

# 关卡 4

## 基于 Hive 的数据处理



HUAWEI

华为技术有限公司



# 目录

---

<b>1 实验环境介绍 .....</b>	<b>2</b>
1.1 实验介绍 .....	2
1.1.1 关于本实验 .....	2
1.1.2 实验目的 .....	2
<b>2 基于 Hive 的数据处理 .....</b>	<b>3</b>
2.1 实验任务 .....	3
2.1.1 上传模拟数据 .....	3
2.1.2 连接 Hive 环境 .....	4
2.1.3 问题一 .....	4
2.1.4 问题二 .....	5
2.1.5 问题三 .....	6
2.1.6 问题四 .....	7
2.1.7 部署 Kunpeng Hyper Tuner .....	8
2.1.8 使用 Kunpeng Hyper Tuner 优化 .....	9
2.2 思考题 .....	16

# 1

## 实验环境介绍

---

### 1.1 实验介绍

#### 1.1.1 关于本实验

本实验主要介绍在 Hadoop 的帮助下存储大量数据时，使用 Hive 的 HQL 语句简化 MapReduce 任务的实现，根据要求完成数据分析的任务。同时部署 Kunpeng Hyper Tuner 定位相关性能问题并使用优化手段缩短任务运行的时间。

#### 1.1.2 实验目的

帮助学员了解基础大数据环境，完成简单数据分析任务，并部署 Kunpeng Hyper Tuner 分析性能问题，使用优化手段缩短任务运行时间。

# 2 基于 Hive 的数据处理

## 2.1 实验任务

### 2.1.1 上传模拟数据

#### 步骤 1 模拟数据地址

```
/src/main/resources/others/
```

数据文件为在后端代码工程文件 others 目录下的 conversation.txt 和 message.txt。

```
cd ~  
mkdir testData  
cd /home/IMSystem/src/main/resources/others  
cp conversation.txt /root/testData/conversation.txt  
cp messages.txt /root/testData/messages.txt  
cd /root/testData  
ll
```

```
[root@server1 testData]# ll  
total 8.6M  
-rw----- 1 root root 445K Jan 20 17:49 conversation.txt  
-rw----- 1 root root 8.1M Jan 20 17:49 messages.txt
```

#### 步骤 2 将模拟数据上传 hdfs

```
hdfs dfs -ls /  
hdfs dfs -mkdir /user/Hadoop  
hdfs dfs -put /root/testData /user/Hadoop  
hdfs dfs -ls /user/Hadoop/testData
```

```
[root@server1 testData]# hdfs dfs -ls /user/Hadoop/testData  
Found 2 items  
-rw-r--r-- 1 root supergroup 455450 2022-01-22 14:57 /user/Hadoop/testData/conversation.txt  
-rw-r--r-- 1 root supergroup 8456388 2022-01-22 14:57 /user/Hadoop/testData/messages.txt
```

#### 步骤 3 修改文件权限

```
hdfs dfs -chmod -R 777 /user  
hdfs dfs -chmod -R 777 /tmp
```

## 2.1.2 连接 Hive 环境

### 步骤 1 运行 hive 及 hadoop

如果 hive 和 hadoop 关闭, 可根据关卡一的操作启动 hive 和 hadoop。

### 步骤 2 使用 beeline 连接 Hive

```
beeline -u jdbc:hive2://server1:10000
```

```
[root@server1 testData]# beeline -u jdbc:hive2://server1:10000
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop-3.3.1/share/hadoop/common/lib/slf4j-log4j12-1.7.30.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://server1:10000
Connected to: Apache Hive (version 3.1.2)
Driver: Hive JDBC (version 3.1.2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.2 by Apache Hive
0: jdbc:hive2://server1:10000>
```

## 2.1.3 问题一

假设 00:00:00-00:59:59 为一个小时, 请问每个用户的聊天记录在这 24 个时间段内出现的次数分别为多少?

