier = Bytes.toBytes(hiveColumnLst.get(x));
                byte[] b_val = Bytes.toBytes(each_val);
                put.add(b_family, b_qualifier, b_val);
                context.write(new Text(hbasetablename), put);
            }
        }
    }
}
```

map方法的输入key对应hive表中的一行数据，字段之间在hive中以逗号分隔。从这些字段中找出要做行键的值。

➤ HBase - 项目示例(1)

演示每天从网站上抓取的基金净值信息，如何从hive以日期为分区按天导入HBase，并能在页面上查询展现。



➤ HBase - 项目示例(2)

HBase基金表中为了前端检索需要，把日期+基金编号作为rowkey，存储格式是“10位日期_6位基金编号”，而把7日收益率和年化收益率作为列族中的值。查询时候可以按照日期时间段和基金编号组合查询。

```
hbase(main):005:0> scan 'tbl_fund_info', {LIMIT=>3}
ROW                                COLUMN+CELL
2004-01-05_040003                  column=cf:fundno, timestamp=1416796262483, value=040003
2004-01-05_040003                  column=cf:roidays, timestamp=1416796262483, value=0.02244
2004-01-05_040003                  column=cf:roiyear, timestamp=1416796262483, value=4.3041
2004-01-06_040003                  column=cf:fundno, timestamp=1416796915015, value=040003
2004-01-06_040003                  column=cf:roidays, timestamp=1416796915015, value=0.02456
2004-01-06_040003                  column=cf:roiyear, timestamp=1416796915015, value=0.5431
2004-01-07_040003                  column=cf:fundno, timestamp=1416797569947, value=040003
2004-01-07_040003                  column=cf:roidays, timestamp=1416797569947, value=0.01947
2004-01-07_040003                  column=cf:roiyear, timestamp=1416797569947, value=0.528
3 row(s) in 0.4180 seconds
```

➤ HBase - 项目示例(3)

对基金表的组合查询，采用FilterList类和Scan类配合使用，并使用PageFilter实现分页效果。

```
//日期介于开始结束之间(检索列是rowkey)
if (!"".equals(sd)&& "".equals(ed))) {
    filterList.addFilter( new RowFilter(
