Tachyon 介绍及安装部署

目 录

Tachyon 介绍及安装部署

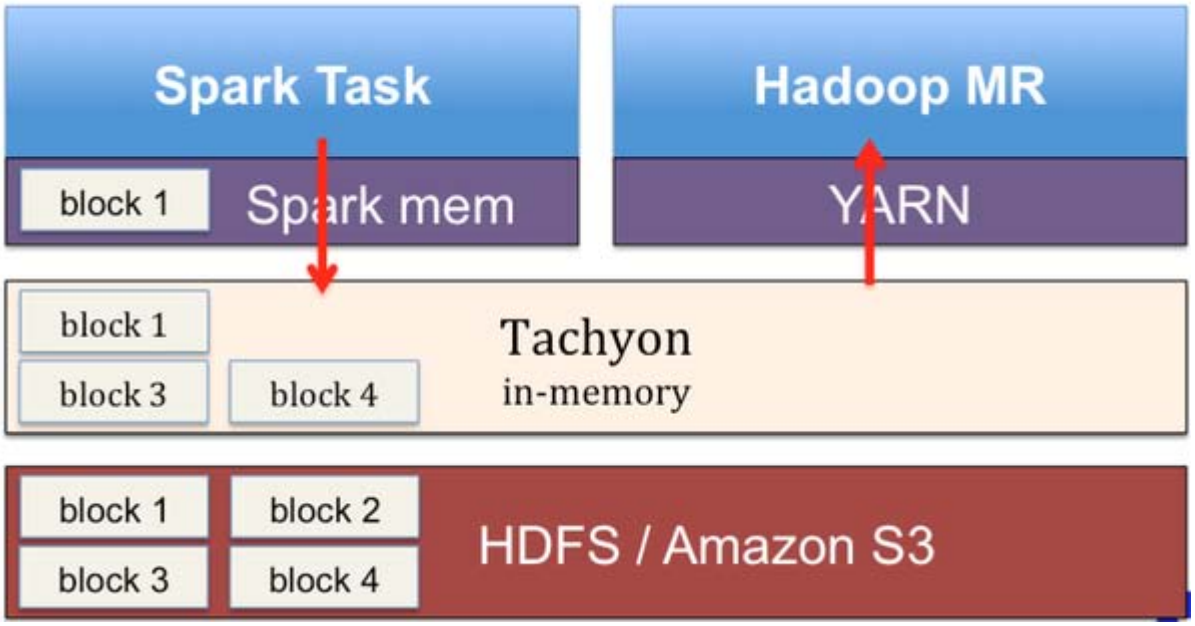
1 Tachyon 介绍

1.1 Tachyon 简介

随着实时计算的需求日益增多，分布式内存计算也持续升温，怎样将海量数据近乎实时地处理，或者说怎样把离线批处理的速度再提升到一个新的高度是当前研究的重点。近年来，内存的吞吐量成指数倍增长，而磁盘的吞吐量增长缓慢，那么将原有计算框架中文件落地磁盘替换为文件落地内存，也是提高效率的优化点。

目前已经使用基于内存计算的分布式计算框架有：Spark、Impala 及 SAP 的 HANA 等。但是其中不乏一些还是有文件落地磁盘的操作，如果能让这些落地磁盘的操作全部落地到一个共享的内存中，那么这些基于内存的计算框架的效率会更高。

Tachyon是AmpLab的李浩源所开发的一个分布式内存文件系统，可以在集群里以访问内存的速度来访问存在Tachyon里的文件。Tachyon是架构在最底层的分布式文件存储和上层的各种计算框架之间的一种中间件，其主要职责是将那些不需要落地到DFS里的文件落地到分布式内存文件系统中来达到共享内存，从而提高效率。同时可以减少内存冗余、GC时间等，Tachyon的在大数据中层次关系如下图所示：



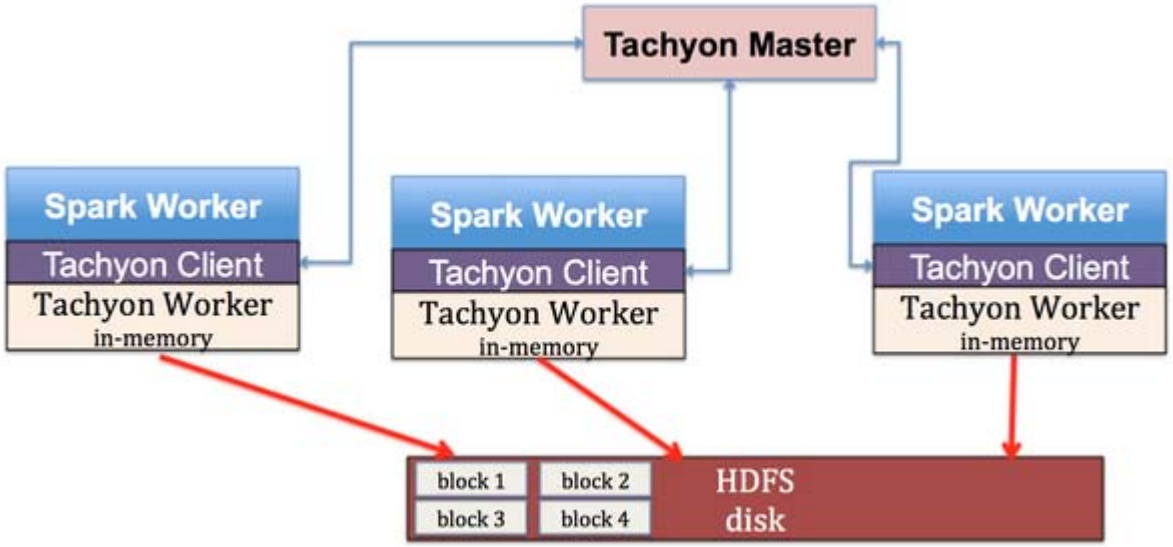
Tachyon 允许文件以内存的速度在集群框架中进行可靠的共享 就像 Spark 和 MapReduce 那样。通过利用信息继承、内存侵入，Tachyon 获得了高性能。Tachyon 工作集文件缓存在内存

中，并且让不同的 Jobs/Queries 以及框架都能以内存的速度来访问缓存文件。因此，Tachyon 可以减少那些需要经常使用数据集通过访问磁盘来获得的次数。

