Hive开发

课程目标

1. 了解基于hadoop的数据平台架构
2. 了解Hive的应用环境、作用及原理

培训目录

培训目录

前期课程内容回顾

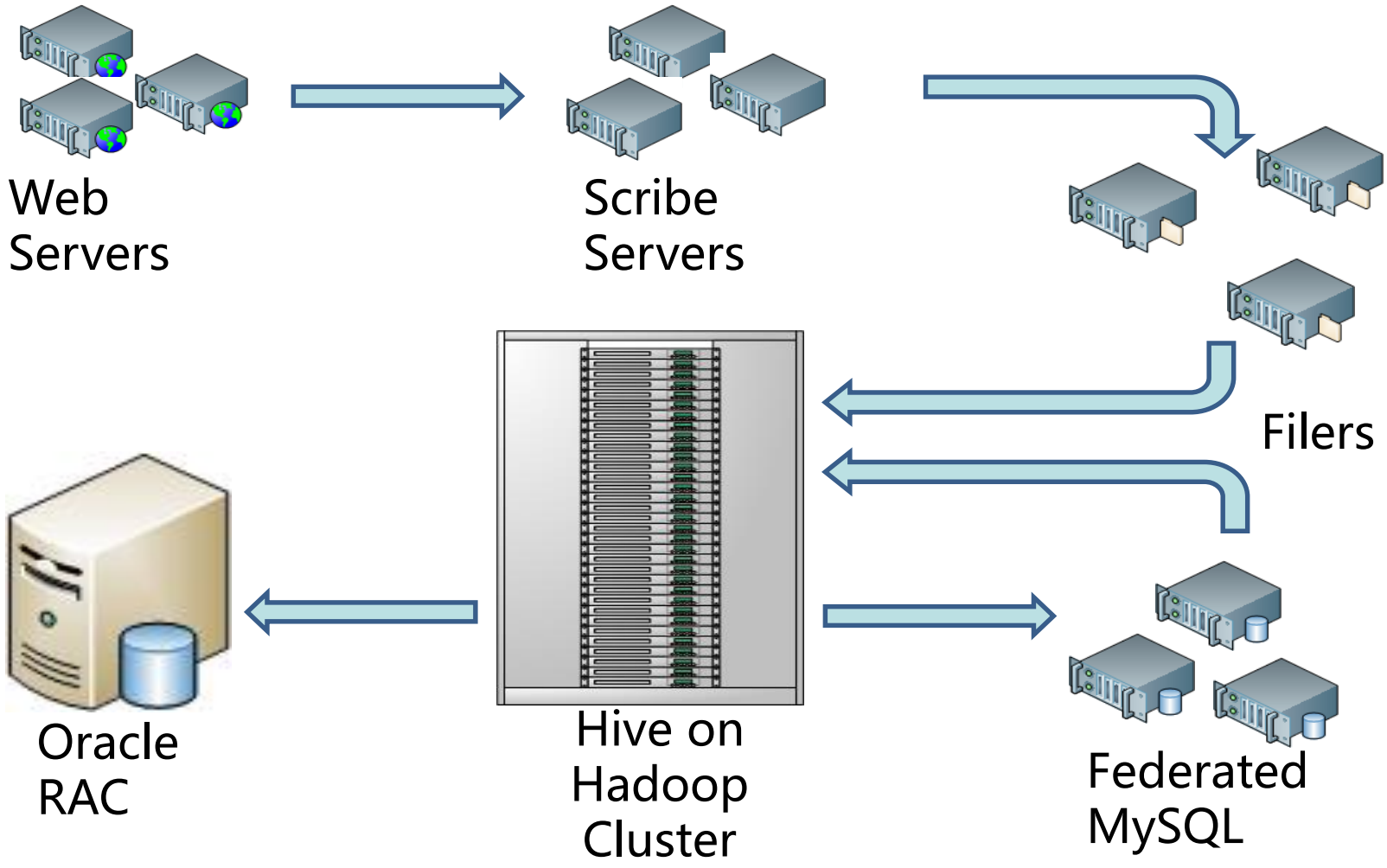
数据仓库：Hive

培训目录

1 前期课程内容回顾

2 数据仓库：Hive

Data Warehousing at Facebook



What is Hive

- 构建于Hadoop的HDFS和MapReduce上，用于管理和查询结构化/非结构化数据的数据仓库

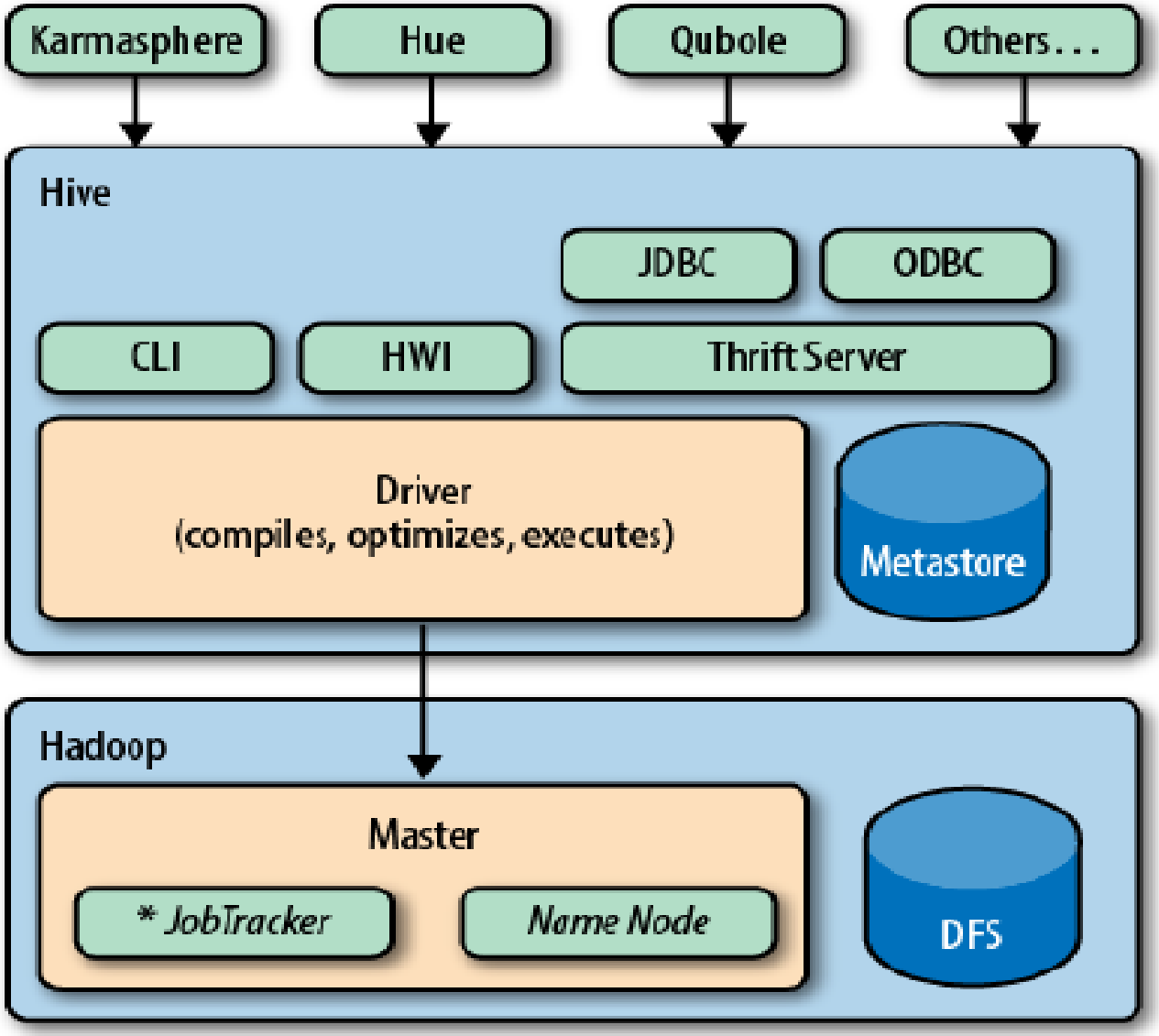
✓ 使用HQL作为查询接口

✓ 使用HDFS作为底层存储

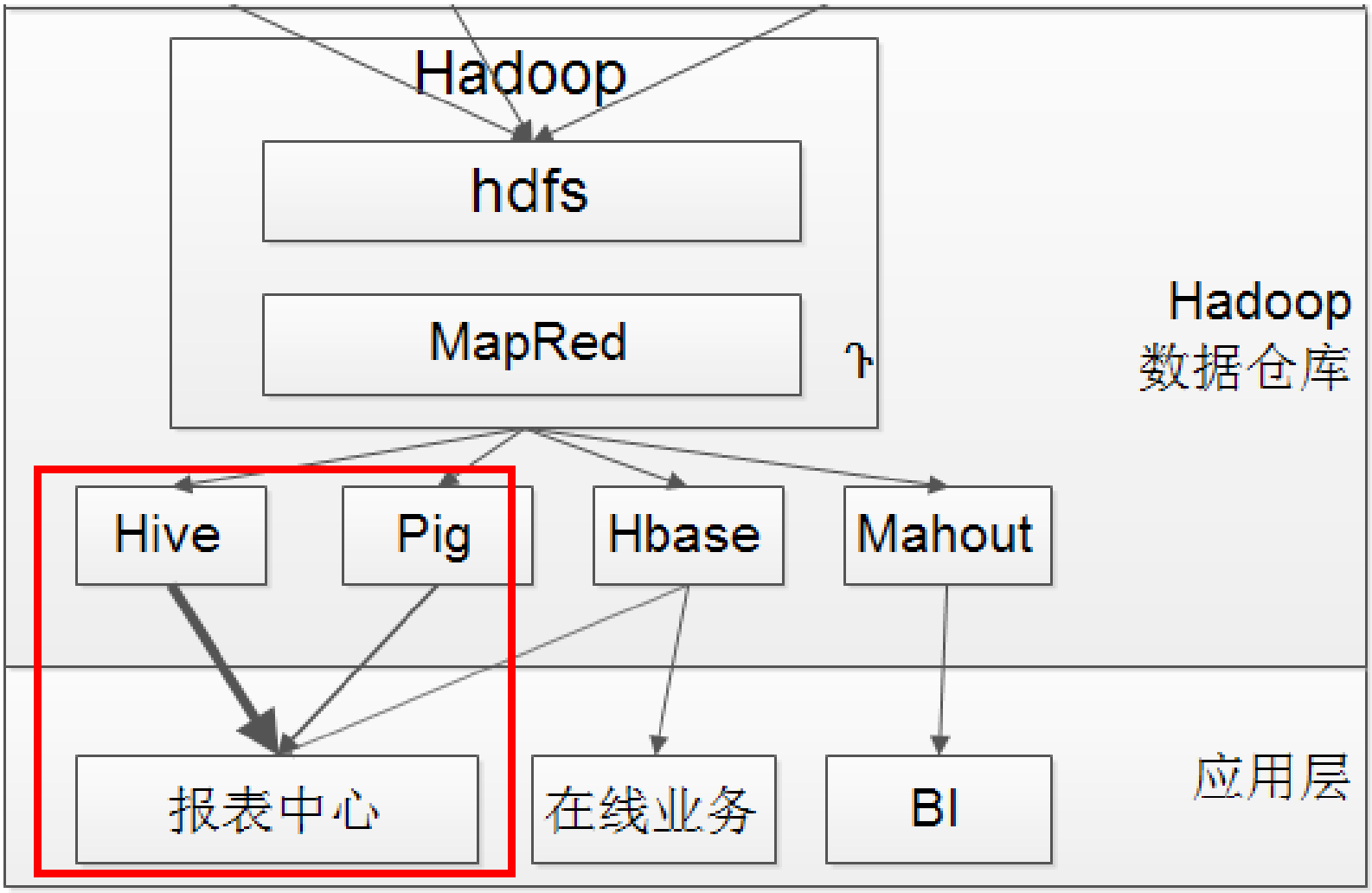
✓ 使用MapReduce作为执行层


HIVE

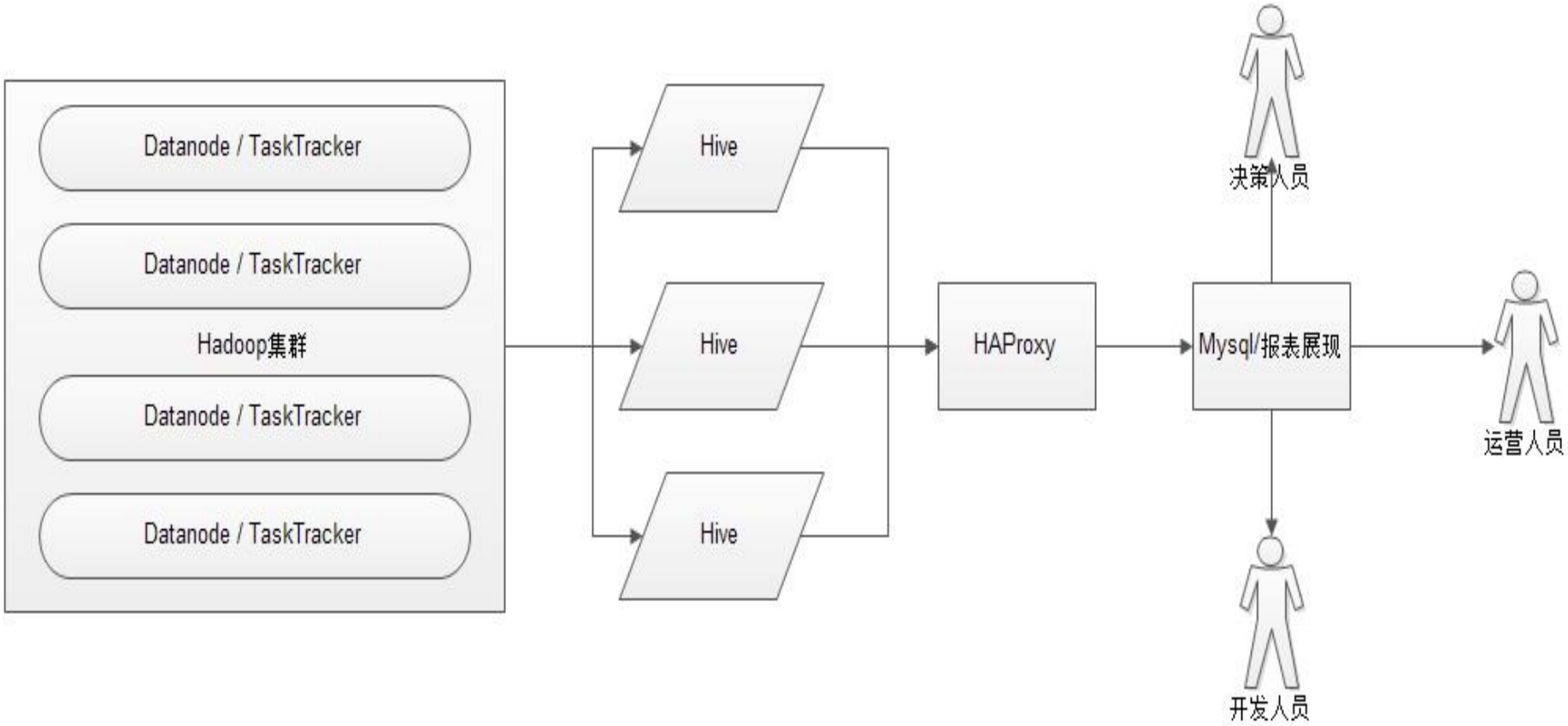
What is Hive


Karmasphere
Hue
Qubole
Others...
Hive
JDBC
ODBC
CLI
HWI
Thrift Server
Driver
(compiles, optimizes, executes)
Metastore