### 步骤 1 新建 hive 表

```
CREATE TABLE chatfrequency(`id` string, `conversation_id` string, `sender_id` string, `msg_type` string,
`msg_content` string, `chat_time` string)
ROW FORMAT delimited
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n';
```

### 步骤 2 验证表 chatfrequency 是否创建完成

```
desc chatfrequency;
```

col_name	data_type	comment
id	string	
conversation_id	string	
sender_id	string	
msg_type	string	
msg_content	string	
chat_time	string	

### 步骤 3 从 hdfs 加载数据到 hive 表

```
LOAD DATA INPATH 'hdfs://server1:9000/user/Hadoop/testData/messages.txt' OVERWRITE INTO TABLE
chatfrequency;
```

### 步骤 4 验证数据是否导入成功

```
select count(1) from chatfrequency;
```

```
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
+-----+
| _c0 |
+-----+
| 100000 |
+-----+
```

### 2.1.3.2 参考答案

```
SELECT sender_id,hour(regexp_replace(chat_time, '/', '-')),count(1) AS count
FROM chatfrequency c
GROUP BY hour(regexp_replace(chat_time, '/', '-')),c.sender_id
ORDER BY c.sender_id,count desc;
```

sender_id	_c1	count
1	15	2155
1	19	2138
1	11	2134
1	17	2133
1	7	2124
1	1	2120
1	12	2119
1	10	2116
1	8	2115
1	9	2112
1	13	2110
1	14	2109
1	2	2108
1	18	2095
1	3	2089
1	21	2088
1	20	2078
1	5	2077
1	22	2074
1	4	2060
1	16	2052
1	6	2051
1	23	2047
1	0	2037
2	13	2173
2	1	2153
2	8	2149
2	11	2146
2	17	2129
2	9	2123
2	7	2104
2	10	2092
2	12	2091
2	14	2066
2	4	2056
2	6	2054
2	22	2053
2	21	2050
2	3	2048
2	2	2041
2	19	2038
2	23	2022
2	0	2020
2	20	2018
2	18	2015
2	16	2012
2	15	2004
2	5	2002

48 rows selected (40.198 seconds)  
0: jdbc:hive2://server1:10000>

### 2.1.4 问题二

第一小题: 在所有的聊天内容中, 每个汉字各自出现了多少次? 将结果按照递减的方法进行排序 ( message.txt ) 。

#### 2.1.4.1 参考答案

```
SELECT word,count(1) AS count
FROM (SELECT explode(split(regexp_replace(msg_content,'\u3002',''),('?!\\A|\\z')))) AS word FROM
chatfrequency) w
```

```
GROUP BY word
ORDER BY count DESC;
```

```
INFO : Concurrency mode is
+----+-----+
| word | count |
+----+-----+
| 电   | 3165  |
| 运   | 3163  |
| 列   | 3137  |
| 放   | 3137  |
| 近   | 3135  |
| 产   | 3127  |
| 术   | 3126  |
| 小   | 3125  |
| 济   | 3118  |
| 起   | 3111  |
| 设   | 3111  |
| 件   | 3110  |
| 将   | 3109  |
| 同   | 3109  |
| 每   | 3108  |
| 题   | 3105  |
| 状   | 3105  |
| 已   | 3104  |
| 消   | 3103  |
| 高   | 3101  |
| 参   | 3100  |
| 江   | 3098  |
| 识   | 3096  |
| 平   | 3096  |
| 声   | 3096  |
| 亲   | 3095  |
| 以   | 3093  |
| 称   | 3093  |
| 车   | 3091  |
| 代   | 3086  |
| 完   | 3086  |
| 全   | 3084  |
| 做   | 3083  |
| 织   | 3082  |
| 义   | 3082  |
| 业   | 3081  |
| 深   | 3081  |
| 持   | 3079  |
```

## 2.1.5 问题三

在所有的对话标题中，每个汉字各自出现了多少次？将结果按照递减的方法进行排序（ conversation.txt ）。

### 步骤 1 新建 hive 表

```
CREATE TABLE conversation(`id` string, `title` string, `creator_id` string, `create_time` string)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ';'
LINES TERMINATED BY '\n';
```

### 步骤 2 验证表 conversation 是否创建完成

```
desc conversation;
```

```
+----+-----+
| col_name | data_type | comment |
+----+-----+
| id       | string   |         |
| title    | string   |         |
| creator_id | string   |         |
| create_time | string   |         |
+----+-----+
```