1.2 Tachyon 系统架构

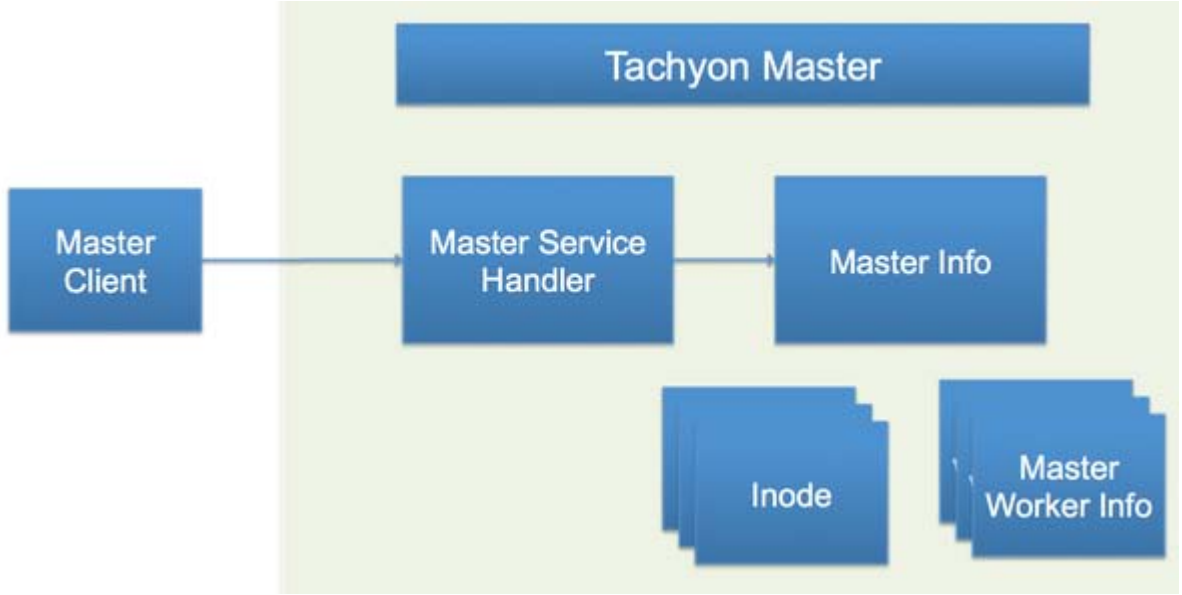
1.2.1 系统架构

Tachyon 在 Spark 平台的部署：总的来说，Tachyon 有三个主要的部件：Master， Client，与 Worker。在每个 Spark Worker 节点上，都部署了一个 Tachyon Worker，Spark Worker 通过 Tachyon Client 访问 Tachyon 进行数据读写。所有的 Tachyon Worker 都被 Tachyon Master 所管理，Tachyon Master 通过 Tachyon Worker 定时发出的心跳来判断 Worker 是否已经崩溃以及每个 Worker 剩余的内存空间量。



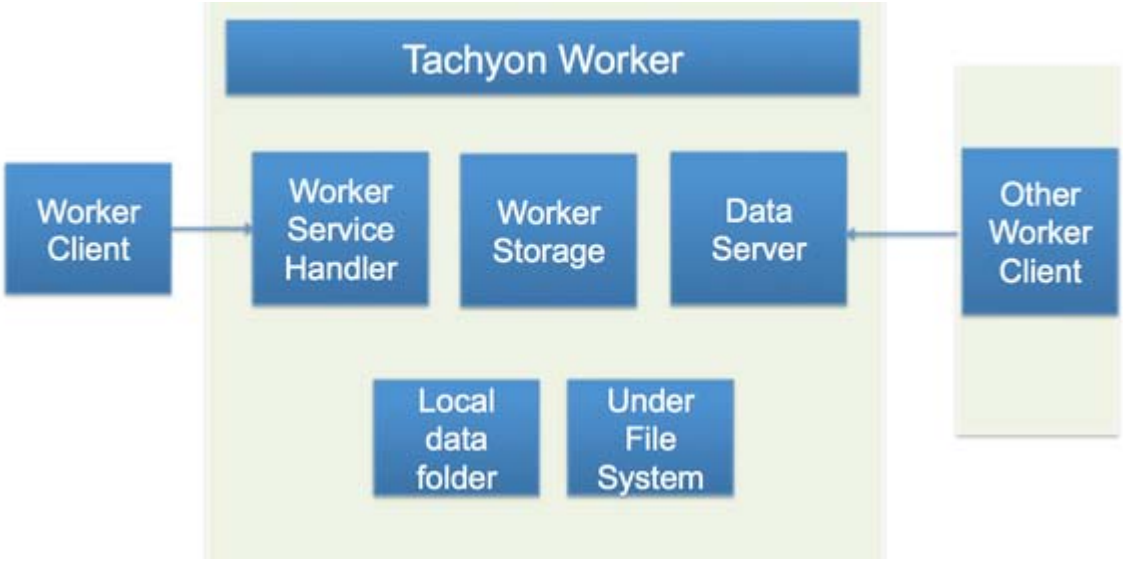
1.2.2 Tachyon Master 结构

Tachyon Master 的结构其主要功能如下：首先，Tachyon Master 是个主管理器，处理从各个 Client 发出的请求 ,这一系列的工作由 Service Handler 来完成。这些请求包括 :获取 Worker 的信息，读取 File 的 Block 信息， 创建 File 等等；其次，Tachyon Master 是个 Name Node，存放着所有文件的信息，每个文件的信息都被封装成一个 Inode，每个 Inode 都记录着属于这个文件的所有 Block 信息。在 Tachyon 中，Block 是文件系统存储的最小单位，假设每个 Block 是 256MB，如果有一个文件的大小是 1GB，那么这个文件会被切为 4 个 Block。每个 Block 可能存在多个副本，被存储在多个 Tachyon Worker 中，因此 Master 里面也必须记录每个 Block 被存储的 Worker 地址；第三，Tachyon Master 同时管理着所有的 Worker，Worker 会定时向 Master 发送心跳通知本次活跃状态以及剩余存储空间。Master 是通过 Master Worker Info 去记录每个 Worker 的上次心跳时间，已使用的内存空间，以及总存储空间等信息。



1.2.3 Tachyon Worker 结构

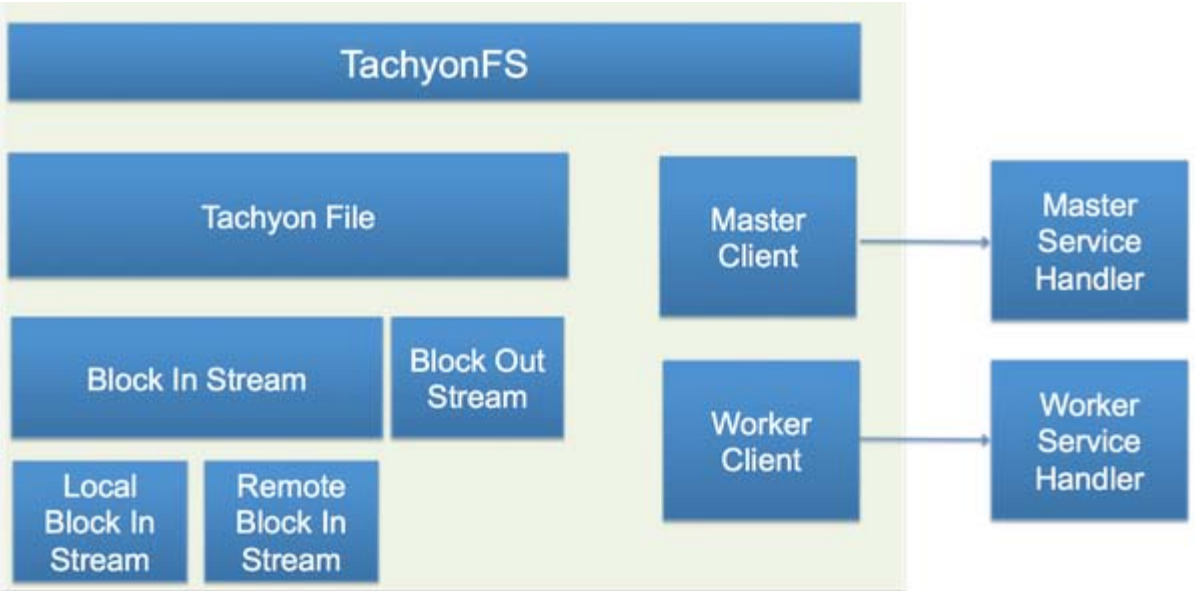
Tachyon Worker 主要负责存储管理：首先，Tachyon Worker 的 Service Handler 处理来自 Client 发来的请求 ,这些请求包括 :读取某个 Block 的信息 ,缓存某个 Block ,锁住某个 Block，向本地内存存储要求空间等等。第二，Tachyon Worker 的主要部件是 Worker Storage，其作用是管理 Local Data（本地的内存文件系统）以及 Under File System（Tachyon 以下的磁盘文件系统，比如 HDFS）。第三，Tachyon Worker 还有个 Data Server 以便处理其他的 Client 对其发起的数据读写请求。当由请求达到时，Tachyon 会先在本地的内存存储找数据，如果没有找到则会尝试去其他的 Tachyon Worker 的内存存储中进行查找。如果数据完全不在 Tachyon 里，则需要通过 Under File System 的接口去磁盘文件系统（HDFS）中读取。