Hadoop
Master
* JobTracker
Name Node
DFS

What is Hive


Hadoop
hdfs
MapRed
Hadoop
数据仓库
Hive
Pig
Hbase
Mahout
报表中心
在线业务
BI
应用层

Hive 应用


Datanode / TaskTracker
Datanode / TaskTracker
Hadoop集群
Datanode / TaskTracker
Datanode / TaskTracker
Hive
Hive
Hive
HAProxy
Mysql/报表展现
决策人员
运营人员
开发人员

Hive Application Cases

某视频播放和搜索指数

Hive Application Cases

某视频播放和搜索指数

Hive Application Cases

相关搜索

相关搜索 | 太极

相关搜索词		上升最快相关搜索词	
1	太极2	太极2英雄崛起	大于500%
2	太极1	太极	大于500%
3	太极2英雄崛起	太极2英雄崛起完...	大于500%
4	太极2英雄崛起完...	太极2英雄崛起完...	大于500%
5	太极2英雄崛起完...	太极2	大于500%
6	太极1从零开始	太极1从零开始	大于500%
7	太极2英雄崛起在...	太极1	440%
8	太极1从零开始2	太极2英雄崛起在...	430%
9	皇太极秘史	太极1从零开始2	340%
10	太极2从零开始完...	皇太极秘史	320%

Hive Application Cases

相关播放

相关播放 | 太极

总播放	正片播放	预告片/花絮播放	集均播放
10,506,137	10,506,137	0	420,245

正片 | 预告片/花絮

名称	播放数	顶	踩	评论	收藏	引用
太极 01	1,398,680	3,586	454	285	307	23,433
太极 02	651,126	1,600	168	229	63	5,152
太极 03	573,310	1,427	171	208	40	4,118
太极 04	525,656	1,313	130	242	37	3,712
太极 05	511,797	1,203	135	280	28	3,810
太极 06	451,932	976	109	144	25	3,266
太极 07	417,588	813	86	129	21	2,985
太极 08	404,026	872	96	123	23	2,918
太极 09	387,360	847	85	109	19	2,695
太极 10	380,503	830	95	141	25	2,691