### 步骤 3 加载数据

```
LOAD DATA INPATH 'hdfs://server1:9000/user/Hadoop/testData/conversation.txt' OVERWRITE INTO
TABLE conversation;
```

#### 步骤 4 验证数据是否导入成功

```
select count(1) from conversation;
```

```
INFO : OK
INFO : Concurrency mode is disabled, not creating a lock manager
+-----+
| _c0    |
+-----+
| 10000  |
+-----+
```

### 2.1.5.2 参考答案

```
SELECT title,count(1) AS tcount
FROM (SELECT explode(split(regexp_replace(title,'\u3002',''),'(?!\A|\\z)')) AS title FROM conversation) c
GROUP BY title
ORDER BY tcount DESC;
```

### 2.1.6 问题四

在聊天内容中出现频率较高的汉字在对话标题中各自出现了多少次？将结果以聊天内容中出现频率递减的方式排序。

#### 2.1.6.1 参考答案

```
SELECT * FROM
(SELECT word,count(1) AS wcount
FROM (SELECT explode(split(regexp_replace(msg_content,'\u3002',''),'(?!\A|\\z)')) AS word FROM
chatfrequency) w
GROUP BY word) w
JOIN
(SELECT title,count(1) AS tcount
FROM (SELECT explode(split(regexp_replace(title,'\u3002',''),'(?!\A|\\z)')) AS title FROM conversation) c
GROUP BY title) t
ON (w.word=t.title)
ORDER BY w.wcount DESC;
```



w.word	w.wcount	t.title	t.tcount
电	3165	电	85
运	3163	运	117
列	3137	列	105
放	3137	放	92
近	3135	近	93
产	3127	产	108
术	3126	术	88
小	3125	小	103
济	3118	济	104
起	3111	起	96
设	3111	设	92
件	3110	件	113
将	3109	将	92
间	3109	间	92
每	3108	每	102
题	3105	题	111
状	3105	状	119
已	3104	已	117
消	3103	消	100
高	3101	高	104
参	3100	参	109
江	3098	江	97
识	3096	识	103
平	3096	平	125
声	3096	声	87
亲	3095	亲	106
以	3093	以	96
称	3093	称	108
车	3091	车	108
代	3086	代	98
完	3086	完	73
金	3084	金	79
做	3083	做	103
织	3082	织	104
义	3082	义	98
业	3081	业	103
深	3081	深	122
持	3079	持	94
十	3078	十	101
百	3076	百	98
族	3076	族	97
格	3075	格	117
对	3073	对	112
四	3073	四	106

```

+-----+-----+-----+-----+
500 rows selected (77.966 seconds)
0: jdbc:hive2://server1:10000>

```

## 2.1.7 部署 Kunpeng Hyper Tuner

### 步骤 1 下载 Hyper Tuner

```
cd /home
wget https://mirror.iscas.ac.cn/kunpeng/archive/Tuning_kit/Packages/Hyper-tuner_2.3.T20_linux.tar.gz
```

### 步骤 2 解压

```
tar -zxvf Hyper-tuner_2.3.T20_linux.tar.gz
```

### 步骤 3 临时关闭 SELinux

```
setenforce 0
```

### 步骤 4 安装

**注意：需要自己填写！有两个命令需要填写！**

此时仅安装系统性能分析和系统诊断，故初始选项输入 2，后续步骤的设置默认即可。

```
Start installing, please wait!

Hyper_tuner Config Generate
install tool:
[1] : System Profiler, System Diagnosis, Tuning Assistant and Java Profiler will be installed
[2] : System Profiler, System Diagnosis, Tuning Assistant will be installed
[3] : Java Profiler will be installed
Please enter a number as install tool. (The default install tool is all):2
```