        CompareFilter.CompareOp.GREATER_OR_EQUAL,
        new BinaryComparator(Bytes.toBytes(sd))));

    filterList.addFilter( new RowFilter(
        CompareFilter.CompareOp.LESS_OR_EQUAL,
        new BinaryComparator(Bytes.toBytes(ed))));
}
```

```
//姓名匹配(检索列是rowkey)
if (!"".equals(fundno)) {
    filterList.addFilter( new RowFilter(
        CompareFilter.CompareOp.EQUAL,
        new SubstringComparator(fundno)));
}
s1.setFilter(filterList);
```


➤ HBase - 项目示例(4)

查询展现

基金净值查询

检索条件

基金编号

全部

开始时间:

2014-11-01

结束时间:

2014-11-30

查询

导出

查询结果 检索耗时:78.0毫秒

上一页

下一页

	基金编号	基金名称	日期	7日年化收益率	年收益率
1	110006	易方达货币A	2014-11-02	0.03742	2.0094
2	110016	易方达货币B	2014-11-02	0.03983	2.1409
3	200003	长城货币A	2014-11-02	0.06387	2.2019
4	200103	长城货币B	2014-11-02	0.06642	2.3334
5	202301	南方现金增利货币A	2014-11-02	0.04852	2.2994
6	202302	南方现金增利货币B	2014-11-02	0.05091	2.431
7	213009	宝盈货币A	2014-11-02	0.09751	1.0295
8	213909	宝盈货币B	2014-11-02	0.09997	1.161
9	270004	广发货币A	2014-11-02	0.04249	2.3146
10	270014	广发货币B	2014-11-02	0.04487	2.4461

Hadoop生态系统

1. 大数据概述及和传统数据仓库的区别
2. 大数据应用场景
3. Hadoop生态系统概述
4. Hadoop核心模块
5. 从关系型数据到HDFS
6. HDFS数据处理方式
7. NoSQL及HBase概述
8. 组件协做方式及性能监控

➤ 组件协做方式 - ZooKeeper

Hadoop的Master-Slave分布式架构及其之上的Hive, HBase等应用均需要使用到模块/机器间的状态同步或消息通信。为了解决底层的这类问题, Hadoop采用了ZooKeeper组件。ZooKeeper是一个分布式应用程序协调服务, 分布式应用程序可以基于它实现同步服务, 配置维护和命名服务等。

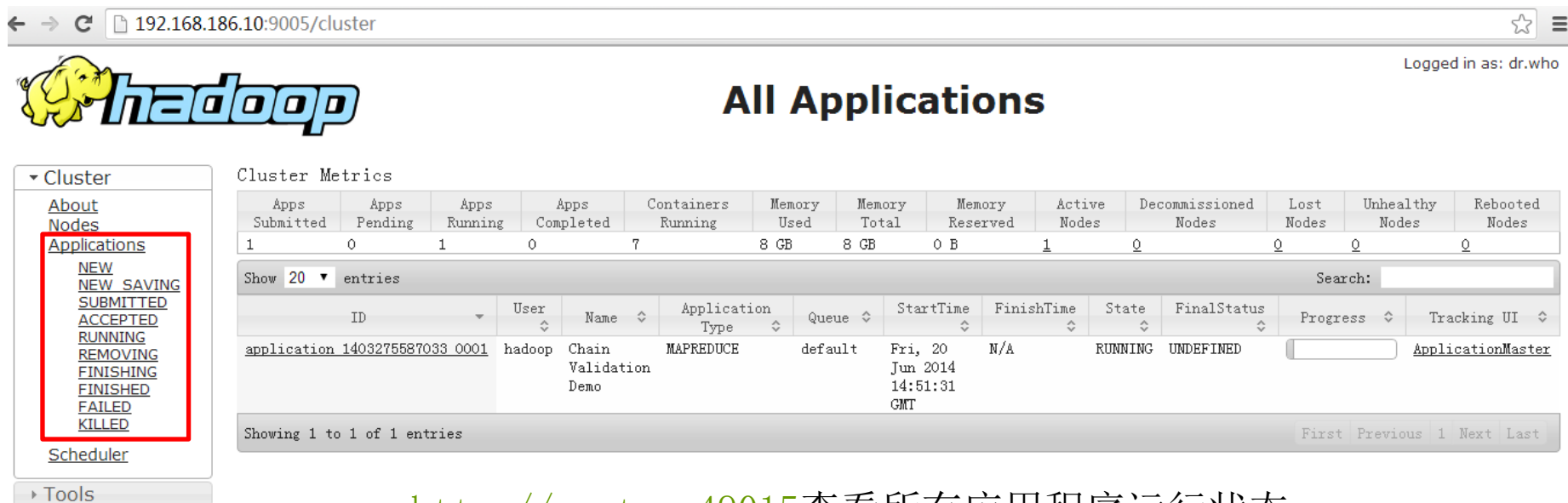
在分布式应用中, 由于工程师不能很好地使用锁机制, 以及基于消息的协调机制不适合在某些应用中使用, 因此需要有一种可靠的、可扩展的、分布式的、可配置的协调机制来统一系统的状态。ZooKeeper的目的就在于此。



比如在应用集群中, 我们常常需要让每一个机器知道集群中哪些机器是活着的, 并且在集群机器出现因为宕机、网络断链等原因能够不在人工介入的情况下迅速通知到每一个机器。使用ZooKeeper可以很方便做到这点。

➤ 性能监控 - Hadoop All Applications

作为一个分布式系统，用户要想了解Hadoop的状态和Job的运行状况是非常重要的。对Hadoop的监控主要分为两块：对文件系统的监控和对任务的监控。用户可以通过查看输出的log日志来监控，也可以通过Hadoop提供的Web监控界面来完成。