第1-10/25条

1 2 3 ‹上一页 下一页›

Hive vs SQL

对比项目	Hive	SQL
数据插入	支持批量导入	支持单条和批量导入
数据更新	不支持	支持
索引	支持	支持
分区	支持	支持
执行延迟	高	低
扩展性	好	有限

Hive简介

Hive可以被认为是一种数据仓库，包括数据的存储以及查询Hive包括一个高层语言的执行引擎，类似于SQL的执行 Hive建立在Hadoop的其它组成部分之上，包括Hive依赖于HDFS进行数据保存，依赖于MapReduce完成查询操作。

Hive最初的开发由Facebook推动，在Facebook内部每天会搜集大量的数据，并需要在这些数据上进行大量分析最初的分析是通过手工的python脚本形式进行数据分析的数量十分巨大，在2006年每天需要分析数十个10 GB左右的数据，在2007年增长到大约TB的量级，现在数据分析的数量可能是这个数量的10倍。

使用Hadoop进行数据分析

- 上述的分析任务可以通过Hadoop集群进行，即可以通过Hadoop集群将任务分布到数百甚至上千个节点中进行分析，通过并行执行锁定分析的时间。
- 如果原始数据使用数据库形式的话，则需要进行数据的转换，将数据存储到分布式文件系统HDFS中，然后通过MapReduce程序进行分析。

- Hadoop通过MapReduce的并行化方式进行并行处理，能够充分利用数目庞大的分析机器。但是，MapReduce是一个底层的编程接口，对于数据分析人员来说，这个编程接口并不是十分友好，还需要进行大量的编程以及调试工作。

在Hadoop上加入数据分析的功能

显然，为了能够支持一个类似于SQL相关的数据查询语言，Hadoop还需要加入一些额外的模块才能够方便数据分析人员的使用，这些额外的模块包括：

- 数据查询语言本身的定义与构造，这是与终端用户进行交互的接口，最简单的可以通过命令行接口的方式展开用户与系统的交互
- 构造数据查询语言的执行引擎，即将上述的查询语言进行编译，并通过分布式的执行引擎完成查询，在Hive中，执行引擎会将查询语言翻译为多个MapReduce的任务序列，交给MapReduce程序去执行
- 数据查询语言本身需要定义一套数据组织的格式

Hive的组成模块（1）

Hive的模块非常类似于传统的数据库的模块，下面是 Hive 的必要组成模块以及对应的功能介绍

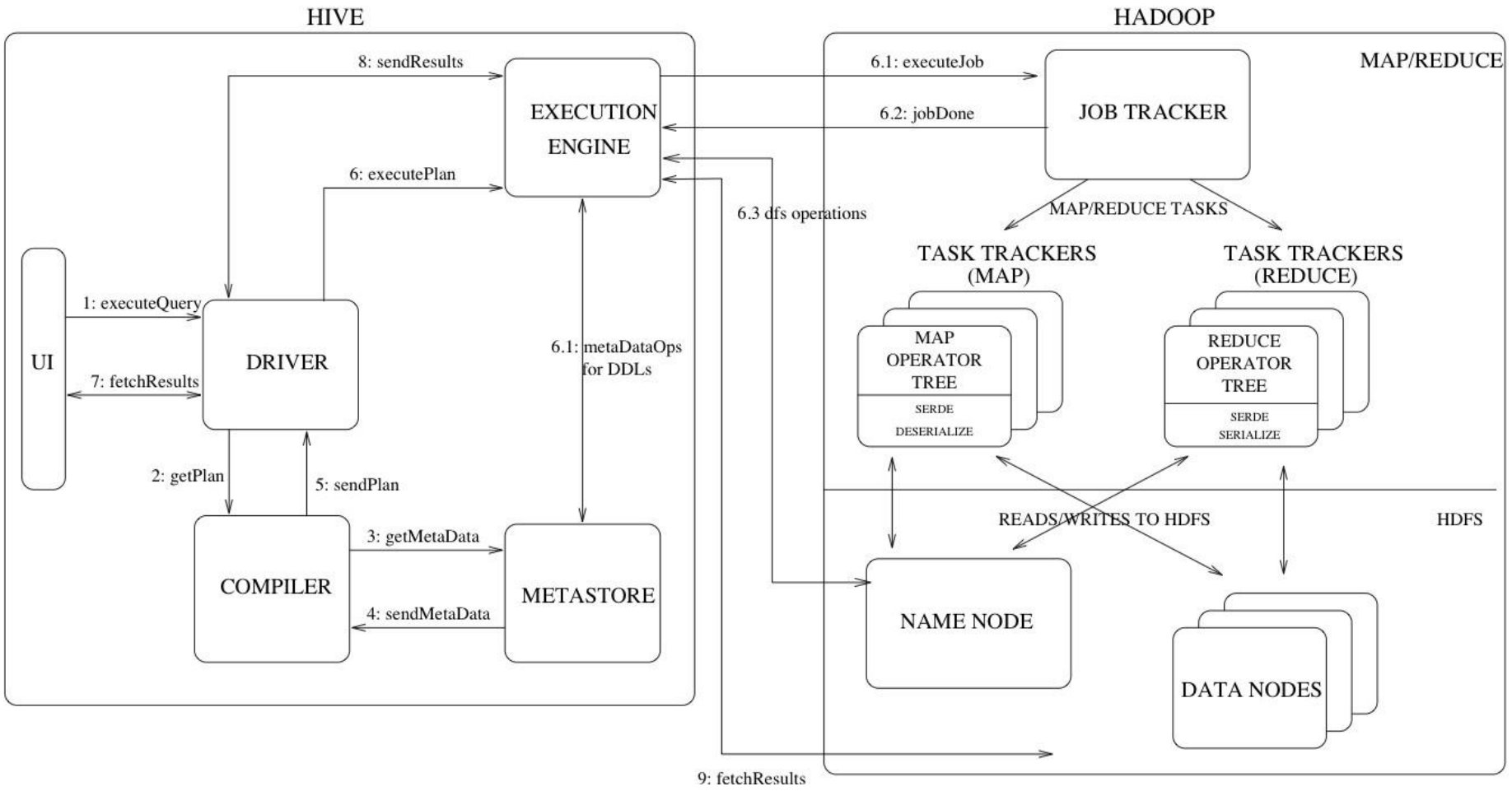
- HiveQL：这是Hive的数据查询语言，与SQL非常类似。Hive提供了这个数据查询语言与用户的接口，包括一个 shell 的接口，可以进行用户的交互,以及网络接口与 JDBC接口。
- JDBC接口可以用于编程，与传统的数据库编程类似， 使得程序可以直接使用Hive功能。
- Driver: 执行的驱动，用以将各个组成部分形成一个有机的执行系统，包括会话的处理，查询获取以及执行驱动。

