

# 尚硅谷大数据技术之 Impala

(作者：尚硅谷大数据研发部)

版本：V1.0

## 第 1 章 Impala 的基本概念

### 1.1 什么是 Impala

Cloudera 公司推出，提供对 HDFS、Hbase 数据的高性能、低延迟的交互式 SQL 查询功能。

基于 Hive，使用内存计算，兼顾数据仓库、具有实时、批处理、多并发等优点。

是 CDH 平台首选的 PB 级大数据实时查询分析引擎。

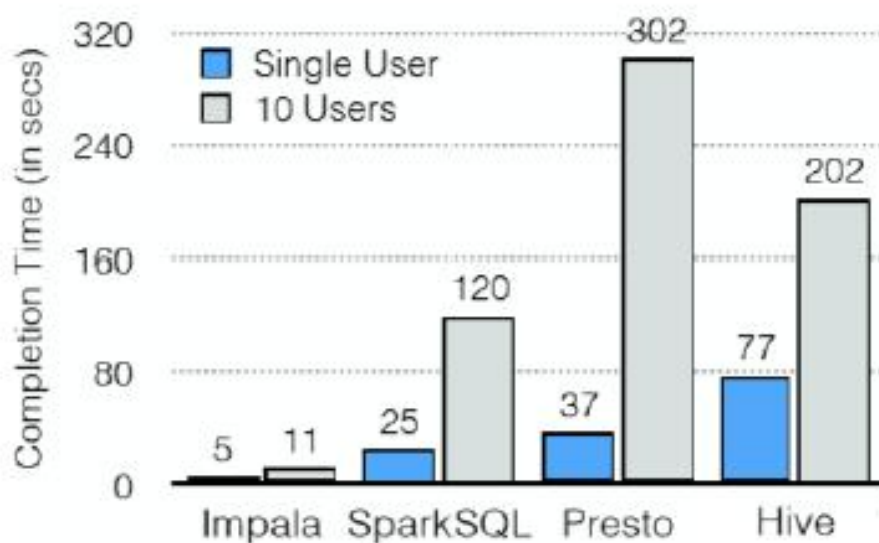


Apache Impala is the open source, native analytic database for Apache Hadoop. Impala is shipped by Cloudera, MapR, Oracle, and Amazon.

### 1.2 Impala 的优缺点

#### 1.2.1 优点

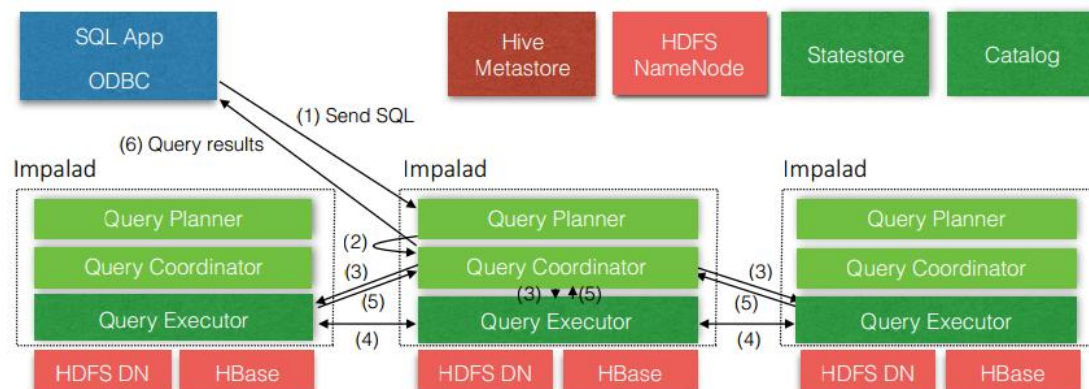
- 1) 基于内存运算，不需要把中间结果写入磁盘，省掉了大量的 I/O 开销。
- 2) 无需转换为 Mapreduce，直接访问存储在 HDFS，HBase 中的数据进行作业调度，速度快。
- 3) 使用了支持 Data locality 的 I/O 调度机制，尽可能地将数据和计算分配在同一台机器上进行，减少了网络开销。
- 4) 支持各种文件格式，如 TEXTFILE、SEQUENCEFILE、RCFile、Parquet。
- 5) 可以访问 hive 的 metastore，对 hive 数据直接做数据分析。