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	1	0	7	8 GB	8 GB	0 B	1	0	0	0	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1403275587033_0001	hadoop	Chain Validation Demo	MAPREDUCE	default	Fri, 20 Jun 2014 14:51:31 GMT	N/A	RUNNING	UNDEFINED	<div></div>	ApplicationMaster

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

<http://master:49015>查看所有应用程序运行状态

➤ 性能监控 - Hadoop JobHistory, HDFS



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes
1	0	1	0	7	8 GB	8 GB	0 B	1	0

Showing 1 to 1 of 1 entries

Cluster Metrics

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus
application_1403275687033_0001	hadoop	Chain Validation Demo	MAPREDUCE	default	Fri, 20 Jun 2014 14:51:31 GMT	N/A	RUNNING	UNDEFINED

Showing 1 to 1 of 1 entries

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	1	0	7	8 GB	8 GB	0 B	1	0	0	0	0

Showing 1 to 1 of 1 entries

Cluster Metrics

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1403275687033_0001	hadoop	Chain Validation Demo	MAPREDUCE	default	Fri, 20 Jun 2014 14:51:31 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaster

Showing 1 to 1 of 1 entries

<http://master:19988>

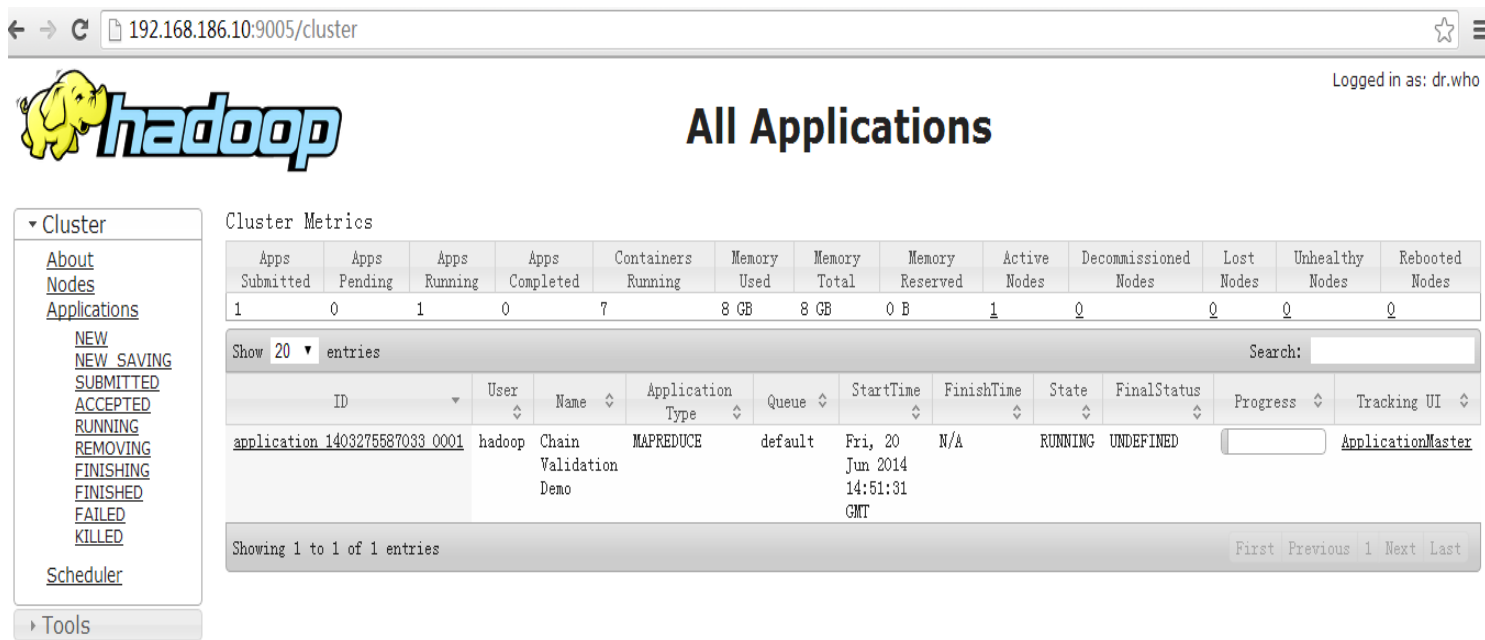
查看Job运行历史

<http://master:50070>

查看HDFS信息

➤ 性能监控 - HBase Web UI(1)

HBase提供了一种Web方式的用户接口，可用于查看HBase集群属性、表信息、RegionServer信息和ZooKeeper状态信息等。



192.168.186.10:9005/cluster

Logged in as: dr.who

hadoop

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	1	0	7	8 GB	8 GB	0 B	1	0	0	0	0

Show 20 entries


ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application 1403275587033 0001	hadoop	Chain Validation Demo	MAPREDUCE	default	Fri, 20 Jun 2014 14:51:31 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaster

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

<http://master:60010>查看HBase集群信息

➤ 性能监控 - HBase Web UI (2)



192.168.186.10:9005/cluster

Logged in as: dr.who

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes
1	0	1	0	7	8 GB	8 GB	0 B	1	0	0

Show 20 entries


ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress
application_140327587033_0001	hadoop	Chain Validation Demo	MAPREDUCE	default	Fri, 20 Jun 2014 14:51:31 GMT	N/A	RUNNING	UNDEFINED	

Showing 1 to 1 of 1 entries

Cluster

- About
- Nodes
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- REMOVING
- FINISHING
- FINISHED
- FAILED
- KILLED
- Scheduler
- Tools

<http://master:60030>查
看RegionServer运行状态



192.168.186.10:9005/cluster

Logged in as: dr.who

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	1	0	7	8 GB	8 GB	0 B	1	0	0	0	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_140327587033_0001	hadoop	Chain Validation Demo	MAPREDUCE	default	Fri, 20 Jun 2014 14:51:31 GMT	N/A	RUNNING	UNDEFINED		ApplicationMaster

Showing 1 to 1 of 1 entries

Cluster

- About
- Nodes
- Applications
- NEW
- NEW SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- REMOVING
- FINISHING
- FINISHED
- FAILED
- KILLED
- Scheduler
- Tools

<http://master:60010/zk.jsp>查看
HBase 在ZooKeeper中的注册信息

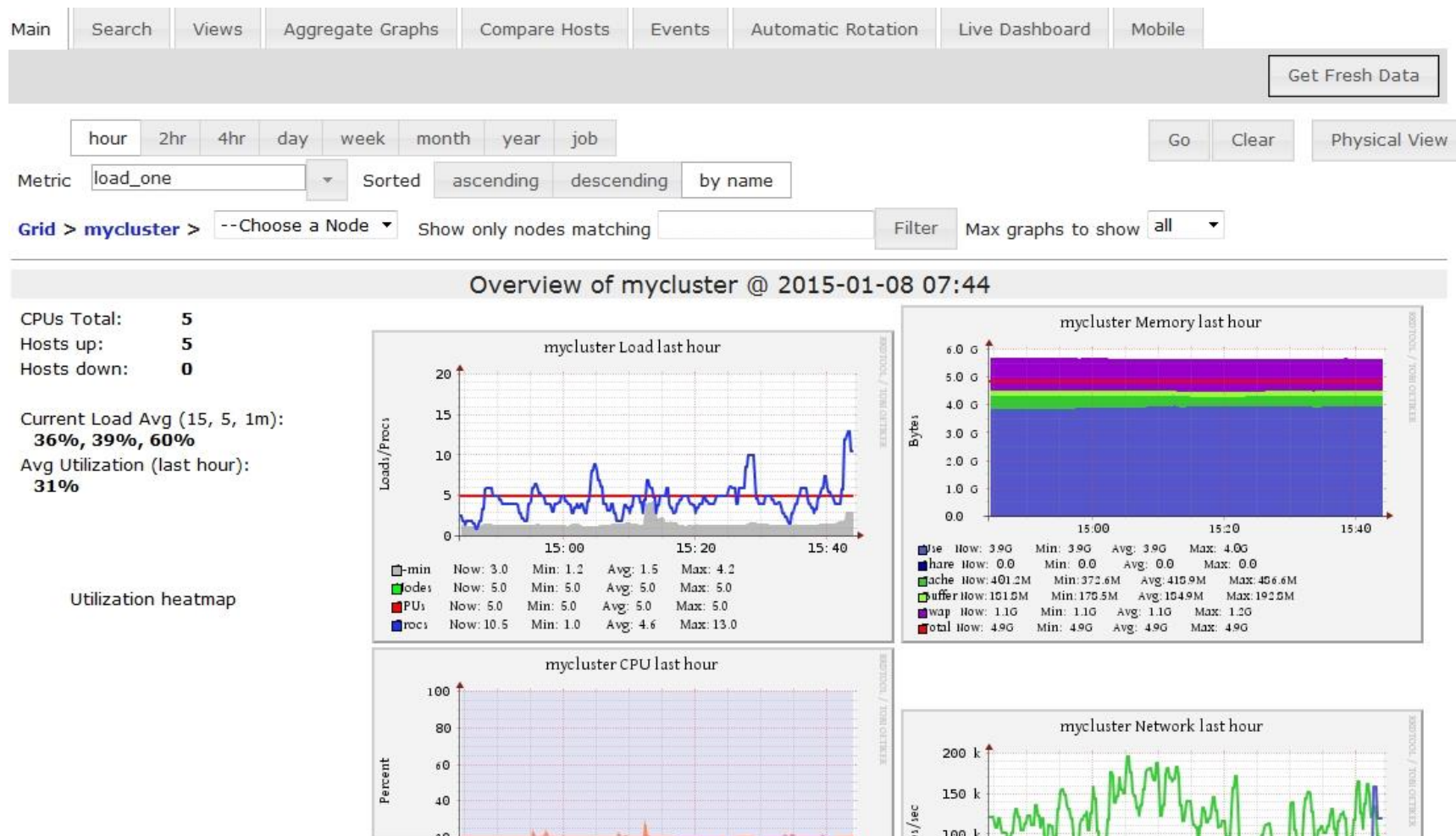
➤ 性能监控 - Ganglia

Ganglia是UC Berkeley发起的一个开源集群监视项目 (<http://ganglia.info/>)。它可以用来监视和显示上千节点集群中的节点的各种状态信息，比如：CPU、MEMORY、硬盘利用率，I/O负载、网络流量情况等，同时可以将历史数据以曲线方式通过php页面呈现。



通过将Ganglia和Hadoop集群整合后，可以查看Hadoop集群中每个Master/Slave节点的运行状态。

➤ 性能监控 - Ganglia



■ 附录：本课件部分参考书籍