Hive的组成模块（2）

- Compiler：Hive需要一个编译器，将HiveQL语言编译成中间表示，包括对于HiveQL语言的分析，执行计划的生成以及优化等工作
- Execution Engine：执行引擎，在Driver的驱动下，具体完成执行操作，包括MapReduce执行，或者HDFS操作，或者元数据操作
- Metastore：用以存储元数据：存储操作的数据对象的格式信息，在HDFS中的存储位置的信息以及其他的用于数据转换的信息SerDe等


HIVE

Hive的系统结构


HIVE
HADOOP
MAP/REDUCE
8: sendResults
EXECUTION ENGINE
6.1: executeJob
6.2: jobDone
JOB TRACKER
6: executePlan
6.3 dfs operations
MAP/REDUCE TASKS
TASK TRACKERS (MAP)
TASK TRACKERS (REDUCE)
UI
1: executeQuery
7: fetchResults
DRIVER
6.1: metaDataOps for DDLs
MAP OPERATOR TREE
SERDE
DESERIALIZE
REDUCE OPERATOR TREE
SERDE
SERIALIZE
2: getPlan
5: sendPlan
READS/WRITES TO HDFS
HDFS
3: getMetaData
COMPILER
METASTORE
4: sendMetaData
NAME NODE
DATA NODES
9: fetchResults

Hive的数据模型

每一个类似于数据库的系统都首先需要定义一个数据模型，然后才是在这个数据模型之上的各种操作

- Tables：Hive的数据模型由数据表组成
  - ✓ 数据表中的列是有类型的（int, float, string, data, boolean）
  - ✓ 也可以是复合的类型，如list: map （类似于JSON形式的数据）
- Partitions：数据表可以按照一定的规则进行划分Partition
  - ✓ 例如，通过日期的方式将数据表进行划分
- Buckets：数据存储的桶

元数据存储：Metastore

在Hive中由一系列的数据表格组成一个名字空间，关于这个名字空间的描述信息会保存在Metastore的空间中

元数据使用SQL的形式存储在传统的关系数据库中，因此可以使用任意一种关系数据库，例如Derby(apache的关系数据库实现)，MySQL以及其他的多种关系数据库存储方法

数据的物理分布情况

Hive在HDFS中有固定的位置，通常被放置在HDFS的如下目录中

/home/hive/warehouse

每个数据表被存放在warehouse的子目录中

- 数据划分Partition，数据桶Buckets形成了数据表的子目录

上述数据实际上是数据表文件的元数据，保存在HDFS的NameNode中，实际数据则存储在DataNode中，可能以任意一种形式存储，例如：

- 使用分隔符的文本文件，或者是SequenceFile
- 使用用户自定义的形式

Hive系统的配置

要想Hive使用HDFS进行数据仓库的存储，使用MapReduce进行HQL语言的执行，需要进行相应的配置。

首先，需要创建Hive的配置文件hive-site.xml(注意，Hive安装包中包含一个配置文件模板—hive-default.xml.template）。

- fs.default.name用以告知Hive对应的HDFS文件系统的名字节点在什么位置
- mapred.job.tracker用以告知Hive对应的MapReduce处理系统的任务管理器在什么位置

通过上面两点的配置，就可以让Hive系统与对应的HDFS以及MapReduce进行交互，完成数据的存取以及查询的操作

Hive部署及安装

1. 下载tar.gz，解压缩
2. 设置HADOOP_HOME
3. 为什么使用JDBC连接数据库做Metastore
4. 访问Hive 通过 thrift或JDBC
5. Hive与其他数据库的对接与数据ETL

Hive部署及安装

hive 的元数据存储

- hive默认使用内存数据库derby存储元数据，使用时不需要修改任何配置，缺点：hive server重启后所有的元数据都会丢失。
- hive支持mysql、oracle等任何支持JDBC连接方式的数据库来存储元数据，需要修改相应的配置项

Hive部署及安装

```
<property>
  <name>hive.metastore.local</name>
  <value>false</value>
</property>
```

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://db1.mydomain.pvt/hive_db?createDatabaseIfNotExist=true</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.jdbc.Driver</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>database_user</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>database_pass</value>
</property>
```

启动Hive的命令行界面shell

完成Hive系统的合适配置之后，打开任意一个命令行界面，执行下面的命令就可以启动hive的界面

$cd /hive/install/directory 进入hive的安装目录，如果通过IDH安装，并配置好环境变量，则不需要这一步

hive

hive>

等待用户输入查询命令

HiveQL

1. Select * from db.table1
2. Select count distinct uid from db.table1
3. 支持select、union all、join（left、right、mapjoin）like、where、各种聚合函数、支持json解析
4. 类SQL，并不完全支持

创建数据表的命令

hive> show tables;

显示所有的数据表

hive> create table shakespeare (freq int, word string) row format delimited fields terminated by '\t' stored as textfile;

创建一个表，这个表包括两列，分别是整数类型以及字符串类型，使用文本文件表达，数据域之间的分隔符为\t

hive> describe shakespeare;

显示所创建的数据表的描述，即创建时候对于数据表的定义

HiveQL

1. Hive建表语法

Create Table