### 1.2.2 缺点

- 1) 对内存的依赖大，且完全依赖于 hive。
- 2) 实践中，分区超过 1 万，性能严重下降。
- 3) 只能读取文本文件，而不能直接读取自定义二进制文件。
- 4) 每当新的记录/文件被添加到 HDFS 中的数据目录时，该表需要被刷新。

### 1.3 Impala 的架构



从上图可以看出，Impala 自身包含三个模块：Impalad、Statestore 和 Catalog，除此之外它还依赖 Hive Metastore 和 HDFS。

- 1) Impalad:
  - 接收 client 的请求、Query 执行并返回给中心协调节点；
  - 子节点上的守护进程，负责向 statestore 保持通信，汇报工作。
- 2) Catalog:

分发表的元数据信息到各个 impalad 中；

接收来自 statestore 的所有请求。

### 3) Statestore:

负责收集分布在集群中各个 impalad 进程的资源信息、各节点健康状况，同步节点信息；

负责 query 的协调调度。

## 第 2 章 Impala 的安装

### 2.1 Impala 的地址

#### 1. Impala 的官网

<http://impala.apache.org/>

#### 2. Impala 文档查看

<http://impala.apache.org/impala-docs.html>

#### 3. 下载地址

<http://impala.apache.org/downloads.html>

### 2.2 Impala 的安装方式

- Cloudera Manager (CDH 首推)
- 手动安装

下面我们使用 Cloudera Manager 安装 Impala


#### 1. 在主页中点击添加服务



## 2.选择 Impala 服务

将服务添加到 Cluster 1

选择您要添加的服务类型。

服务类型	说明
 Accumulo	The Apache Accumulo sorted, distributed key/value store is a robust, scalable, high performance data storage and retrieval system. This service only works with releases based on Apache Accumulo 1.6 or later.
 Flume	Flume 从几乎所有来源收集数据并将这些数据聚合到永久性存储（如 HDFS）中。
 HBase	Apache HBase 提供对大型数据集的随机、实时的读/写访问权限（需要 HDFS 和 ZooKeeper）。
 HDFS	Apache Hadoop 分布式文件系统 (HDFS) 是 Hadoop 应用程序使用的主要存储系统。HDFS 创建多个数据块副本并将它们分布在整个群集的计算机上，以启用可靠且极其快速的计算功能。
 Hive	Hive 是一种数据仓库系统，提供名为 HiveQL 的 SQL 类语言。
 Hue	Hue 是与包括 Apache Hadoop 的 Cloudera Distribution 配合使用的图形用户界面(需要 HDFS、MapReduce 和 Hive)。
 Impala	Impala 为存储在 HDFS 和 HBase 中的数据提供了一个实时 SQL 查询接口。Impala 需要 Hive 服务，并与 Hue 共享 Hive Metastore。
 Isilon	EMC Isilon is a distributed filesystem.
 Java KeyStore KMS	The Hadoop Key Management Service with file-based Java KeyStore. Maintains a single copy of keys, using simple password-based protection. Requires CDH 5.3+. <b>Not recommended for production use.</b>
 Kafka	Apache Kafka is publish-subscribe messaging rethought as a distributed commit log. <b>Before adding this service, ensure that either the Kafka parcel is activated or the Kafka package is installed.</b>
 Key-Value Store Indexer	键/值 Store Indexer 侦听 HBase 中所含表内的数据变化，并使用 Solr 为其创建索引。
 Kudu	Kudu is a true column store for the Hadoop ecosystem.
 MapReduce	Apache Hadoop MapReduce 支持对整个群集中的大型数据集进行分布式计算（需要 HDFS）。 <b>建议改用 YARN（包括 MapReduce 2）。包括 MapReduce 用于向后兼容性。</b>

