

# Elastic Memory Management for Cloud Data Analytics

## 1 实验环境搭建

### 1.1 论文实验环境

通过对论文内容的解读与分析，我们明确了作者的全部实验都是在 Amazon EC2 平台上进行的，使用一个 r3.4xlarge 实例运行 TPC-H 查询计划，计划里面包含的查询语句都是使用 MyriaL 查询语言编写的，而计划本身则是由 Raco 生成的。然后通过 JVM 中动态的调整堆大小以及动态的分配内存空间和估计运行时间权值来实现 Elastic Memory Manage，而这些都是通过 Docker 容器进行封装的。此外，作者还使用 Spark1、Spark2 和 Myria 进行了自动内存管理的性能比较。

### 1.2 重现的实验环境

由于条件限制，我们拟在 Ubuntu 下搭建 Docker 环境，搭建 JVM 环境，利用容器运行 Myria 查询。

#### (1) Docker 配置

```
$sudo apt-get update
```

```
$sudo apt-get install docker.io
```

```
root@zzhPC:/home/zzh# docker -v
Docker version 1.12.6, build 78d1802
root@zzhPC:/home/zzh#
```

#### (2) JDK 配置

下载 jdk-8u151-linux-x64.tar.gz

```
cd /usr/lib/java
```

```
sudo tar -zxvf /home/zzh/下载/jdk-8u151-linux-x64.tar.gz
```

```
vim /etc/profile
```

添加

```
export JAVA_HOME=/usr/lib/java/jdk1.8.0_151
```

```
export CLASSPATH=.:$JAVA_HOME/lib:$JAVA_HOME/jre/lib:$CLASSPATH
```

```
export PATH=$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$PATH
```

```
sudo source /etc/profile
```

```
root@zzhPC:/usr/lib/jvm# java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
root@zzhPC:/usr/lib/jvm#
```

### (3) 镜像配置

\$docker search jdk

\$docker pull docker-oracle-jdk-8

```
root@zzhPC:/home/zzh# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
jvm                  latest             aa8a60aa18f5       6 days ago         506.4 MB
kurron/docker-oracle-jdk-8  latest            033d08e1709e       15 months ago      656.1 MB
root@zzhPC:/home/zzh#
```

### (4) 容器启动

\$docker run -d kurron/docker-oracle-jdk-8 /bin/bash -c "while true;do sleep 1;done"

```
root@zzhPC:/home/zzh# docker run -d kurron/docker-oracle-jdk-8 /bin/bash -c "while true;do sleep 1;done"
4a79b85e7c4fe7794d965ce8cfb54a72b1fffa076c72c00dbdb062ed167adb1d
root@zzhPC:/home/zzh# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS              NAMES
4a79b85e7c4f        kurron/docker-oracle-jdk-8  "/bin/bash -c 'while "  13 seconds ago     Up 11 seconds                               stupefted_hodgkin
root@zzhPC:/home/zzh#
```

### (5) 配置本地 Myria

准备 sqlite

sudo apt-get install sqlite3

下载 Myria

git clone https://github.com/uwescience/myria

建立 Myria jar (在 myria 文件夹下执行, 会生成 build/libs/myria-0.1-all.jar, 确保其存在)

./gradlew clean shadowJar check

```
root@zzhPC:/usr/lib/myria# cd build
root@zzhPC:/usr/lib/myria/build# ls
dependency-cache  google-java-format  jacoco  libs  main  reports  resources  test  test-results  tmp
root@zzhPC:/usr/lib/myria/build# cd libs
root@zzhPC:/usr/lib/myria/build/libs# ls
myria-0.1-all.jar
root@zzhPC:/usr/lib/myria/build/libs#
```

运行集群

cd myriadeploy

./launch\_local\_cluster

下载数据集

curl localhost:8753/dataset/user-jwang/program-global\_join/relation-smallTable\_join\_smallTable/data

获取数据集(源数据文件 ingest\_smallTable.json)

cd jsonQueries/getting\_started

vim ./ingest\_smallTable.json

"columnTypes" : ["LONG\_TYPE", "LONG\_TYPE"],

"columnNames" : ["col1", "col2"]

运行查询

curl -i -XPOST localhost:8753/query -H "Content-type: application/json" -d

@./global\_join.json

SQL 的等价物

SELECT t1.col1, t2.col2

FROM smallTable AS t1, smallTable AS t2

WHERE t1.col2 = t2.col1

## （6）Hadoop 配置

创建 Hadoop 用户

```
sudo useradd -m hadoop -s /bin/bash
```

```
sudo passwd hadoop
```

```
sudo adduser hadoop sudo
```

安装 SSH、配置 SSH 无密码连接

```
sudo apt-get install openssh-server
```

```
ssh localhost
```

