星环大数据平台

# 环境准备

（参考星环官网文档与支持>产品文档>TDH5.2安装手册）

## 服务器信息

仿真环境六台服务器：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IP | 主机名 | 操作系统 | 处理器 | 内存 | 硬盘 |
| 10.64.141.30 | bak-BigDataNode01 | RedHat6.5 | 8Core/2.20GHz | 64G | 300G |
| 10.64.141.31 | bak-BigDataNode2 | RedHat6.5 | 8Core/2.20GHz | 64G | 300G |
| 10.64.141.32 | bak-BigDataNode3 | RedHat6.5 | 8Core/2.20GHz | 32G | 300G |
| 10.64.141.33 | bak-BigDataNode4 | RedHat6.5 | 8Core/2.20GHz | 32G | 300G |
| 10.64.141.34 | bak-BigDataNode5 | RedHat6.5 | 8Core/2.20GHz | 32G | 300G |
| 10.64.141.35 | bak-BigDataNode6 | RedHat6.5 | 8Core/2.20GHz | 32G | 300G |

TDH支持的JDK版本：

JDK1.7（无须自行安装，TDH安装会自动安装，如果有其它版本可能会冲突）

## 安装前检查

磁盘分区要求

内存容量要求

网络设置

NTP服务器设置

## 系统配置

修改每个节点的/etc/hosts文件，确保hostname和ip地址映射：

10.64.141.30 bak-BigDataNode01

10.64.141.31 bak-BigDataNode2

10.64.141.32 bak-BigDataNode3

10.64.141.33 bak-BigDataNode4

10.64.141.34 bak-BigDataNode5

10.64.141.35 bak-BigDataNode6

登入到节点使用 hostname 命令检查节点的名字，并保证名字和上述/etc/hosts里设置的名字相同；

/etc/hosts文件的第一行必须为127.0.0.1的记录，不能将此行注释掉，也不能将当前主机名写在该行中。通常如下：

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

# TDH5.2平台安装

## 安装介质

将OS的ISO文件事先挂载到服务器上；

解压客户服务提供的压缩包，上传到服务器上；

TDH-Image-Registry-Transwarp-5.2.0-final.tar.gz

TDH-Basic-Component-Transwarp-5.2.0-final.tar.gz

TDH-Machine-Learning-Transwarp-5.2.0-final.tar.gz

执行命令开始安装：

tar xvzf TDH-Image-Registry-Transwarp-5.2.0-final.tar.gz

tar xvzf TDH-Basic-Component-Transwarp-5.2.0-final.tar.gz

mv registry transwarp/

cd transwarp

./install

## 用Transwarp Manager配置集群

打开客户端浏览器（推荐使用Google Chrome浏览器），输入安装好的管理节点IP或DNS地址，比如http://10.64.141.30:8180/，账户名密码均为admin。

点击同意协议，输入集群名称，然后下一步。

修改机柜，添加节点，配置RPM库。

## 用Transwarp Manager安装服务

## TDH Client安装

获取TDH-Client，在Manager界面上点击管理->下载客户端，下载TDH客户端。

放置到集群上并解压；

执行脚本source TDH-Client/init.sh；

注意：TDH-Client在集群添加修改服务后需要重新下载配置。

# TDH组件测试

## HDFS

## HIVE

参考3.5Inceptor

## Hyperbase

建立在Apache HBase和Elastic Search之上的NoSQL数据库。

Hyperbase高效地支持企业在线OLTP应用、高并发OLAP应用、批处理应用、全文搜索或高并发图形数据库检索应用。

HBase是一种Hadoop数据库，经常被描述为一种稀疏的，分布式的，持久化的，多维有序映射，它基于行键、列键和时间戳建立索引，是一个可以随机访问的存储和检索数据的平台。HBase不限制存储的数据的种类，允许动态的、灵活的数据模型，不用SQL语言，也不强调数据之间的关系。HBase被设计成在一个服务器集群上运行，可以相应地横向扩展。