返回

继续

#### 4. 进行角色分配

## 将 Impala 服务添加到 Cluster 1

### 自定义 Impala 的角色分配

您可以在此处自定义新服务的角色分配，但请注意，如果分配不正确（例如，分配到某个主机上的角色太多），性能受到影响。

还可以按主机查看角色分配。

[按主机查看](#)

ISS Impala StateStore × 1 新建

ICS Impala Catalog Server × 1 新建

ID Impala Daemon × 3 新建

hadoop102

hadoop102

hadoop[102-104] ▼

注意：最好将 StateStore 和 CataLog Sever 单独部署在同一节点上。

## 5.配置 Impala

### 将 Impala 服务添加到 Cluster 1

审核更改

Kudu 服务

Impala (服务范围)

none

Impala Daemon 暂存目录

Impala Daemon Default Group ...和另 1 个

scratch\_dir

/impala/impalad

编辑单个值

## 6.启动 Impala

### 将 Impala 服务添加到 Cluster 1

首次运行 命令

状态 已完成 10月 20, 4:28:01 下午 26.86s

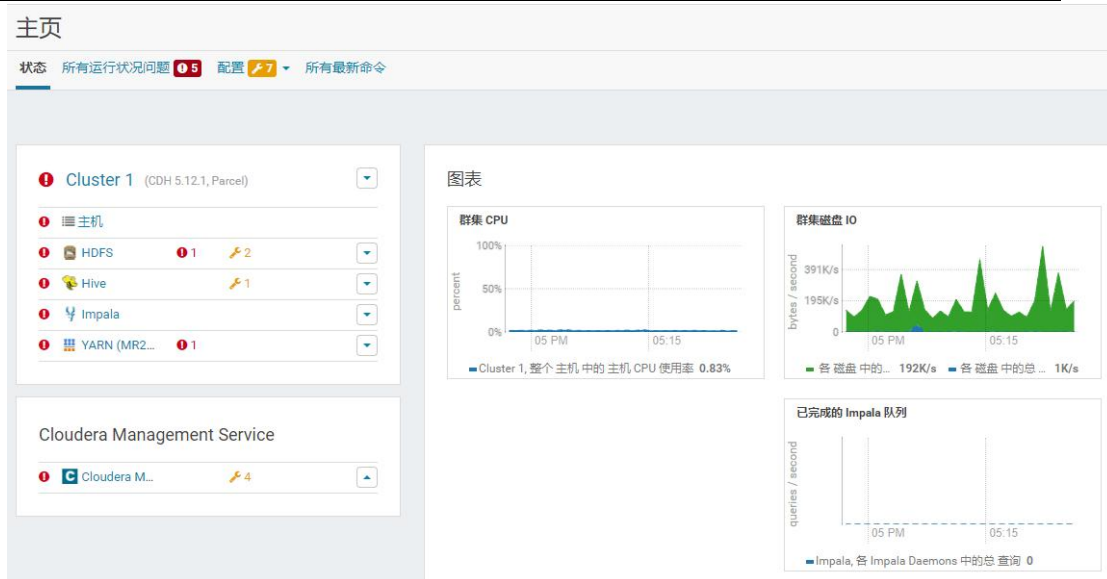
Finished First Run of the following services successfully: Impala.

✓ 已完成 3 个步骤 (共 3 个)

☒ Show All Steps
 ☐ Show Only Failed Steps
 ☐ Show Only Running Steps

<div>Ensuring that the expected software releases are installed on hosts.</div> <div>已成功完成 1 个步骤。</div>	10月 20, 4:28:01 下午	27ms
<div>正在创建 Impala 用户目录</div> <div>Successfully created HDFS directory.</div>	10月 20, 4:28:01 下午	4.17s
<div>启动 Impala</div> <div>Successfully started service.</div>	10月 20, 4:28:05 下午	22.57s