安装完成。

```
Start hyper-tuner service success

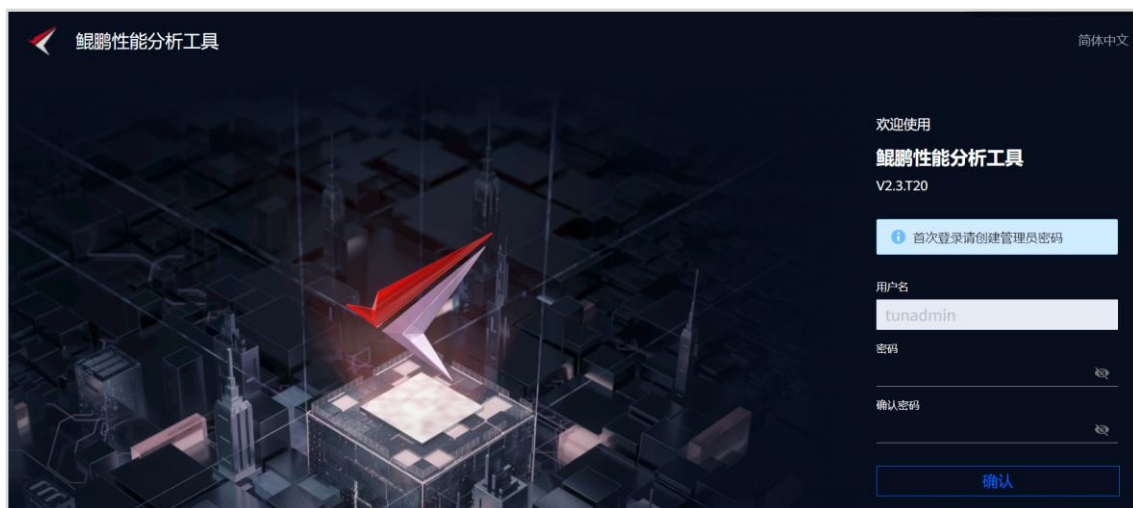
Hyper_tuner install Success
=====
The login URL of Hyper_Tuner is https://192.168.0.70:8086/user-management/#/login
=====
If 192.168.0.70:8086 has mapping IP, please use the mapping IP.
```

如有问题，可以参考

[https://support.huaweicloud.com/hypertuner-install/kunpenginstall\\_06\\_0001.html](https://support.huaweicloud.com/hypertuner-install/kunpenginstall_06_0001.html)。

## 步骤 5 使用

在浏览器输入 <https://IP:8086/user-management/#/login> ，首次登陆需要设置密码。



## 2.1.8 使用 Kunpeng Hyper Tuner 优化

可以参考大数据场景 Spark 组件的调优实践

[https://support.huaweicloud.com/bestpractice-tk-kunpengdevps/kunpengbestspark2\\_21\\_0001.html](https://support.huaweicloud.com/bestpractice-tk-kunpengdevps/kunpengbestspark2_21_0001.html)。

### 步骤 1 选择系统性能分析



## 步骤 2 创建工程



按照以下内容配置参数。

创建工程

\* 工程名称

hive-test

\* 场景选择

通用场景

大数据

分布式存储

HPC

数据库

采集分析服务器上通用的CPU、内存、存储IO、网络IO等资源，以及大数据的典型配置、Top数据

\* 组件

Hive

\* 应用场景

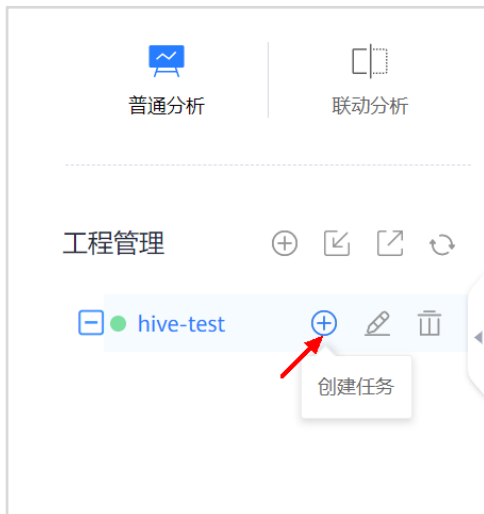
SQL1

\* 选择节点

确认

取消

## 步骤 3 创建任务



按照下图中信息配置数据。

任务名称

hive-test-1

导入模板

分析对象

系统

应用

采集整个系统的数据分析，无需关注系统中有哪些类型的应用在运行，采样时长由配置参数控制，适用于多业务混合运行和有子进程的場景

分析类型

通用分析

全局分析

进程/线程性能分析

热点函数分析

微架构分析

访存分析

I/O分析

资源调度分析

锁与等待分析

系统部件分析

专项分析

通过采集系统软硬件配置信息，以及系统CPU、内存、存储IO、网络IO资源的运行情况，识别系统瓶颈并提供优化建议

采样时长 (s)