### HBase shell入门

使用su - tdhadmin切换用户和shell环境；

$ hbase shell 启动shell；

hbase > list 列出所有的表；

hbase > create ‘mytable’,’hb’ 创建名为mytable表，含有一个列族hb；

hbase > put  'mytable' , 'first' , 'hb:data' , 'hello HBase'  在‘mytable’表的'first'行中的‘hb:data’列对应的数据单元中插入字节数组‘hello HBase’；

hbase > get 'mytable' , 'first'  读取mytable表的first行数据；

hbase > scan‘mytable'  读取mytable表的所有内容；

(详细可参考Hyberbase使用手册“14. HBase表SQL使用说明”)

create table hbase\_inner\_table(

key1 string,

bi bigint,

dc decimal(10,2),

ch varchar(10),

ts timestamp,

en string)STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler';

### Hyberdriver SQL

使用前需要现在Inceptor控制台中配置：

Inceptor1 ->更多操作 ->更新依赖 ->配置服务 ->重启Inceptor1。

create\_table 'htable', ':key:string, f:c1:int, f:c2:double, f:c3:varchar(10)’

### HBase数据模型

HBase以“表”为结构来组织数据，表中有行和列，但是HBase中的这些概念和关系型数据库的二维表不同。HBase表应当被看做一个稀疏的、分布式的、持久的、多维度有序map。

1. 数据对象：

命名空间：表的逻辑分组

表：HBase表、Hyperdrive表

行：每一行由一个独特的row key和多个列组成。

列族、列限定名、列

单元格：一组row key、列族和和列限定符指定唯一单元格。单元格 中存放一个值和timestamp。

时间戳：单元格的值有不同版本，由时间戳来区分。

索引：全局索引、全文索引

1. 表结构：

HBase更接近多维表，由多层键值对组成。

{Table: {RowKey: {ColumnFamily: {ColumnQualifier: {Timestamp: Value}}}}}

通过Inceptor使用SQL对HBase表进行操作时，Inceptor中的表将是HBase的二维映射。映射表的列单元格只有timestamp最新的值。

1. 全局索引、全文索引：

全局索引（Global Index）：全局索引中的数据做为另一张相对独立的HBase表存在，它的row key即为原表的索引字段，整个过程对用户完全透明的完成。

全文索引（Fulltext Index）：全文索引利用Elasticsearch作为索引数据的存储，用于加速对指定字段的 模糊查询。

### HBase架构

HBase是包含三类服务器的主从式架构。

RegionServer负责响应用户I/O请求，并向HDFS中读写数据。

Region的分配、DDL操作则有HMaster来处理。

所有的HBase数据都存储在HDFS中。

HBase表按照行键被水平地拆分成多个Region。

HBase使用Zookeeper来提供分布式协调服务，以维持集群中的服务器状态。

优势：

* 强一致模型：当写操作返回时，所有 读者都能看到相同的值。
* 自动化规模：当数据过大时，会自动分割Region；利用HDFS来传播和备份数据。
* 内置恢复：WAL的使用
* 分布式集群：基于Hadoop，使用MapReduce

不足：

* 复杂的故障修复
* 架构设计复杂，占用较多资源

### HBase表SQL

#### 创建内表

create table hbase\_inner\_table(

key1 string,

bi bigint,

dc decimal(10,2),

ch varchar(10),

ts timestamp,

en string)STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler';

hbase表支持的数据类型包括:BOOLEAN, TINYINT, SMALLINT, INT, BIGINT, DATE, TIMESTAMP, FLOAT, DOUBLE, STRING, VARCHAR, DECIMAL 和 STRUCT。

查看元数据信息：describe formatted hbase\_inner\_table;

#### 创建外表

hbase\_table 必须为HBase中一个已存在的表名。

CREATE EXTERNAL TABLE hbase\_external\_table(

key1 string,

ex1 double,

ex3 date,

ex5 string

)

STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'

WITH SERDEPROPERTIES ("hbase.columns.mapping"=":key,f:q1,f:q4,f:q5")

TBLPROPERTIES ("hbase.table.name"="assetmap.hbase\_inner\_table");