## 7.安装成功



## 2.3 Impala 的监护管理

可以通过下面的链接来访问 Impala 的监护管理页面：

- 查看 StateStore

<http://hadoop102:25020/>

- 查看 Catalog

<http://hadoop102:25010/>

## 2.4 Impala 的初体验

### 1.启动 Impala

```
[root@hadoop102 ~]# impala-shell
```

### 2.查看数据库

```
[hadoop102:21000] > show databases;
```

### 3.打开默认数据库

```
[hadoop102:21000] > use default;
```

### 4.显示数据库中的表

```
[hadoop102:21000] > show tables;
```

### 5.创建一张 student 表

```
[hadoop102:21000] > create table student(id int, name string)
> row format delimited
> fields terminated by '\t';
```

### 6.向表中导入数据

```
[hadoop103:21000] > load data inpath '/student.txt' into table student;
```

注意：

- 1) 关闭（修改 hdfs 的配置 dfs.permissions 为 false）或修改 hdfs 的权限，否则 impala 没有写的权限

```
[hdfs@hadoop103 ~]$ hadoop fs -chmod -R 777 /
```

2) Impala 不支持将本地文件导入到表中

#### 7.查询

```
[hadoop103:21000] > select * from student;
```

#### 8.退出 impala

```
[hadoop103:21000] > quit;
```

## 第 3 章 Impala 的操作命令

### 3.1 Impala 的外部 shell

选项	描述
-h, --help	显示帮助信息
-v or --version	显示版本信息
-i hostname, --impalad=hostname	指定连接运行 impalad 守护进程的主机。默认端口是 21000。
-q query, --query=query	从命令行中传递一个 shell 命令。执行完这一语句后 shell 会立即退出。
-f query_file, --query_file=query_file	传递一个文件中的 SQL 查询。文件内容必须以分号分隔
-o filename or --output_file filename	保存所有查询结果到指定的文件。通常用于保存在命令行使用 -q 选项执行单个查询时的查询结果。
-c	查询执行失败时继续执行
-d default_db or --database=default_db	指定启动后使用的数据库,与建立连接后使用 use 语句选择数据库作用相同,如果没有指定,那么使用 default 数据库
-r or --refresh_after_connect	建立连接后刷新 Impala 元数据
-p, --show_profiles	对 shell 中执行的每一个查询,显示其查询执行计划
-B (--delimited)	去格式化输出
--output_delimiter=character	指定分隔符
--print_header	打印列名

#### 1. 连接指定 hadoop103 的 impala 主机

```
[root@hadoop102 datas]# impala-shell -i hadoop103
```

#### 2. 使用-q 查询表中数据,并将数据写入文件中

```
[hdfs@hadoop103 ~]$ impala-shell -q 'select * from student' -o output.txt
```

#### 3. 查询执行失败时继续执行

```
[hdfs@hadoop103 ~]$ vim impala.sql
select * from student;
select * from stu;
select * from student;
[hdfs@hadoop103 ~]$ impala-shell -f impala.sql;
[hdfs@hadoop103 ~]$ impala-shell -c -f impala.sql;
```

#### 4. 在 hive 中创建表后,使用-r 刷新元数据



```
hive> create table stu(id int, name string);
[hadoop103:21000] > show tables;
Query: show tables
+-----+
| name   |
+-----+
| student |
+-----+

[hdfs@hadoop103 ~]$ impala-shell -r
[hadoop103:21000] > show tables;
Query: show tables
+-----+
| name   |
+-----+
| stu    |
| student |
+-----+
```

#### 5. 显示查询执行计划

```
[hdfs@hadoop103 ~]$ impala-shell -p
[hadoop103:21000] > select * from student;
```

#### 6. 去格式化输出

```
[root@hadoop103 ~]# impala-shell -q 'select * from student' -B
--output_delimiter="\t" -o output.txt
[root@hadoop103 ~]# cat output.txt
1001    tignitgn
1002    yuanyuan
1003    haohao
1004    yunyun
```