1.2.4 Tachyon Client 结构

Tachyon Client 主要功能是向用户抽象一个文件系统接口以屏蔽掉底层实现细节。首先，Tachyon Client 会通过 Master Client 部件跟 Tachyon Master 交互，比如可以向 Tachyon

Master 查询某个文件的某个 Block 在哪里。Tachyon Client 也会通过 Worker Client 部件跟 Tachyon Worker 交互， 比如向某个 Tachyon Worker 请求存储空间。在 Tachyon Client 实现中最主要的是 Tachyon File 这个部件。在 Tachyon File 下实现了 Block Out Stream，其主要用于写本地内存文件；实现了 Block In Stream 主要负责读内存文件。在 Block In Stream 内包含了两个不同的实现：Local Block In Stream 主要是用来读本地的内存文件，而 Remote Block In Stream 主要是读非本地的内存文件。请注意，非本地可以是在其它的 Tachyon Worker 的内存文件里，也可以是在 Under File System 的文件里。



1.2.5 场景说明

现在我们通过一个简单的场景把各个部件都串起来 :假设一个 Spark 作业发起了一个读请求，它首先会通过 Tachyon Client 去 Tachyon Master 查询所需要的 Block 所在的位置。如果所在的 Block 不在本地的 Tachyon Worker 里，此 Client 则会通过 Remote Block In Stream 向别的 Tachyon Worker 发出读请求，同时在 Block 读入的过程中，Client 也会通过 Block Out Stream 把 Block 写入到本地的内存存储里，这样就可以保证下次同样的请求可以由本机完成。

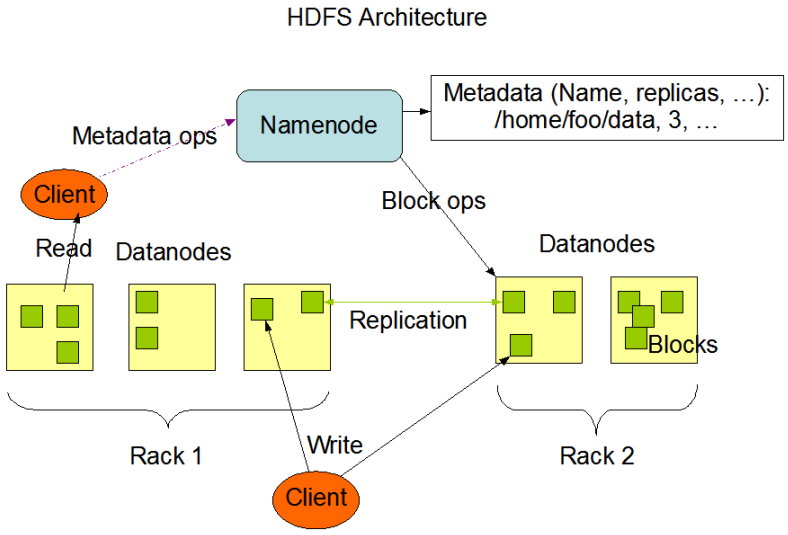
1.3 HDFS 与 Tachyon

HDFS（Hadoop Distributed File System）是一个分布式文件系统。HDFS 具有高容错性（fault-tolerant）特点，并且设计用来部署在低廉的硬件上。而且它提供高吞吐量（high throughput）来访问应用程序的数据，适合那些有着超大数据集（large data set）的应用程序。HDFS 放宽了 POSIX 的要求，这样可以实现以流的形式访问（streaming access）文件系统中的数据。

HDFS 采用 Master/Slave 架构。HDFS 集群是由一个 Namenode 和一定数目的 Datanode 组成的。Namenode 是一台中心服务器，负责管理文件系统的名字空间（namespace）以及客

户端对文件的访问。集群中的 Datanode 一般是一个节点一个，负责管理它所在节点上的存储。HDFS 暴露了文件系统的名字空间，用户能够以文件的形式在上面存储数据。从内部看，一个文件其实被分成一个或多个数据块，这些块存储在一组 Datanode 上。Namenode 执行文件系统的名字空间操作，比如打开、关闭、重命名文件或目录，它也负责确定数据块到具体 Datanode 节点的映射。Datanode 负责处理文件系统客户端的读写请求，在 Namenode 的统一调度下对数据块进行创建、删除和复制。

HDFS 架构示意图如下图所示。



Namenode 和 Datanode 被设计成可以在普通的商用机器上运行，这些机器一般运行着 GNU/Linux 操作系统。HDFS 采用 Java 语言开发，因此任何支持 Java 的机器都可以部署 Namenode 或 Datanode。由于采用了可移植性极强的 Java 语言，使得 HDFS 可以部署到多种类型的机器上。一个典型的部署场景是一台机器上只运行一个 Namenode 实例，而集群中的其他机器则分别运行一个 Datanode 实例。这种架构并不排斥在一台机器上运行多个 Datanode，只不过这样的情况比较少见。

集群中单一 Namenode 的结构大大简化了系统的架构。Namenode 是所有 HDFS 元数据的仲裁者和管理者，这样用户数据永远不会流过 Namenode。

对比 HDFS 和 Tachyon，首先从两者的存储结构来看，HDFS 设计为用来存储海量文件的分布式系统，Tachyon 设计为用来缓存常用数据的分布式内存文件系统。从这点来看，Tachyon 可以认为是操作系统层面上的 Cache，HDFS 可以认为是磁盘。

在可靠性方面，HDFS 采用副本技术来保证出现系统宕机等意外情况时文件访问的一致性以及可靠性；而 Tachyon 是依赖于底层文件系统的可靠性来实现自身文件的可靠性的。由于相对于磁盘资源来说，内存是非常宝贵的，所以 Tachyon 通过在其 underfs（一般使用 HDFS）上写入 CheckPoint 日志信息来实现对文件系统的可恢复性。

从文件的读取以及写入方式来看，Tachyon 可以更好地利用本地模式来读取文件信息，当文件读取客户端和文件所在的 Worker 位于一台机器上时，客户端会直接绕过 Worker 直接读取对应的物理文件，减少了本机的数据交互。而 HDFS 在遇到这样的情况时，会通过本地 Socket 进行数据交换，这也会有一定的系统资源开销。在写入文件时，HDFS 只能写入磁盘，而 Tachyon 却提供了 5 种数据写入模式用以满足不同需求。

2 Tachyon 编译部署

Tachyon 目前的最新发布版为 0.7.1 ,其官方网址为 http://tachyon-project.org/。Tachyon 文件系统有 3 种部署方式：单机模式、集群模式和高可用集群模式，集群模式相比于高可用集群模式区别在于多 Master 节点。下面将介绍单机和集群环境下去安装、配置和使用 Tachyon。

2.1 编译 Tachyon

2.1.1 下载并上传源代码

第一步 下载到 Tachyon 源代码：

对于已经发布的版本可以直接从 github 下载 Tachyon 编译好的安装包并解压 ,由于 Tachyon 与 Spark 版本有对应关系，另外该系列搭建环境为 Spark1.1.0，对应下载 Tachyon0.5.0，版本对应参考 http://tachyon-project.org/documentation/Running-Spark-on-Tachyon.html 描述：