```
CREATE [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name data_type [COMMENT col_comment], ...)]
  [COMMENT table_comment]
  [PARTITIONED BY (col_name data_type [COMMENT col_comment], ...)]
  [CLUSTERED BY (col_name, col_name, ...) [SORTED BY (col_name [ASC|DESC], ...)] INTO num_buckets BUCKETS]
  [SKEWED BY (col_name, col_name, ...) ON ([(col_value, col_value, ...), ...|col_value, col_value, ...]) (N
  [
   [ROW FORMAT row_format] [STORED AS file_format]
   | STORED BY 'storage.handler.class.name' [WITH SERDEPROPERTIES (...)]  (Note: only available starting wi
  ]
  [LOCATION hdfs_path]
  [TBLPROPERTIES (property_name=property_value, ...)]  (Note: only available starting with 0.6.0)
  [AS select_statement]  (Note: this feature is only available starting with 0.5.0, and is not supported wh
```

HiveQL

hive建表语法格式

- external 外部表，类似于mysql的csv引擎
- partitioned by 指定分区字段
- clustered by sorted by 可以对表和分区对某个列进行分桶操作，也可以利用sorted by对某个字段进行排序
- row format指定数据行中字段间的分隔符和数据行分隔符
- stored as 指定数据文件格式：textfile sequence rcfile inputformat (自定义的inputformat 类)
- location 指定数据文件存放的hdfs目录

HiveQL

hive建表语句

```
CREATE TABLE page_view
  (viewTime INT, userid BIGINT, page_url STRING, referrer_url
  STRING, ip STRING COMMENT 'IP Address of the User')
  COMMENT 'This is the page view table'
  PARTITIONED BY(dt STRING, country STRING)
  CLUSTERED BY(userid) SORTED BY(viewTime) INTO 32 BUCKETS
  ROW FORMAT DELIMITED
    FIELDS TERMINATED BY '\001'
   COLLECTION ITEMS TERMINATED BY '\002'
   MAP KEYS TERMINATED BY '\003'
STORED AS SEQUENCEFILE;
```

HiveQL

删除表

- drop table [IF EXISTS] table_name
- 删除内部表时会删除元数据和表数据文件
- 删除外部表（external）时只删除元数据

HiveQL

- **修改表 增加分区**

  ALTER TABLE page_view ADD PARTITION (dt='2008-08-08', country='us') location '/path/to/us/part080808' PARTITION (dt='2008-08-09', country='us') location '/path/to/us/part080809';

HiveQL

- **修改表 删除分区**

  ALTER TABLE page_view DROP PARTITION (dt='2008-08-08', country='us');

- **修改表 重命名表**

  ALTER TABLE table_name RENAME TO new_table_name

- **修改表 修改字段**

  ALTER TABLE test_change CHANGE a a1 STRING AFTER b;

HiveQL

加载数据

LOAD DATA INPATH '/user/myname/kv2.txt'
OVERWRITE INTO TABLE invites PARTITION
(ds='2008-08-15');

加载（本地、hdfs）文件到指定的表 分区

FROM src
INSERT OVERWRITE TABLE dest1 SELECT src.*
WHERE src.key < 100
INSERT OVERWRITE TABLE dest2 SELECT src.key,
src.value WHERE src.key >= 100 and src.key < 200

从指定表中选取数据插入到其他表中

装入数据

数据装入到Hive中

hive> load data inpath “shakespeare_freq” into table shakespeare;

HiveQL

select 语法结构

```
SELECT [ALL | DISTINCT] select_expr, select_expr,
...
    FROM table_reference
    [WHERE where_condition]
    [GROUP BY col_list]
    [ CLUSTER BY col_list | [DISTRIBUTE BY col_list]
[SORT BY col_list] ]
    [LIMIT number]
```

HiveQL

select 案例

```
SELECT pv.pageid, u.age FROM page_view p JOIN
user u ON (pv.userid = u.userid) JOIN newuser x on
(u.age = x.age);
```

数据查询select语句

```
hive> select * from shakespeare limit 10;
```

```
hive> select * from shakespeare where freq > 100
sort by freq asc limit 10;
```

查询语句，获取频率最高的单词

```
hive> select freq, count(1) as f2 from shakespeare
group by freq sort by f2 desc limit 10;
```

```
hive> explain select freq, count(1) as f2 from
shakespeare group by freq sort by f2 desc limit 10;
```

Hive中的连接操作

Hive中的连接

内连接

外连接

Map连接

数据表举例

hive> **SELECT * FROM sales;**

Joe 2

Hank 4

Ali 0

Eve 3

Hank 2

hive> **SELECT * FROM things;**

2 Tie

4 Coat

3 Hat

1 Scarf

内连接的结果

hive> **SELECT sales.*, things.* FROM sales JOIN things ON (sales.id = things.id);**

Joe 2 2 Tie

Hank 2 2 Tie

Eve 3 3 Hat

Hank 4 4 Coat Hive中只支持等值连接，在谓

词中只能使用等号

左外连接的结果

- hive> **SELECT sales.*, things.***
- > **FROM sales LEFT OUTER JOIN things ON (sales.id**
- **= things.id);**
- Ali 0 NULL NULL
- Joe 2 2 Tie
- Hank 2 2 Tie
- Eve 3 3 Hat

左表中的所有数据（没有对应匹配的数据在表things中）

- Hank 4 4 Coat

右外连接的结果

- hive> **SELECT sales.*, things.***
- > **FROM sales RIGHT OUTER JOIN things ON**
- **(sales.id = things.id);**
- NULL NULL 1 Scarf
- Joe 2 2 Tie
- Hank 2 2 Tie
- Eve 3 3 Hat
- Hank 4 4 Coat

返回所有在things里面出现的内容

全外连接

- hive> **SELECT sales.*, things.***
- > **FROM sales FULL OUTER JOIN things ON (sales.id**
- **= things.id);**
- Ali 0 NULL NULL

NULL NULL 1 Scarf

- Joe 2 2 Tie
- Hank 2 2 Tie

Hank 4 4 Coat

- Eve 3 3 Hat

Hive Extension

- **User-defined Function**
  - ✓ UDF 作用于单个数据行，输出一个数据 ，如字符处理函数
  - ✓ UDAF 作用于多个数据行，输出一个数据，如count，sum函数
  - ✓ UDTF 作用于单个数据行，输出多个数据行
  - ✓ 支持用户使用java自定义开发UDF函数
- **Hive streaming**
  - ✓ 支持用户在hive QL语句中嵌入自定义的streaming处理脚本

Hive UDF