### 3.2 Impala 的内部 shell

选项	描述
help	显示帮助信息
explain <sql>	显示执行计划
profile	(查询完成后执行) 查询最近一次查询的底层信息
shell <shell>	不退出 impala-shell 执行 shell 命令
version	显示版本信息 (同于 impala-shell -v)
connect	连接 impalad 主机, 默认端口 21000 (同于 impala-shell -i)
refresh <tablename>	增量刷新元数据库
invalidate metadata	全量刷新元数据库 (慎用) (同于 impala-shell -r)
history	历史命令

#### 1. 查看执行计划

```
explain select * from student;
```

#### 2. 查询最近一次查询的底层信息

```
[hadoop103:21000] > select count(*) from student;
[hadoop103:21000] > profile;
```

#### 3. 查看 hdfs 及 linux 文件系统

```
[hadoop103:21000] > shell hadoop fs -ls /;
```



```
[hadoop103:21000] > shell ls -al ./;
```

#### 4. 刷新指定表的元数据

```
hive> load data local inpath '/opt/module/datas/student.txt' into table student;
[hadoop103:21000] > select * from student;
[hadoop103:21000] > refresh student;
[hadoop103:21000] > select * from student;
```

#### 5. 查看历史命令

```
[hadoop103:21000] > history;
```

## 第 4 章 Impala 的数据类型

Hive 数据类型	Impala 数据类型	长度
TINYINT	TINYINT	1byte 有符号整数
SMALLINT	SMALLINT	2byte 有符号整数
INT	INT	4byte 有符号整数
BIGINT	BIGINT	8byte 有符号整数
BOOLEAN	BOOLEAN	布尔类型, true 或者 false
FLOAT	FLOAT	单精度浮点数
DOUBLE	DOUBLE	双精度浮点数
STRING	STRING	字符系列。可以指定字符集。可以使用单引号或者双引号。
TIMESTAMP	TIMESTAMP	时间类型
BINARY	不支持	字节数组

注意: Impala 虽然支持 array, map, struct 复杂数据类型, 但是支持并不完全, 一般处理方法, 将复杂类型转化为基本类型, 通过 hive 创建表。

## 第 5 章 DDL 数据定义

### 5.1 创建数据库

```
CREATE DATABASE [IF NOT EXISTS] database_name
[COMMENT database_comment]
[LOCATION hdfs_path];
```

注: Impala 不支持 WITH DBPROPERTIES...语法

```
[hadoop103:21000] > create database db_hive
> WITH DBPROPERTIES('name' = 'ttd');
Query: create database db_hive
WITH DBPROPERTIES('name' = 'ttd')
ERROR: AnalysisException: Syntax error in line 2:
WITH DBPROPERTIES('name' = 'ttd')
^
Encountered: WITH
Expected: COMMENT, LOCATION
```

### 5.2 查询数据库

#### 5.2.1 显示数据库

```
[hadoop103:21000] > show databases;
[hadoop103:21000] > show databases like 'hive*';
```

```
Query: show databases like 'hive*'
+-----+-----+
| name   | comment |
+-----+-----+
| hive_db |         |
+-----+-----+
[hadoop103:21000] > desc database hive_db;
Query: describe database hive_db
+-----+-----+-----+
| name   | location | comment |
+-----+-----+-----+
| hive_db |         |         |
+-----+-----+-----+
```

## 5.2.2 删除数据库

```
[hadoop103:21000] > drop database hive_db;
[hadoop103:21000] > drop database hive_db cascade;
```

注:

Impala 不支持 alter database 语法

当数据库被 USE 语句选中时, 无法删除

## 5.3 创建表

### 5.3.1 管理表

```
[hadoop103:21000] > create table if not exists student2(
    > id int, name string
    > )
    > row format delimited fields terminated by '\t'
    > stored as textfile
    > location '/user/hive/warehouse/student2';
[hadoop103:21000] > desc formatted student2;
```

### 5.3.2 外部表

```
[hadoop103:21000] > create external table stu_external(
    > id int,
    > name string)
    > row format delimited fields terminated by '\t' ;
```

## 5.4 分区表

### 5.4.1 创建分区表

```
[hadoop103:21000] > create table stu_par(id int, name string)
    > partitioned by (month string)
    > row format delimited
    > fields terminated by '\t';
```