#### 插入数据

批量插入数据：

BATCHINSERT INTO hbase\_inner\_table BATCHVALUES (

VALUES('004',4,4.01,'esdrive','2017-01-11 15:05:20','wednesday'),

VALUES('005',5,5.01,'transwarp es','2017-01-12 15:18:18','thursday'),

VALUES('006',6,6.01,'hyperdrive','2017-01-13 05:13:13','friday'),

VALUES('007',7,7.01,'inceptor','2017-01-14 10:55:20','saturday'),

VALUES('008',8,8.01,'fulltext','2017-01-15 17:23:40','tuestuesday')

);

#### 查询数据

查询内表： select \* from hbase\_inner\_table;

查询外表： select \* from hbase\_external\_table;

虽然并没有直接向外表中插入数据，从hbase\_external\_table中也可以查询到相同数据，因两者映射HBase中的同一张表。

update hbase\_inner\_table set en=false where key1='004';

DELETE FROM hbase\_inner\_table WHERE key1 = '004';

插入、更新、删除操作都是同时对两张表生效。

Inceptor Engine的优化器有一套自动选择索引的机制，当 SELECT 查询语句未明确使用哪个全局索引时，优化器会为HBase表自动选择认为合适的全局索引。

#### 索引查询

1）为内表创建全局索引：

CREATE GLOBAL INDEX ch\_global ON hbase\_inner\_table(ch(10));

根据列ch创建一个名为ch\_global的全局索引，并指定该索引字段的长度为10.

1. 为内表创建全文索引;

CREATE FULLTEXT INDEX ON hbase\_inner\_table(bi,ch,en) SHARD NUM 1;

为内表hbase\_inner\_table根据列en、cn来创建全文索引。

### HBase Java API

下载依赖包jars

下载hbase-site.xml并配置到resource路径下

参考incepter工程代码-HBaseAPIDemo

## Inceptor

Transwarp Inceptor是星环科技推出的用于数据仓库和交互式分析的大数据平台软件，它基于Hadoop和Spark技术平台打造，可提供完整的SQL支持，支持主流的SQL模块化扩展，兼容通用开发框架和工具，支持事务特性保证数据的准确性，且能够利用内存或者SSD来加速数据的读取，支持与关系型数据库实时对接并做统计分析，辅以高性能的SQL执行引擎，为企业提供高性价比和高度可扩展的解决方案。

### Inceptor连接信息

Inceptor连接服务在10.64.141.30服务器，管理员用户密码为hive/hive。

### beeline命令行连接Inceptor

切换到tdhadmin用户：su - tdhadmin

beeline -u "jdbc:hive2://10.64.141.30:10000/test01" -n hive -p hive

展示所有数据库：show databses；

展示所有表：show tables；

### 使用DbVisylizer可视化工具连接

首先在Inceptor运维手册中下载依赖包；

下载DbVisulizer客户端并安装；

Tools-Driver Manager配置驱动；

Connections新建连接；

### Inceptor命令行

1. 使用tdhadmin用户登录，新建inceptor-quickstart.txt文件，并插入两行数据。
2. 新建外表quickstart

(此处的local path需要为pod中的地址，详情参考2.7章)

DROP TABLE IF EXISTS quickstart;

CREATE EXTERNAL TABLE quickstart (a INT, b STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|';

CREATE EXTERNAL TABLE quickstart (a INT, b STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' LOCATION '/test/';

(需要使用hdfs用户上传quickstart数据到hdfs的/test/目录下)

LOAD DATA LOCAL INPATH '/test/quickstart.txt' OVERWRITE INTO TABLE quickstart;

SELECT \* FROM quickstart;

1. Inceptor JDBC操作

（参考8. Inceptor JDBC手册预览）

从Transwarp Manager上直接下载Inceptor JDBC驱动。

org.apache.hive.jdbc.HiveDriver为InceptorServer的JDBC驱动，在和InceptorServer交互时使用。

### Inceptor Java API

下载依赖包jars

参考incepter工程代码-InceptorJDBCExample

## Kafka

（详情可参考Slipstream使用手册附录部分）

登录主节点进入kafka目录：/home/tdhadmin/TDH-Client/kafka/bin

### 创建topic

用于创建topic的脚本为kafka-topics.sh；

./kafka-topics.sh --topic kafka-topic --zookeeper bak-BigDataNode4:2181 --create --partitions 1 --replication-factor 1

### Console-Producer发送消息

./kafka-console-producer.sh --broker-list bak-BigDataNode4:9092 --topic kafka-topic