```
add jar /tmp/helloUDF.jar;
create temporary function helloworld as
    'com.hrj.hive.udf.HelloUDF';
select helloworld(t.col1) from t limit 10;
drop temporary function helloworld;
```

Hive 优化策略思路

- 使用 Partition 减少扫描数量
- 使用Map端Join
- 配置Reduce数量
- 使用 INSERT INTO LOCAL DIRECTORY ‘/home/me/pv_age_sum.dir’ ,而非适用 HiveServer 。

Hive 优化策略思路

- 使用 Partition 减少扫描数量
- 使用Map端Join
- 配置Reduce数量
- 使用 INSERT INTO LOCAL DIRECTORY ‘/home/me/pv_age_sum.dir’ ,而非适用 HiveServer 。

Hive JDBC编程

Hive JDBC

Hive 支持了标准的数据库查询接口JDBC 在

JDBC中需要指定驱动字符串以及连接字符串

驱动器字符串为“org.apache.hadoop.hive.jdbc.HiveDriver”，即在Hive的软件包中已经加入了对应的JDBC的驱动程序

连接字符串则是标志了对应Hive服务器，例如"jdbc:hive://master:10000/default"

之后，可以直接使用传统的JDBC编程技术去访问Hive所提供的功能

Hive JDBC程序举例

```
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;

public class HiveExample {
 private static String driverName = "org.apache.hadoop.hive.jdbc.HiveDriver";
 public static void main(String[] args)throws SQLException {
  try {
       Class.forName(driverName);
     } catch (ClassNotFoundException e) {
      System.out.println("classForNameError");
       System.exit(1);
     }
     Connection con = DriverManager.getConnection("jdbc:hive://master:10000/default", "", "");
     Statement stmt = con.createStatement();