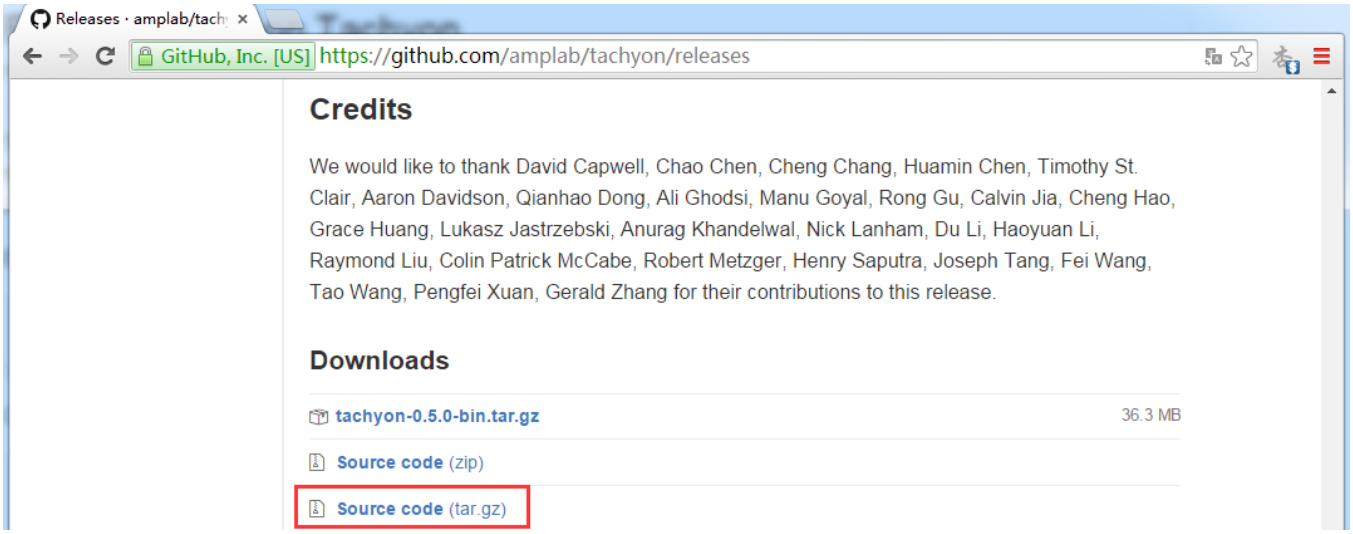
Running Spark on Tachyon

Compatibility

If you plan to run Spark on Tachyon, the following version pairings will work together out-of-the-box. If you plan to use a different version than the default supported version, please recompile Spark with the right version of tachyon-client by changing the version in `spark/core/pom.xml`.

Spark Version	Tachyon Version
1.0.x and Below	v0.4.1
1.1.x	v0.5.0
1.2.x	v0.5.0
1.3.x	v0.5.0
1.4.x	v0.6.4
1.5.x and Above	v0.7.1

下载地址为 https://github.com/amplab/tachyon/releases ，为以下演示我们在这里下载的是 tachyon-0.5.0.tar.gz 源代码包，文件大小为 831K，如下图所示：



第二步 在主节点上解压缩

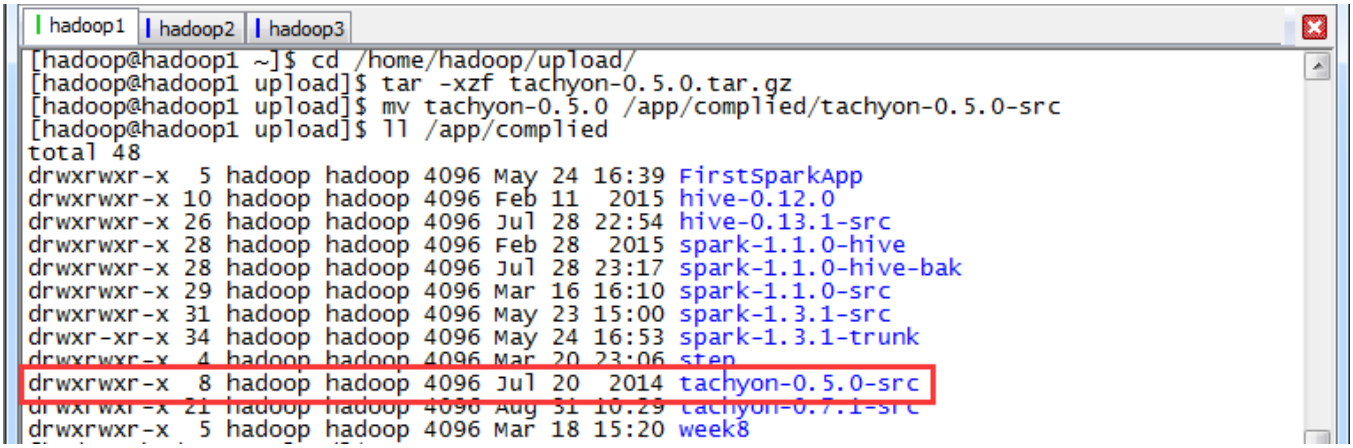
$cd /home/hadoop/upload/

$tar -xzf tachyon-0.5.0.tar.gz

第三步 把 tachyon-0.5.0.tar.gz 改名并移动到/app/complied 目录下

$mv tachyon-0.5.0 /app/complied/tachyon-0.5.0-src

$ll /app/complied



2.1.2 编译代码

为了更好地契合用户的本地环境，如 Java 版本、Hadoop 版本或其他一些软件包的版本，可以下载 Tachyon 源码自行编译。Tachyon 开源在 GitHub 上，可以很方便地获得其不同版本的源码。Tachyon 项目采用 Maven 进行管理，因此可以采用 mvn package 命令进行编译打包。编译 Tachyon 源代码的时候 ,需要从网上下载依赖包 ,所以整个编译过程机器必须保证在联网状态。编译执行如下脚本：

$cd /app/complied/tachyon-0.5.0-src

$export MAVEN_OPTS="-Xmx2g -XX:MaxPermSize=512M -XX:ReservedCodeCacheSize=512m"

$mvn clean package -Djava.version=1.7 -Dhadoop.version=2.2.0 -DskipTests

```
hadoop1
[hadoop@hadoop1 tachyon-0.5.0-src]$ export MAVEN_OPTS="-Xmx2g -XX:MaxPermSize=512M -XX:ReservedCodeCacheSize=512m"
[hadoop@hadoop1 tachyon-0.5.0-src]$ mvn clean package -Djava.version=1.7 -Dhadoop.version=2.2.0 -DskipTests
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Build Order:
[INFO]
[INFO] Tachyon Project Parent
[INFO] Tachyon Project Core
[INFO] Tachyon Project Client
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Tachyon Project Parent 0.5.0
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.4.1:clean (default-clean) @ tachyon-parent ---
[INFO]
[INFO] --- maven-enforcer-plugin:1.0:enforce (enforce-maven) @ tachyon-parent ---
```

整个编译过程编译了约 3 个任务，整个过程耗时大约 4 分钟。

```
hadoop1
[INFO] --- maven-jar-plugin:2.3.2:jar (default-jar) @ tachyon-client ---
Downloading: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-archiver/2.0.1/plexus-archiver-2.0.1.jar
Downloaded: http://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-archiver/2.0.1/plexus-archiver-2.0.1.jar
[WARNING] JAR will be empty - no content was marked for inclusion!
[INFO] Building jar: /app/complied/tachyon-0.5.0-src/client/target/tachyon-client-0.5.0.jar
[INFO]
[INFO] --- maven-assembly-plugin:2.4:single (make-assembly) @ tachyon-client ---
[INFO] Building jar: /app/complied/tachyon-0.5.0-src/client/target/tachyon-client-0.5.0-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Tachyon Project Parent ........................... SUCCESS [1.244s]
[INFO] Tachyon Project Core ............................. SUCCESS [3:01.486s]
[INFO] Tachyon Project Client ........................... SUCCESS [36.544s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3:39.571s
[INFO] Finished at: Mon Aug 31 16:31:20 CST 2015
[INFO] Final Memory: 69M/647M
[INFO] ------------------------------------------------------------------------
```