-

30

+

(2~300)

采样间隔 (s)

-

1

+

采样间隔应当小于或等于采样时长的1/2且最大为10s

大数据测试工具路径

工具路径

采集Top活跃进程

预约定时启动

立即执行

确认

取消

保存为模板

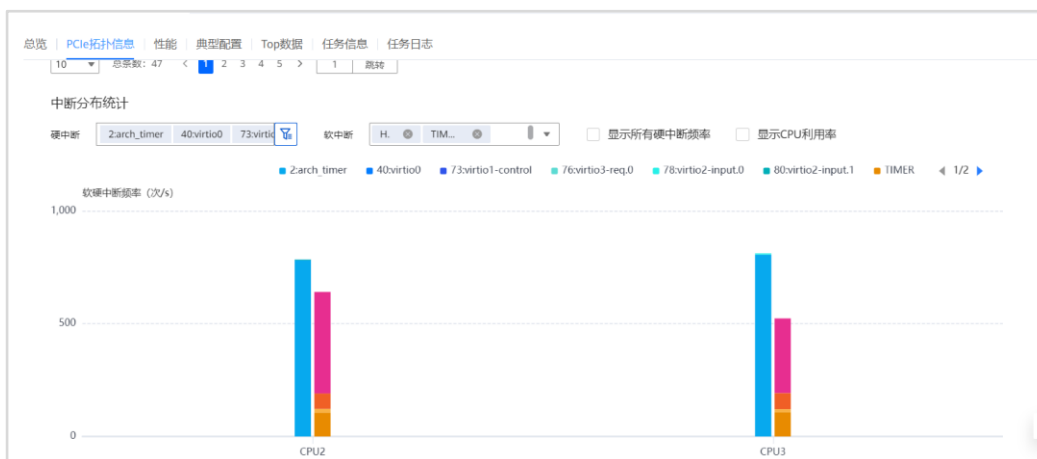
## 步骤 4 性能分析

执行任务完成后，出现分析结果页面，以下为各模块截图。

总览提供优化建议以及各项核心硬件的指标数据。



PCIe 拓扑信息页面提供 NUMA 拓扑图、绑核信息以及中断信息。



性能模块展示 CPU、内存、网络等使用信息图表。



典型配置根据应用场景，提供配置优化建议。

The dashboard provides configuration optimization suggestions based on the application scenario. It includes sections for hardware and software configuration, system settings, and component settings.

- 硬件及软件配置:** A table showing hardware and software configurations. The table has columns for hardware (e.g., smmu, cpu\_prefetching, raid0, rdcache, wrcache, iommu), system (e.g., rx\_buff, ring\_buff\_rx, ring\_buff\_tx, LRO, irqbalance\_enable, imbalance\_numnodes), and components (e.g., ResourceManager Java heap size (MB), NodeManager Java heap size (MB), yarn.nodemanager.resource.cpu-vcores, yarn.nodemanager.resource.memory-mb, yarn.nodemanager.numa-awareness.enabled, yarn.nodemanager.numa-awareness.type).
- 节点列表:** A table showing a list of nodes. The table has columns for node ID, IP address, and other details. The table shows a single node with IP address 10.71.44.96.