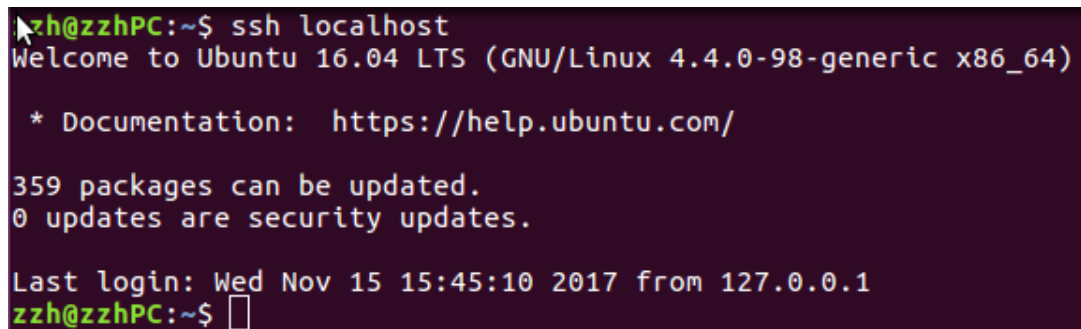
```
exit
```

```
cd ~/.ssh
```

```
ssh-keygen -t rsa
```

```
cat ./id_rsa.pub >> ./authorized_keys
```

```
ssh localhost
```



```
zzh@zzhPC:~$ ssh localhost
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-98-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

359 packages can be updated.
0 updates are security updates.

Last login: Wed Nov 15 15:45:10 2017 from 127.0.0.1
zzh@zzhPC:~$
```

配置 hadoop

下载 hadoop-2.6.5 (mirror.bit.edu.cn/apache/hadoop/common/hadoop-2.6.5)

```
sudo tar -zxf hadoop-2.6.5.tar.gz -C /usr/local
```

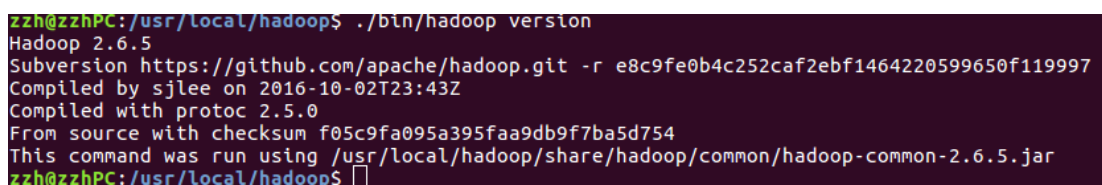
```
cd /usr/local
```

```
sudo mv ./hadoop-2.6.5 ./hadoop
```

```
sudo chown -R hadoop ./hadoop
```

```
cd /usr/local/hadoop
```

```
./bin/hadoop version
```



```
zzh@zzhPC:/usr/local/hadoop$ ./bin/hadoop version
Hadoop 2.6.5
Subversion https://github.com/apache/hadoop.git -r e8c9fe0b4c252caf2ebf1464220599650f119997
Compiled by sjlee on 2016-10-02T23:43Z
Compiled with protoc 2.5.0
From source with checksum f05c9fa095a395faa9db9f7ba5d754
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-2.6.5.jar
zzh@zzhPC:/usr/local/hadoop$
```

## （7）配置 Spark1.6

```
sudo git clone git://github.com/apache/spark.git -b branch-1.6
```

```
sudo mv spark /usr/local
```

```
cd /usr/local/spark
```

```
cp ./conf/spark-env.sh.template ./conf/spark-env.sh
```

```
sudo vim ./conf/spark-env.sh
```

```
export SPARK_DIST_CLASSPATH=$(/usr/local/hadoop/bin/hadoop classpath)
```

## 2 需要完成的实验内容

(1) 对比分析在 Spark1.1, Spark2.0 和 Myria 上, GC (garbage collection) 对查询执行时间的影响。(即通过实验得出论文中 Figure3 的结果)

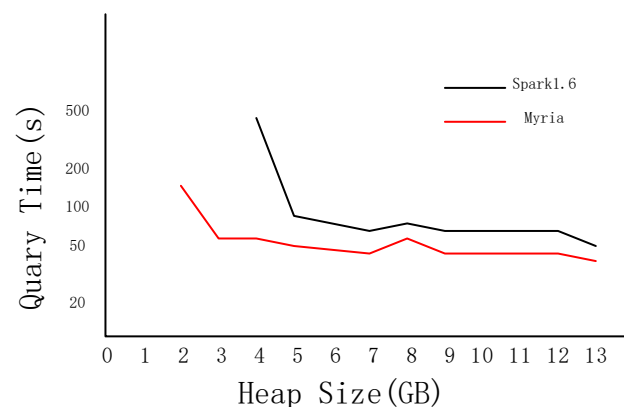
(2) 通过限制 Total Memory 的大小, 比较原生的内存管理系统在并行度分别为 1、4、8 以及 Elastic 和经过改进的 Elastic-Resubmit 系统中任务的 Elapsed Times。(即通过实验得出论文中 Figure4 的结果)

(3) 在 30 秒的延迟下重复 (2) 中的实验。(即通过实验得出论文中 Figure6 的结果)

(4) 通过调整 Total Memory 的大小, 比较原生的内存管理系统在并行度为 8 以及 Elastic 在 U=1/8GB、U=1/12GB、U=1/16GB、U=500MB 和 U=1000MB 时的 RSS、GC Time Ratio 和 Query Time Ratio。(即论文中 Figure5 的结果)

### 3 已经完成的实验内容

(1) 通过在 Spark1.6 和 Myria 上调整 Heap Size 得到的查询时间如下图所示:



因为只取了 Heap Size 为 1GB 及其整数倍的值, 所以图像在平滑度上与论文中的图会有一些差别, 但整体趋势跟原文接近。

(2) Elastic Memory System 没有能布置起来, 只能在虚拟机上进行内存分配和执行 Myria 查询。

### 4. 结论

实验没有能完全重现, 但是根据已经成功的实验大致能够推断出 Elastic Memory System 对云集群是非常有用的。