使用如下命令查看编译后该 Tachyon 项目大小为 72M

$cd /app/complied/tachyon-0.5.0-src

$du -s /app/complied/tachyon-0.5.0-src

```
hadoop1
[hadoop@hadoop1 ~]$ cd /app/complied/tachyon-0.5.0-src
[hadoop@hadoop1 tachyon-0.5.0-src]$ du -s /app/complied/tachyon-0.5.0-src
72000   /app/complied/tachyon-0.5.0-src
[hadoop@hadoop1 tachyon-0.5.0-src]$
```

完成这一步后，我们就得到了能够运行在用户本地环境的 Tachyon，下面我们分别介绍如何在单机和分布式环境下配置和启动 Tachyon ,在进行部署之前先把编译好的文件复制到/app/hadoop 下并把文件夹命名为 Tachyon-0.5.0：

$cd /app/complied

$cp -r tachyon-0.5.0-src /app/hadoop/tachyon-0.5.0

$ll /app/hadoop

```
hadoop1
[hadoop@hadoop1 ~]$ cd /app/complied
[hadoop@hadoop1 complied]$ cp -r tachyon-0.5.0-src /app/hadoop/tachyon-0.5.0
[hadoop@hadoop1 complied]$ ll /app/hadoop
total 32
drwxr-xr-x 13 hadoop hadoop 4096 Jan 15  2015 hadoop-2.2.0
drwxr-xr-x 13 hadoop hadoop 4096 May 22 15:16 hadoop-2.4.1
drwxrwxr-x  9 hadoop hadoop 4096 Feb 15  2015 hive-0.13.1
drwxr-xr-x 12 hadoop hadoop 4096 Apr 10  2014 shark-0.9.1
drwxrwxr-x 28 hadoop hadoop 4096 Feb  6  2015 spark-0.9.1
drwxrwxr-x 11 hadoop hadoop 4096 Aug 10 11:20 spark-1.1.0
drwxrwxr-x  8 hadoop hadoop 4096 Aug 31 16:34 tachyon-0.5.0
drwxrwxr-x 21 hadoop hadoop 4096 Aug 28 15:08 tachyon-0.7.1
[hadoop@hadoop1 complied]$
```

2.2 单机部署 Tachyon

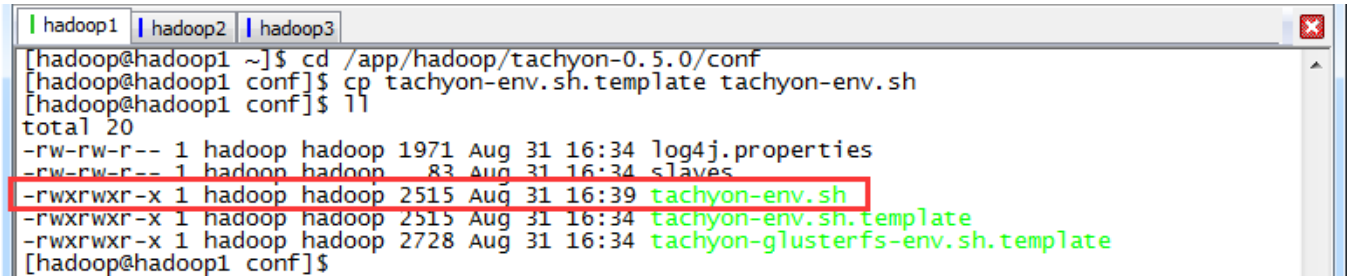
这里要注意一点，Tachyon 在单机（local）模式下启动时会自动挂载 RamFS，所以请保证使用的账户具有 sudo 权限。

【注】编译好的 Tachyon 将本系列附属资源/install 中提供，具体名称为 10.tachyon-0.5.0-hadoop2.2.0-complied.zip

2.2.1 配置 Tachyon

Tachyon 相关配置文件在$TACHYON_HOME/conf 目录下，在 workers 文件中配置需要启动 TachyonWorker 的节点，默认是 localhost，所以在单机模式下不用更改（在 Tachyon-0.5.0 版本中，该文件为 slaves）。在这里需要修改 tachyon-env.sh 配置文件，具体操作是将 tachyon-env.sh.template 复制为 tachyon-env.sh：

$cd /app/hadoop/tachyon-0.5.0/conf
$cp tachyon-env.sh.template tachyon-env.sh
$ll
$vi tachyon-env.sh



并在 tachyon-env.sh 中修改具体配置，下面列举了一些重要的配置项：

- **JAVA_HOME**：系统中 java 的安装路径
- **TACHYON_MASTER_ADDRESS**：启动 TachyonMaster 的地址，默认为 localhost，所以在单机模式下不用更改
- **TACHYON_UNDERFS_ADDRESS**：Tachyon 使用的底层文件系统的路径，在单机模式下可以直接使用本地文件系统，如"/tmp/tachyon"，也可以使用 HDFS，如"hdfs://ip:port"
- **TACHYON_WORKER_MEMORY_SIZE**：每个 TachyonWorker 使用的 RamFS 大小

```
hadoop1
export JAVA_HOME=/usr/lib/java/jdk1.7.0_55
export JAVA="$JAVA_HOME/bin/java"
export TACHYON_MASTER_ADDRESS=localhost
export TACHYON_UNDERFS_ADDRESS=$TACHYON_HOME/underFSStorage
#export TACHYON_UNDERFS_ADDRESS=hdfs://localhost:9000
export TACHYON_WORKER_MEMORY_SIZE=1GB
export TACHYON_UNDERFS_HDFS_IMPL=org.apache.hadoop.hdfs.DistributedFileSystem
export TACHYON_WORKER_MAX_WORKER_THREADS=2048
export TACHYON_MASTER_MAX_WORKER_THREADS=2048
```

2.2.2 格式化 Tachyon

完成配置后即可以单机模式启动 Tachyon，启动前需要格式化存储文件，格式化和启动 Tachyon 的命令分别为：

$cd /app/hadoop/tachyon-0.5.0/bin

$./tachyon format

```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 ~]$ cd /app/hadoop/tachyon-0.5.0/bin
[hadoop@hadoop1 bin]$ ./tachyon format
Connection to localhost... Formatting Tachyon Worker @ hadoop1
Removing local data under folder: /mnt/ramdisk/tachyonworker/
Connection to localhost closed.
Formatting Tachyon Master @ localhost
Formatting JOURNAL_FOLDER: /app/hadoop/tachyon-0.5.0/libexec/../journal/
Formatting UNDERFS_DATA_FOLDER: /app/hadoop/tachyon-0.5.0/libexec/../underfs/tmp/tachyon/da
ta
Formatting UNDERFS_WORKERS_FOLDER: /app/hadoop/tachyon-0.5.0/libexec/../underfs/tmp/tachyon
/workers
```