### 5.4.2 向表中导入数据

```
[hadoop103:21000] > alter table stu_par add partition (month='201810');
[hadoop103:21000] > load data inpath '/student.txt' into table stu_par
partition(month='201810');
[hadoop103:21000] > insert into table stu_par partition (month = '201811')
    > select * from student;
```

注意:

如果分区没有, load data 导入数据时, 不能自动创建分区。

### 5.4.3 查询分区表中的数据

```
[hadoop103:21000] > select * from stu_par where month = '201811';
```

### 5.4.4 增加多个分区

```
[hadoop103:21000] > alter table stu_par add partition (month='201812') partition  
(month='201813');
```

### 5.4.5 删除分区

```
[hadoop103:21000] > alter table stu_par drop partition (month='201812');
```

### 5.4.5 查看分区

```
[hadoop103:21000] > show partitions stu_par;
```

## 第 6 章 DML 数据操作

### 6.1 数据导入 (基本同 hive 类似)

注意: impala 不支持 load data local inpath...

### 6.2 数据的导出

1. impala 不支持 insert overwrite... 语法导出数据

2. impala 数据导出一般使用 impala -o

```
[root@hadoop103 ~]# impala-shell -q 'select * from student' -B  
--output_delimiter="\t" -o output.txt  
[root@hadoop103 ~]# cat output.txt  
1001 tignitgn  
1002 yuanyuan  
1003 haohao  
1004 yunyun
```

Impala 不支持 export 和 import 命令

## 第 7 章 查询

1. 基本的语法跟 hive 的查询语句大体一样
2. Impala 不支持 CLUSTER BY, DISTRIBUTE BY, SORT BY
3. Impala 中不支持分桶表
4. Impala 不支持 COLLECT\_SET(col) 和 explode (col) 函数
5. Impala 支持开窗函数

```
[hadoop103:21000] > select name, orderdate, cost, sum(cost) over(partition by  
month(orderdate)) from business;
```

## 第 8 章 函数

### 8.1 自定义函数

1. 创建一个 Maven 工程 Hive

2. 导入依赖

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.apache.hive/hive-exec -->
  <dependency>
    <groupId>org.apache.hive</groupId>
    <artifactId>hive-exec</artifactId>
    <version>1.2.1</version>
  </dependency>
</dependencies>
```

3. 创建一个类

```
package com.atguigu.hive;
import org.apache.hadoop.hive.ql.exec.UDF;

public class Lower extends UDF {

    public String evaluate (final String s) {

        if (s == null) {
            return null;
        }

        return s.toLowerCase();
    }
}
```

4. 打成 jar 包上传到服务器/opt/module/jars/ hive\_udf-0.0.1-SNAPSHOT.jar

5. 将 jar 包上传到 hdfs 的指定目录

```
hadoop fs -put hive_udf-0.0.1-SNAPSHOT.jar /
```

6. 创建函数

```
[hadoop103:21000] > create function mylower(string) returns string location
'/hive_udf-0.0.1-SNAPSHOT.jar' symbol='com.atguigu.hive_udf.Hive_UDF';
```

7. 使用自定义函数

```
[hadoop103:21000] > select ename, mylower(ename) from emp;
```

8. 通过 show functions 查看自定义的函数

```
[hadoop103:21000] > show functions;
Query: show functions
+-----+-----+-----+-----+
| return type | signature          | binary type | is persistent |
+-----+-----+-----+-----+
| STRING      | mylower(STRING)    | JAVA        | false         |
+-----+-----+-----+-----+
```

## 第 9 章 存储和压缩

文件格式	压缩编码	Impala 是否可直接创建	是否可直接插入
Parquet	Snappy (默认), GZIP;	Yes	支持: CREATE TABLE, INSERT, 查询
TextFile	LZO, gzip, bzip2, snappy	Yes. 不指定 STORED AS 子句的 CREATE TABLE 语句, 默认的文件格式就是未压缩文本	支持: CREATE TABLE, INSERT, 查询。如果使用 LZO 压缩, 则必须在 Hive 中创建表和加载数据
RCFile	Snappy, GZIP, deflate, BZIP2	Yes.	支持 CREATE, 查询, 在 Hive 中加载数据
SequenceFile	Snappy, GZIP, deflate, BZIP2	Yes.	支持: CREATE TABLE, INSERT, 查询。需设置

注: impala 不支持 ORC 格式

### 1. 创建 parquet 格式的表并插入数据进行查询