     String sql = "select * from merged limit 100" ;
     System.out.println("Running: " + sql);
     ResultSet res = stmt.executeQuery(sql);
     while (res.next()) {
      System.out.println(String.valueOf(res.getString(1)) + "\t" + res.getString(2));
     }
 }
}
```

Hive学习参考资料

- 参考资料：
- http://trac.nchc.org.tw/cloud/
- http://www.cloudera.com/downloads/sqoop/
- 查询界面：
- http://www.phphiveadmin.net
- 任务调度：
- http://incubator.apache.org/oozie/
- http://www.cronhub.com/
- http://sna-projects.com/azkaban/screenshots.php

Hive学习参考资料

- 数据交换：
- http://code.taobao.org/p/datax/wiki/DataX%E4%BA%A7%E5%93%81%E8%AF%B4%E6%98%8E/
- https://github.com/zhuyeqing/ComETL
- http://sqoop.apache.org/
- http://archive.cloudera.com/cdh/3/sqoop/SqoopUserGuide.html
- hadoop windows 版本：
- http://trac.nchc.org.tw/cloud/wiki/Hadoop4Win

总结与提问

Hive参数优化

mapred.reduce.tasks = -1 [每任务最大使用reduce数]

hive.exec.reducers.max = 999 [Hive最大使用reduce槽位数]

hive.exec.local.scratchdir = /tmp/${user.name} [本地日志路径]

javax.jdo.option.ConnectionURL = jdbc:mysql://192.168.1.44:3306/hive?createDatabaseIfNotExist=true

Hive参数优化

[元数据保存位置] 默认Derby，建议mysql或pgsql或oracle

javax.jdo.option.ConnectionDriverName=com.mysql.jdbc.Driver

[jdbc驱动]javax.jdo.option.ConnectionUserName = hive

[jdbc 用户名]javax.jdo.option.ConnectionPassword = hive [jdbc密码]

javax.jdo.option.Multithreaded = true [jdbc采用多线程并发]

Hive参数优化

hive.groupby.skewindata = true [group by 数据倾斜校正] 默认false

hive.map.aggr = true [Map聚合，相当于Combiner]

hive.optimize.groupby = true [优化group by]

hive.optimize.union.remove = true [union在大量union情况下 优化] 默认false需要partition表

hive.exec.parallel = true [并行执行嵌套select] 默认false

hive.exec.parallel.thread.number = 12 [并行最大线程数]

Hive其他优化 – 排序优化

排序优化

Order by 实现全局排序，一个reduce实现，效率低

Sort by 实现部分有序，单个reduce输出的结果是有序的，效率高，通常和DISTRIBUTE BY关键字一起使用（DISTRIBUTE BY关键字 可以指定map 到 reduce端的分发key）

CLUSTER BY col1 等价于DISTRIBUTE BY col1 SORT BY col1.

Hive其他优化 –合并小文件

合并小文件

文件数目过多，会给 HDFS 带来压力，并且会影响处理效率，可以通过合并 Map 和 Reduce 的结果文件来消除这样的影响

hive.merge.mapfiles = true是否和并 Map 输出文件，默认为 True

hive.merge.mapredfiles = false是否合并 Reduce 输出文件，默认为 False

hive.merge.size.per.task = 256*1000*1000合并文件的大小。

Hive其他优化 – join优化

- **join优化**

✓ Join查找操作的基本原则：应该将条目少的表/子查询放在 Join 操作符的左边。原因是在 Join 操作的 Reduce 阶段，位于 Join 操作符左边的表的内容会被加载进内存，将条目少的表放在左边，可以有效减少发生内存溢出错误的几率

✓ Join查找操作中如果存在多个join，且所有参与join的表中其参与join的key都相同，则会将所有的join合并到一个mapred程序中。

Hive其他优化 – join优化

- 案例：

✓ SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key1）在一个mapre程序中执行join

✓ SELECT a.val, b.val, c.val FROM a JOIN b ON (a.key = b.key1) JOIN c ON (c.key = b.key2) 在两个mapred程序中执行join

✓ Map join的关键在于join操作中的某个表的数据量很小，案例：

✓ SELECT /*+ MAPJOIN(b) */ a.key, a.value

✓ FROM a join b on a.key = b.key

Hive其他优化 - 使用优化

- **执行队列优化：**
- ✓ set mapred.job.queue.name=name;
- **使用优化：**
- ✓ 大小表，大大表join？
- ✓ 把重复关联少的表放在join前面
- ✓ 小维度group by？
- ✓ 用sum() group by的方式来替换count(distinct)
- ✓ 外部表vs内部表

Hive其他优化 - 并行执行

- **嵌套SQL并行执行优化**

  set hive.exec.parallel=true;

  set hive.exec.parallel.thread.number=16;

  效率可提升至少100%

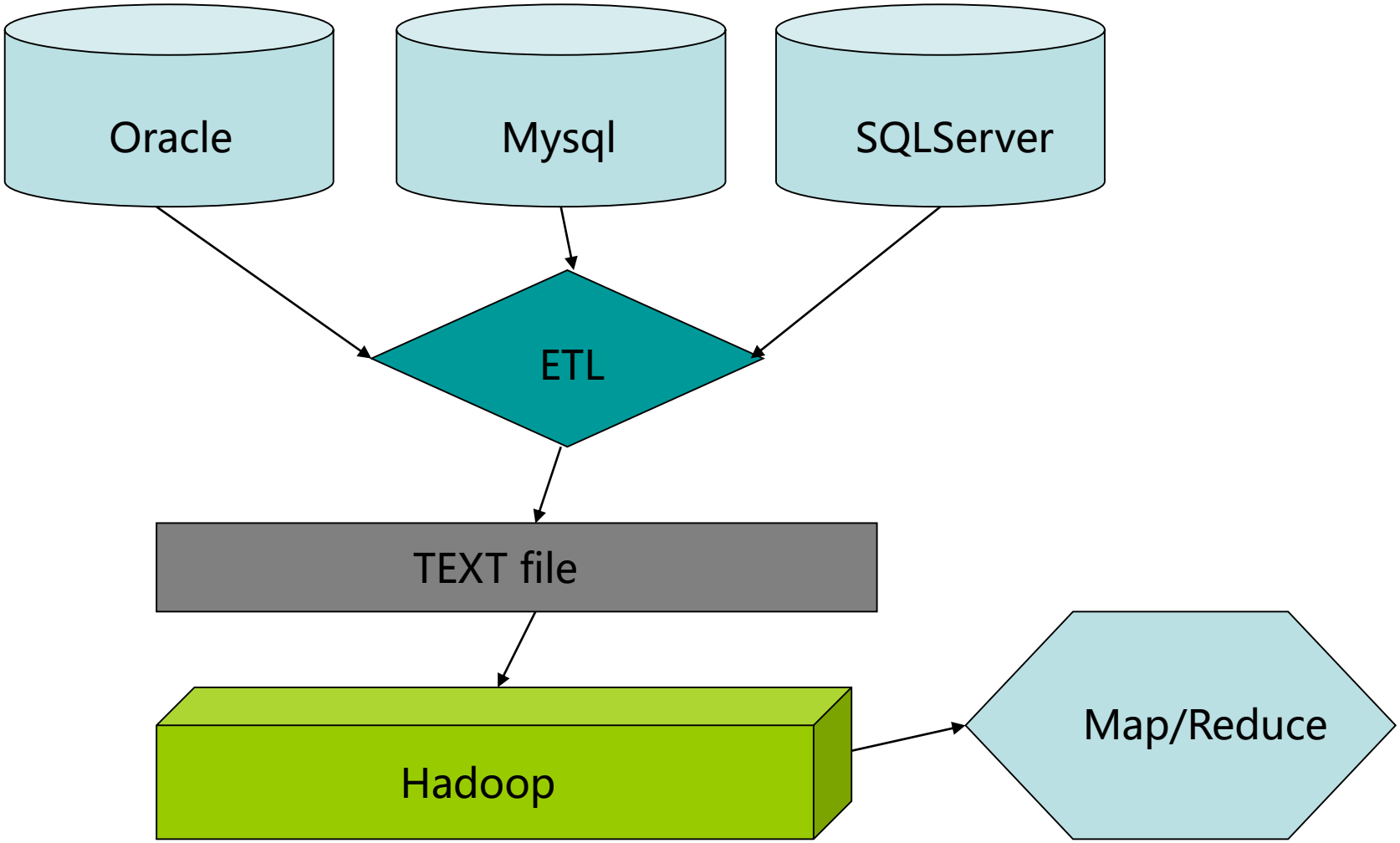
  例如：某job需要11个stage

  ✓非并行35分钟

  ✓并行8个job执行10分钟

  ✓并行16个job执行6分钟

自行编写ETL工具思路

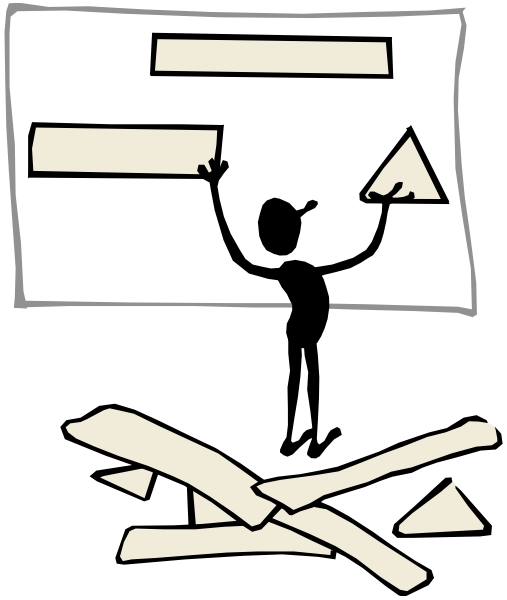


自行编写ETL工具思路

★一切皆配置文件

★将数据库导出成map/reduce可操作的格式，如TEXT或CSV，再put到HDFS

★在数据库，mapred间相互转换



Question