存储文件为$TACHYON_HOME/underfs/tmp/tachyon 目录下

2.2.3 启动 Tachyon

使用如下命令启动 Tachyon，可以看到在/nmt/ramdisk 目录下格式化 RamFS

$cd /app/hadoop/tachyon-0.5.0/bin

$./tachyon-start.sh local

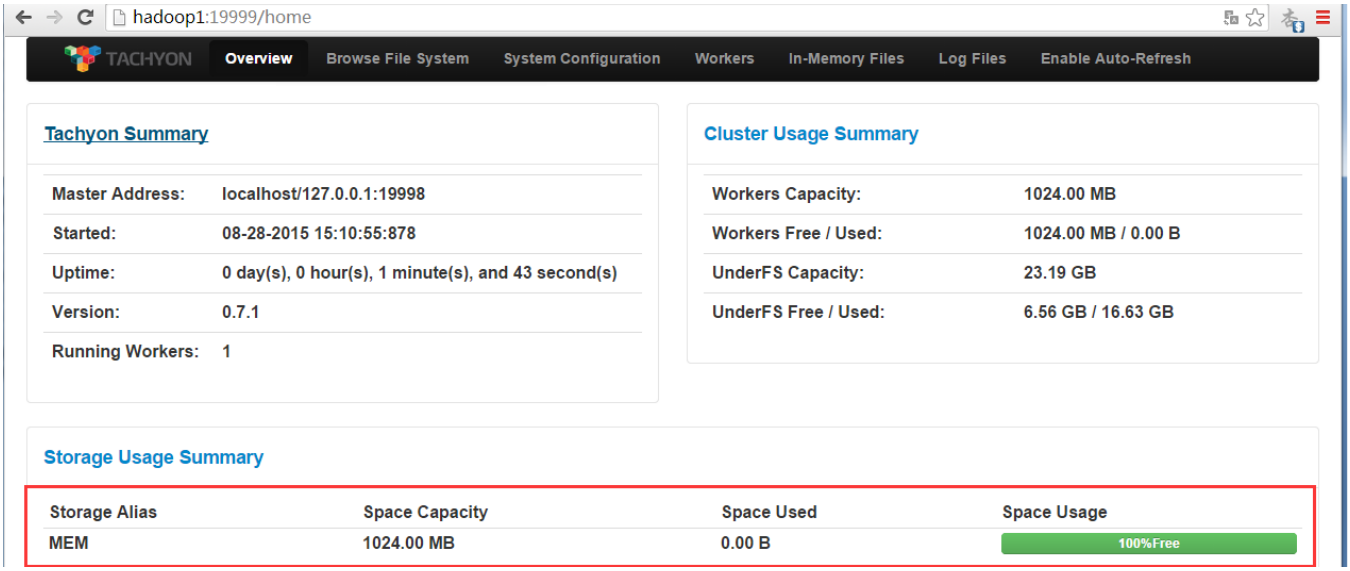
```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 ~]$ cd /app/hadoop/tachyon-0.5.0/bin
[hadoop@hadoop1 bin]$ ./tachyon-start.sh local
TACHYON_LOGS_DIR: /app/hadoop/tachyon-0.5.0/libexec/../logs
Killed 0 processes
Killed 0 processes
Connection to localhost... Killed 0 processes
Connection to localhost closed.
Formatting RamFS: /mnt/ramdisk (1gb)
Starting master @ localhost
Starting worker @ hadoop1
[hadoop@hadoop1 bin]$
```

2.2.4 验证启动

使用 JPS 命令查看 Tachyon 进程，分别为：TachyonWorker 和 TachyonMaster

```
hadoop1
[hadoop@hadoop1 ~]$ jps
6846 TachyonWorker
6821 TachyonMaster
6969 Jps
[hadoop@hadoop1 ~]$
```

查看 Tachyon 监控页面，访问地址为 http://hadoop1:19999

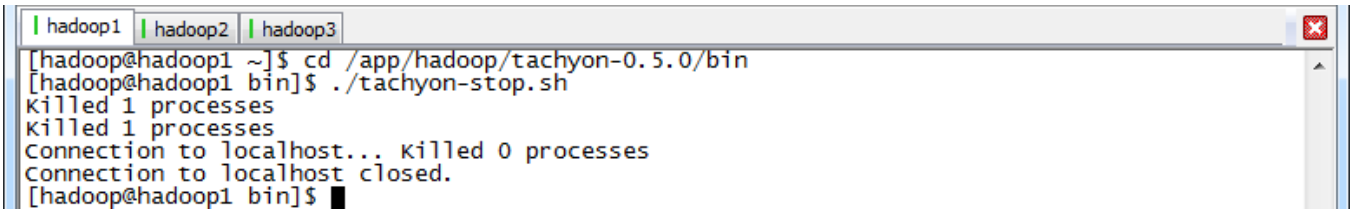


2.2.5 停止 Tachyon

停止 Tachyon 的命令为：

$cd /app/hadoop/tachyon-0.5.0/bin

$./tachyon-stop.sh



2.3 集群模式部署 Tachyon

2.3.1 集群环境

集群包含三个节点（该集群环境可以参考第二课《2.Spark 编译与部署（上）--基础环境搭建》进行搭建），运行进程分布如下：

序号	IP 地址	机器名	运行进程	核数/内存	用户名	目录
1	192.168.0.61	hadoop1	TachyonMaster TachyonWorker	1 核/3G	hadoop	/app /app/hadoop /app/hadoop/Tach...
2	192.168.0.62	hadoop2	TachyonWorker	1 核/2G	hadoop	
3	192.168.0.63	hadoop3	TachyonWorker	1 核/2G	hadoop	

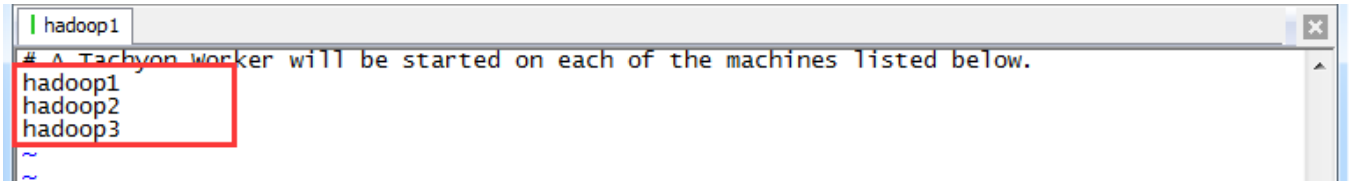
2.3.2 配置 conf/worker

Tachyon 相关配置文件在$TACHYON_HOME/conf 目录下，对 slaves 文件中配置需要启动

TachyonWorker 的节点，在这里需要设置 hadoop1、hadoop2 和 hadoop3 三个节点：

$cd /app/hadoop/tachyon-0.5.0/conf

$vi slaves



2.3.3 配置 conf/tachyon-env.sh

在$TACHYON_HOME/conf 目录下，将 tachyon-env.sh.template 复制为 tachyon-env.sh，并在 achyon-env.sh 中修改具体配置。不同于单机模式，这里需要修改 TachyonMaster 地址以及底层文件系统路径：

$cd /app/hadoop/tachyon-0.5.0/conf

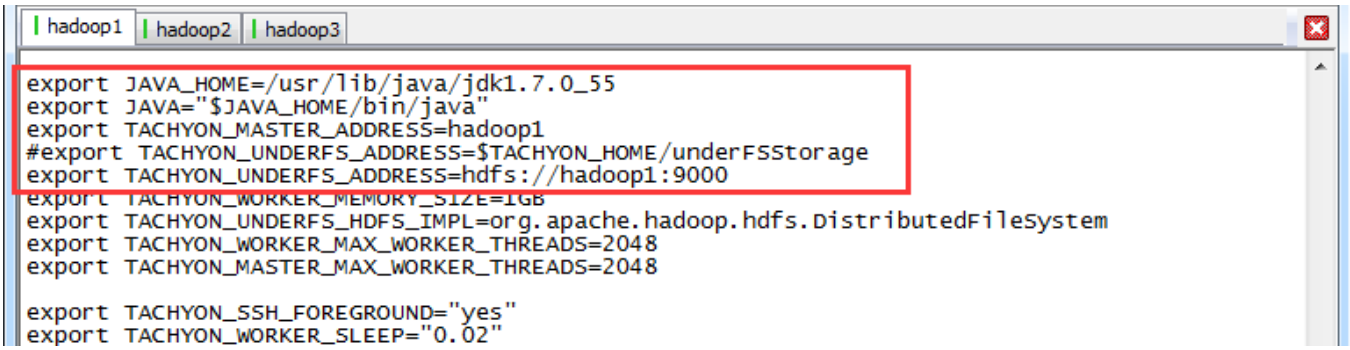
$cp tachyon-env.sh.template tachyon-env.sh