### Console-Consumer消费消息

./kafka-console-consumer.sh --topic kafka-topic --bootstrap-server bak-BigDataNode4:9092

## Slipstream

Transwarp Slipstream允许用户通过SQL的方式实现业务逻辑，大大降低了流应用开发的门槛。Slipstream几乎可以应用于所有业务场景，包括ETL工具，规则报警工具等简单业务场景。对于更复杂的业务逻辑，用户可以选择使用PLSQL，属于Slipstream的高级功能。Slipstream新增了基于事件驱动模式的流处理功能，在低延迟处理方面的性能表现更为出色。

采用Slipstream具有以下优势：

* 微批模式和事件驱动模式的一体化
* 极高的易用性
* 性能提升
* 产品化程度高
* 迁移成本低

### 快速入门

1. 连接到StreamJob：

beeline -u "jdbc:hive2://bak-BigDataNode01:10010" -n hive -p hive

2)创建Stream：CREATE STREAM demo\_stream(id INT, letter STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' TBLPROPERTIES("topic"="kafka-topic","kafka.zookeeper"="10.40.141.33:2181"，"kafka.broker.list"="10.40.141.33:9092");

3)查看Stream：SHOW STREAMS;

4)创建StreamJob：

CREATE TABLE demo\_table(id INT, letter STRING);

1. 事件触发Job：

INSERT INTO demo\_table SELECT \* FROM demo\_stream;

1. 接收并处理kafka数据

kafka producer向topic中发送数据；

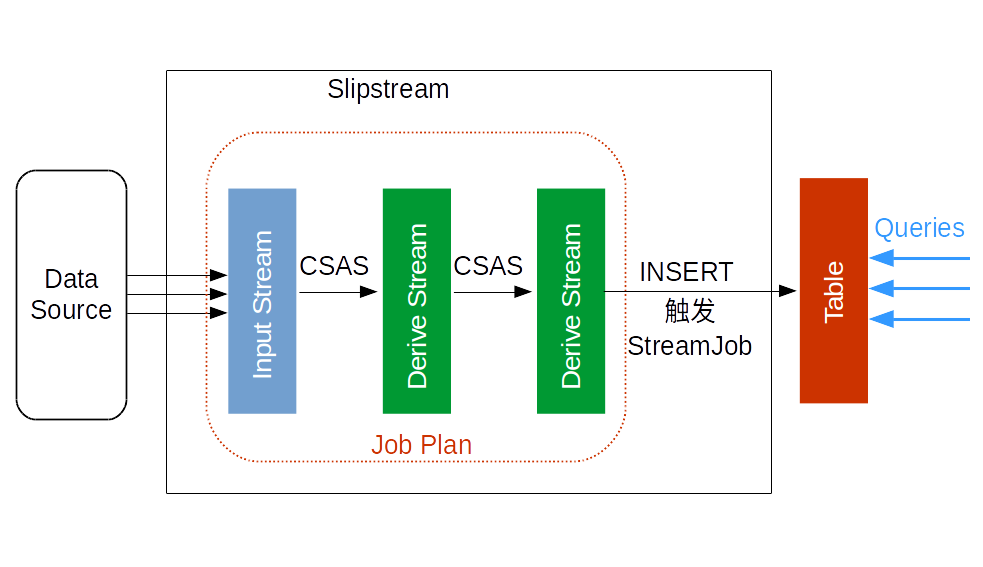
查询数据SELECT \* FROM demo\_table;

1. 停止Streamjob

stop streamjob 29008ed34b9e45bca784362948b88a85;

### Slipstream基础

Slipstream支持从不同的源中获取实时的流数据, 计算处理之后输出到不同的文件系统。三个核心概念：Stream、StreamJob、Application。Stream是数据流，StreamJob是对一个或多个Stream进行计算并将结果写进一张表的任务，Application是一个或多个StreamJob的集合。



### Slipstream的两种模型

微批模型和事件驱动模型。

* 事件驱动模型更接近原生的流处理，一有数据进来就处理，好比流水线上的零件，来一个处理一个。
* 微批模型：基于批的思想，将Input Stream按时间段分割成一个个小块处理，就像是给流水线增加一个缓冲，每隔一段时间去取一批零件加工。
* 事件模型相比微批模型具有更低的延迟，微秒级；复杂事件的处理；更加完善的高可用。