```
[hadoop104:21000] > create table student2(id int, name string)
> row format delimited
> fields terminated by '\t'
> stored as PARQUET;
[hadoop104:21000] > insert into table student2 values(1001,'zhangsan');
[hadoop104:21000] > select * from student2;
```

### 2. 创建 sequenceFile 格式的表, 插入数据时报错

```
[hadoop104:21000] > create table student3(id int, name string)
> row format delimited
> fields terminated by '\t'
> stored as sequenceFile;
[hadoop104:21000] > insert into table student3 values(1001,'zhangsan');
Query: insert into table student3 values(1001,'zhangsan')
Query submitted at: 2018-10-25 20:59:31 (Coordinator: http://hadoop104:25000)
Query progress can be monitored at:
http://hadoop104:25000/query_plan?query_id=da4c59eb23481bdc:26f012ca00000000
WARNINGS: Writing to table format SEQUENCE_FILE is not supported. Use query option
ALLOW_UNUNSUPPORTED_FORMATS to override.
[hadoop104:21000] > set ALLOW_UNUNSUPPORTED_FORMATS=true;
[hadoop104:21000] > insert into table student3 values(1001,'zhangsan');
```

## 第 10 章 优化

- 1、尽量将 StateStore 和 Catalog 单独部署到同一个节点, 保证他们正常通信。
- 2、通过对 Impala Daemon 内存限制(默认 256M)及 StateStore 工作线程数, 来提高 Impala 的执行效率。

- 3、SQL 优化，使用之前调用执行计划
- 4、选择合适的文件格式进行存储，提高查询效率。
- 5、避免产生很多小文件（如果有其他程序产生的小文件，可以使用中间表，将小文件数据存放到中间表。然后通过 insert...select...方式中间表的数据插入到最终表中）
- 6、使用合适的分区技术，根据分区粒度测算
- 7、使用 compute stats 进行表信息搜集，当一个内容表或分区明显变化，重新计算统计相关数据表或分区。因为行和不同值的数量差异可能导致 impala 选择不同的连接顺序时进行查询。

```
[hadoop104:21000] > show table stats student;
Query: show table stats student
+-----+-----+-----+-----+-----+-----+-----+
| #Rows | #Files | Size | Bytes Cached | Cache Replication | Format | Incremental |
stats | Location |
+-----+-----+-----+-----+-----+-----+-----+
| -1    | 1      | 67B  | NOT CACHED   | NOT CACHED        | TEXT   | false       |
| hdf://hadoop102:8020/user/hive/warehouse/student |
+-----+-----+-----+-----+-----+-----+-----+

[hadoop104:21000] > compute stats student;
Query: compute stats student
+-----+
| summary |
+-----+
| Updated 1 partition(s) and 2 column(s). |
+-----+

[hadoop104:21000] > show table stats student;
Query: show table stats student
+-----+-----+-----+-----+-----+-----+-----+
| #Rows | #Files | Size | Bytes Cached | Cache Replication | Format | Incremental |
stats | Location |
+-----+-----+-----+-----+-----+-----+-----+
| 6     | 1      | 67B  | NOT CACHED   | NOT CACHED        | TEXT   | false       |
| hdf://hadoop102:8020/user/hive/warehouse/student |
+-----+-----+-----+-----+-----+-----+-----+
```

- 8、网络 io 的优化：
  - a.避免把整个数据发送到客户端
  - b.尽可能的做条件过滤
  - c.使用 limit 字句
  - d.输出文件时，避免使用美化输出
  - e.尽量少用全量元数据的刷新
- 9、使用 profile 输出底层信息计划，在做相应环境优化