$vi tachyon-env.sh

在该文件中修改一下两个参数，这里使用底层文件系统为 HDFS：

export TACHYON_MASTER_ADDRESS=hadoop1

export TACHYON_UNDERFS_ADDRESS=hdfs://hadoop1:9000



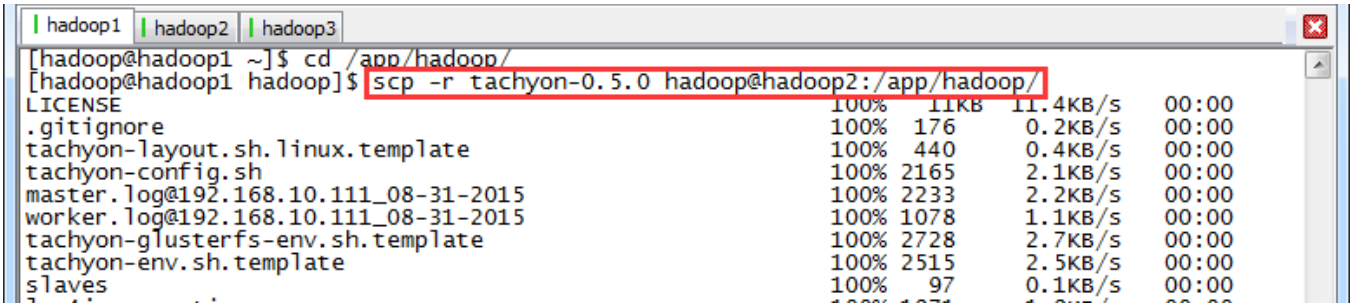
2.3.4 向各个节点分发 Tachyon

使用如下命令把 hadoop 文件夹复制到 hadoop2 和 hadoop3 机器

$cd /app/hadoop/

$scp -r tachyon-0.5.0 hadoop@hadoop2:/app/hadoop/

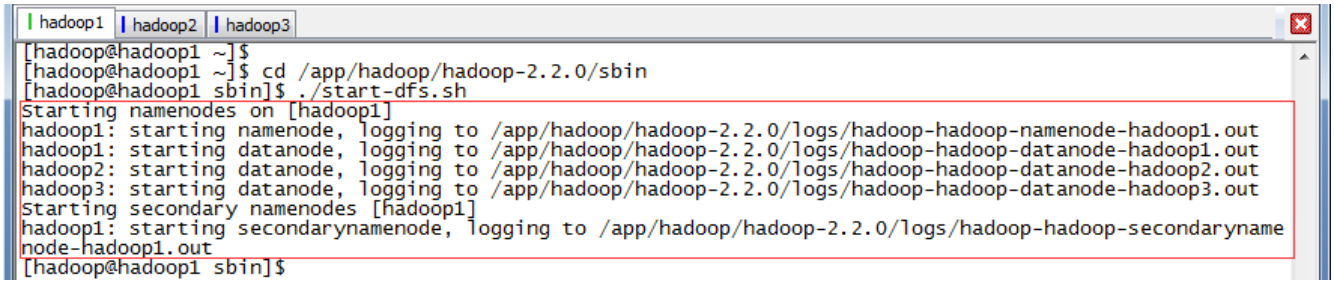
$scp -r tachyon-0.5.0 hadoop@hadoop3:/app/hadoop/