### Slipstream交互

目前Slipstream提供三种方式的交互：

Shell、JDBC、ODBC

## ElasticSearch

# Hive与HBase测试

## Hive建表导入数据

1. 创建融慧探二期新房成交的表，读取指定目录下所有数据文件。

CREATE EXTERNAL TABLE tran\_orig(ID STRING, PROJECT\_ID STRING, CITY STRING, DISTRICT STRING, PROJECT\_NAME STRING, USEAGE STRING, TRAN\_AREA STRING, UNIT\_PRICE STRING, TOTAL\_PRICE STRING, TRAN\_DATE STRING, ADDRESS STRING, STATUS STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION '/test/assetmap/';

1. 导出新房成交数据。

使用PLSQL的Export Query Results导出数据文件。

使用hdfs登录服务器，将数据文件.csv放至HDFS的/test/assetmap/目录下。（要注意上传文件的权限，所有者及所属组）

创建文件夹：hadoop fs -mkdir /test/assetmap

上传文件：hadoop fs -put /home/hdfs/\*.csv /test/assetmap

1. hive中创建数据库及表。

创建数据库：create database assetmap;

use assetmap；

select \* from tran\_orig limit 10；

## HBase建表导入数据

使用hbase提供好的importTSV工具批量导入数据。

1. 首先将需要导入的数据上传到hdfs集群。
2. 在hbase中创建表：create 'tran\_orig','cf'
3. 使用importTSV工具导入

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator="," -Dimporttsv.columns=HBASE\_ROW\_KEY,cf:project\_id,cf:city,cf:district,cf:project\_name,cf:useage,cf:tran\_area,cf:unit\_price,cf:total\_price,cf:tran\_date,cf:address,cf:status tran\_orig /home/hdfs/beihai.csv

（注意的是数据第一列为rowkey）

1. 查询数据

get ‘tran\_orig’, ‘”36847850”’

## hive与hbase关联表

Hive是高延迟、结构化、面向分析的；Hbase是低延迟、非结构化、面向编程的。Hive集成Hbase可以有效利用HBase数据库的存储特性，如行更新和列索引等。Hive集成HBase需要在Hive表和HBase之间建立映射关系。每一个在Hive表的域都存在于HBase中，而在Hive表中的不需要包含所有HBase的列。

1. 建立Hive内表，指向hbase（数据存在于hbase中）

通过下面的方式创建hive的内表，这种情况是hbase本来没有这张表，创建后会在hbase中同样创建一张表，将来数据也是存放在hbase中的；hdfs的hive表目录有hive文件夹，但是里面没有数据。

1）创建hive-hbase表

create table hive\_hbase\_test(id string,name string, age int) stored by'org.apache.hadoop.hive.hbase.HBaseStorageHandler' with serdeproperties("hbase.columns.mapping" = ":key,cf:name,cf:age")tblproperties ("hbase.table.name" = "hive\_hbase\_test");

1. 创建普通hive内表，加载数据

hive> createtable test\_liu (id string,name string, age int) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;

load datalocal inpath '/home/qytt/test' into table test\_liu;

1. 导入数据到hive-hbase表中

insert overwrite table hive\_hbase\_test select \* from test\_liu;

1. 在hbase中查询数据

get 'hive\_hbase\_test',1

当删除hive表时，hbase表也会删除。

1. 建立Hive外表，引用HBase中已经存在的表

删除Hive表对HBase没有影响；

但是先删除hbase表hive就会报错找不到表。

# TDH运维

# 附录

## TDH5.2组件对应的开源组件版本

| 组件 | 版本 |
| --- | --- |
| Kubernetes | 1.5 |
| Docker | 1.13.0 |
| Elastic Search | 5.4.1 |
| HBase | 0.98.6+1.3.1 |
| HDFS | 2.7.3 |
| Kafka | 0.10.2 |
| Spark | 2.2 |
| Tensorflow | 1.1 |
| YARN | 2.7.3 |
| Zepplin | 0.7.0 |
| Zookeeper | 3.4.5 |