Top 数据定时采集 TOP 数据内容。

The dashboard displays the Top data collection results. It includes a table with the following columns: PID, USER, PR, NI, VIRT, RES, SHR, S, %CPU, %MEM, TIME+, and COMMAND. The table lists the top 25 processes running on the system.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	15511 root	20	0	4821952	233024	44352	S	205.9	0.1	0:02.60	java
2	15613 malluna	20	0	110544	4416	1728	R	100.0	0.0	0:00.20	pidstat
3	3784 root	20	0	33.5g	1.1g	44994	S	35.3	0.4	26:33.59	java
4	15723 malluna	20	0	319360	43200	4224	R	29.4	0.0	0:00.05	python3
5	15586 malluna	20	0	170752	42368	4544	R	23.5	0.0	0:00.04	python3
6	1 root	20	0	164608	16384	6080	S	17.6	0.0	48:03.72	systemd
7	15592 malluna	20	0	319424	46016	7040	S	17.6	0.0	0:00.04	python3
8	15772 malluna	20	0	111168	2816	1728	S	17.6	0.0	0:00.03	pidstat
9	4678 dbus	20	0	18560	7808	3712	S	11.8	0.0	30:08.92	dbus-daemon
10	15589 malluna	20	0	318400	42432	4288	S	11.8	0.0	0:00.04	python3
11	15617 malluna	20	0	118336	8064	3712	R	11.8	0.0	0:00.04	top
12	15638 root	20	0	3776	3008	1472	R	11.8	0.0	0:00.02	config_ipa+
13	4673 root	20	0	5888	4992	2688	S	5.9	0.0	12:41.26	systemd-lo+
14	4687 root	20	0	359424	22208	11904	S	5.9	0.0	4:12.01	NetworkMan+
15	4709 polkitd	20	0	544000	18560	9472	S	5.9	0.0	6:16.01	polkitd
16	5078 root	20	0	684352	80064	43136	S	5.9	0.0	3:56.75	rsyslogd
17	5472 malluna	20	0	563456	15616	12032	S	5.9	0.0	0:07.55	nagent
18	15495 malluna	20	0	318336	53824	15744	S	5.9	0.0	0:00.41	python3
19	15711 malluna	20	0	110848	2688	1856	S	5.9	0.0	0:00.01	sadc
20	15762 malluna	20	0	319616	44032	4992	S	5.9	0.0	0:00.01	python3
21	15774 malluna	20	0	319616	44480	5440	R	5.9	0.0	0:00.01	python3
22	24406 root	20	0	0	0	0	I	5.9	0.0	0:00.03	kworker/43+
23	2 root	20	0	0	0	0	S	0.0	0.0	0:03.96	kthreadd
24	4 root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:+
25	5 root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/u1+

## 步骤 5 （补充）典型配置调优

由于典型配置功能模块暂不支持云虚拟机，Hive 相关调优项。



通过 `set hive.exec.parallel=true;`

或修改 `/usr/local/hive/conf/hive-site.xml` 文件中的 `hive.exec.parallel` 值，

将任务改为并行执行，提高处理任务的效率，缩短处理任务的时间。

## 步骤 6 优化结果

以问题四为例：

w.word	w.wcount	t.title	t.tcount
电	3165	电	85
运	3163	运	117
列	3137	列	105
放	3137	放	92
近	3135	近	93
产	3127	产	108
术	3126	术	88
小	3125	小	103
济	3118	济	104
起	3111	起	96
设	3111	设	92
件	3110	件	113
将	3109	将	92
同	3109	同	92
每	3108	每	102
题	3105	题	111
状	3105	状	119
已	3104	已	117
消	3103	消	100
高	3101	高	104
参	3100	参	109
江	3098	江	97
识	3096	识	103
平	3096	平	125
声	3096	声	87
亲	3095	亲	106
以	3093	以	96
称	3093	称	108
车	3091	车	108
代	3086	代	98
完	3086	完	73
金	3084	金	79
做	3083	做	103
织	3082	织	104
义	3082	义	98
业	3081	业	103
深	3081	深	122
持	3079	持	94
十	3078	十	101
百	3076	百	98
族	3076	族	97



```
| 争 | 2864 | 争 | 97 |  
| 飞 | 1519 | 飞 | 53 |  
| 的 | 1504 | 的 | 52 |  
+-----+-----+-----+-----+  
500 rows selected (65.129 seconds)  
0: jdbc:hive2://server1:10000>
```

## 2.2 思考题

在实验中设置了 Hive 的 job 为并发执行，那么在并发执行时，资源消耗和原先的顺序执行是否有所不同？

参考答案：

可在执行问题四中的 HiveQL 语句时，新开命令行窗口，使用 top 命令观察资源使用情况。通过观察可得，资源消耗比顺序执行时更多。