2.3.5 启动 HDFS

$cd /app/hadoop/hadoop-2.2.0/sbin

$./start-dfs.sh

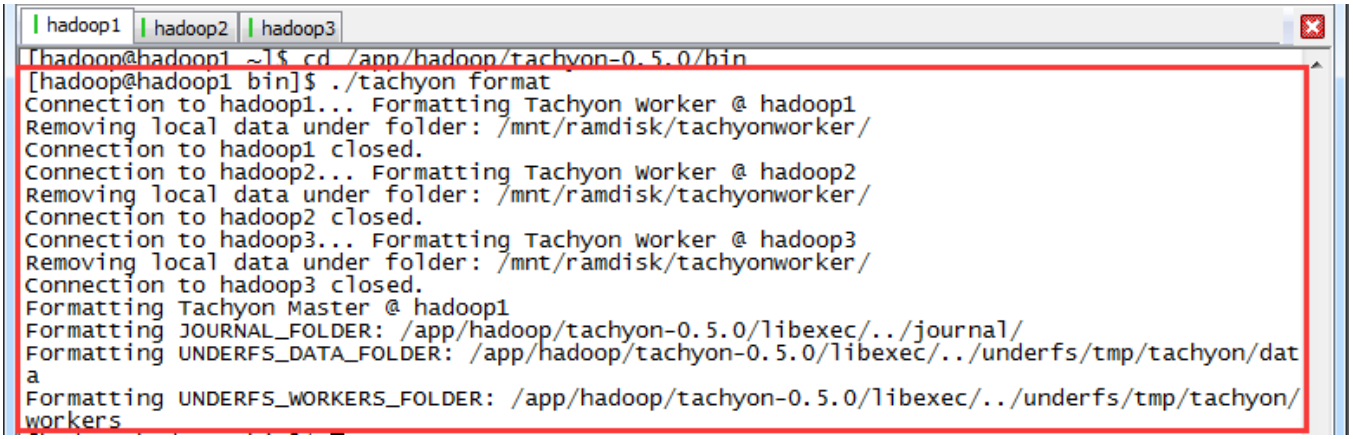


2.3.6 格式化 Tachyon

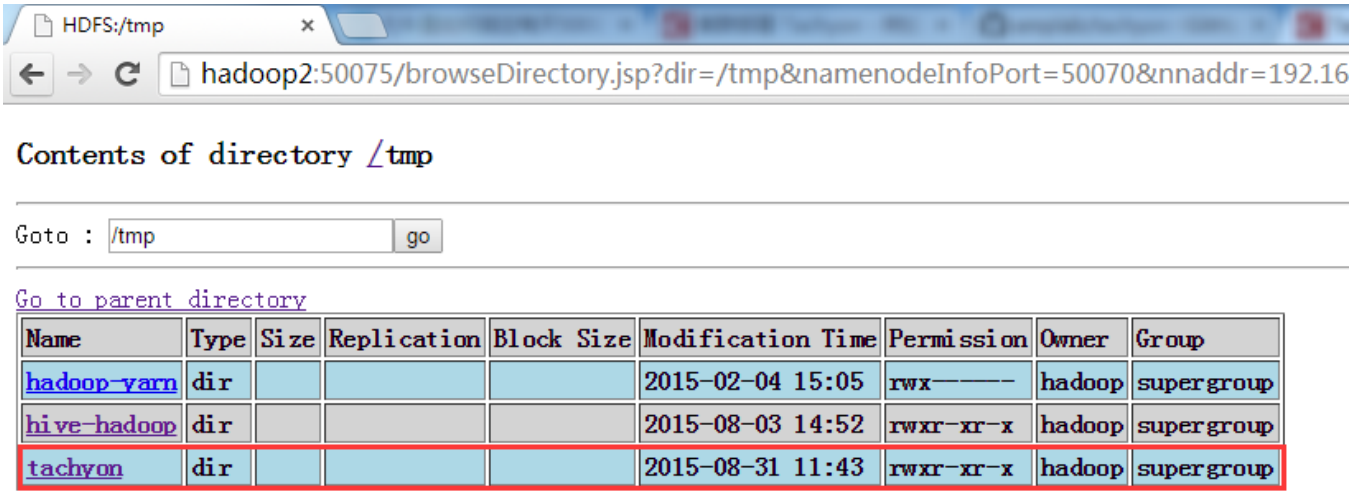
启动前需要格式化存储文件，格式化命令为：

$cd /app/hadoop/tachyon-0.5.0/bin

$./tachyon format



可以看到在 HDFS 的/tmp 创建了 tachyon 文件夹

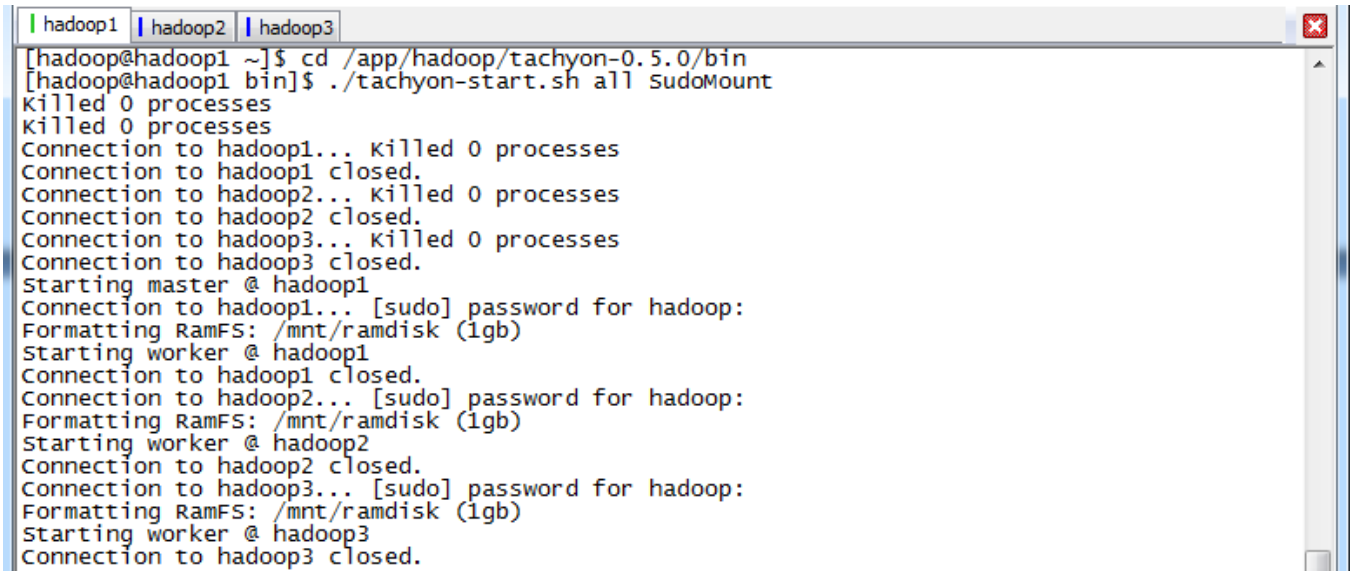


2.3.7 启动 Tachyon

在这里使用 SudoMout 参数，需要在启动过程中输入 hadoop 的密码，具体过程如下：

$cd /app/hadoop/tachyon-0.5.0/bin

$./tachyon-start.sh all SudoMount



启动 Tachyon 有了更多的选项：

- **./tachyon-start.sh all Mount** 在启动前自动挂载 TachyonWorker 所使用的 RamFS，然后启动 TachyonMaster 和所有 TachyonWorker。由于直接使用 mount 命令，所以需要用户为 root；
- **./tachyon-start.sh all SudoMount** 在启动前自动挂载 TachyonWorker 所使用的 RamFS，然后启动 TachyonMaster 和所有 TachyonWorker。由于使用 sudo mount 命令，所以需要用户有 sudo 权限；
- **./tachyon-start.sh all NoMount** 认为 RamFS 已经挂载好，不执行挂载操作，只启动 TachyonMaster 和所有 TachyonWorker

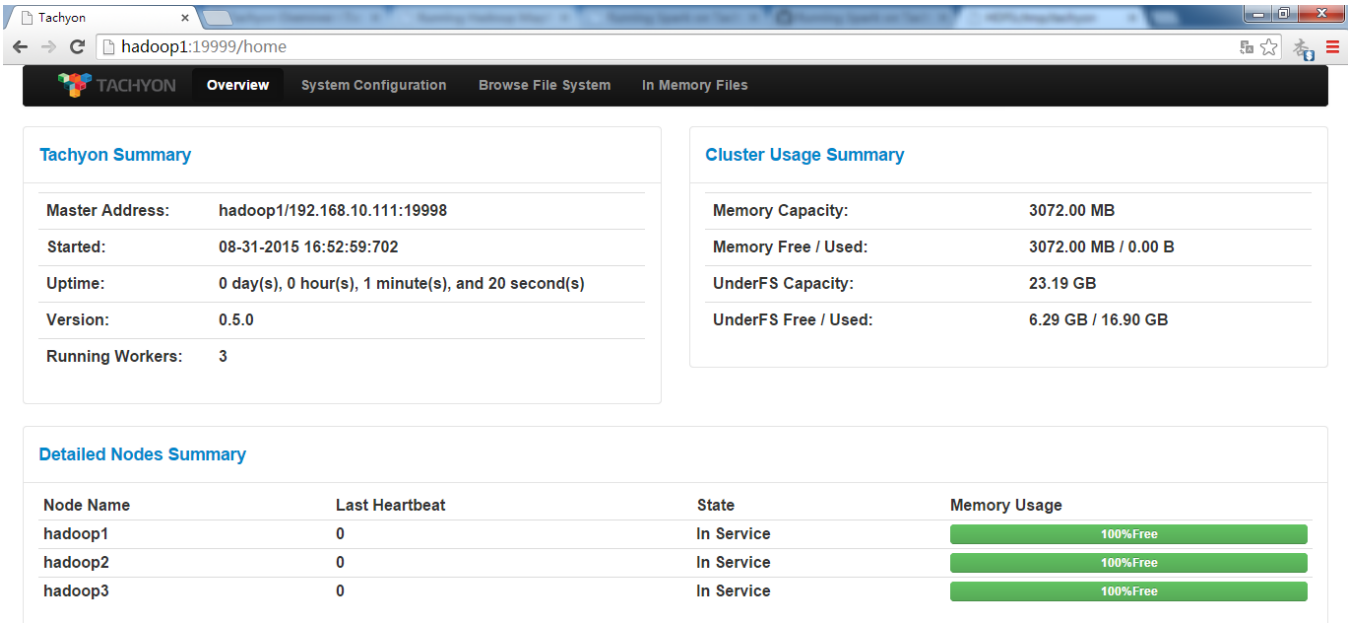
因此，如果不想每次启动 Tachyon 都挂载一次 RamFS，可以先使用命令*./tachyon-mount.sh Mount workers* 或*./tachyon-mount.sh SudoMount workers* 挂载好所有 RamFS，然后使用*./tachyon-start.sh all NoMount* 命令启动 Tachyon。

单机和集群式模式的区别就在于节点配置和启动步骤，事实上，也可以在集群模式下只设置一个 TachyonWorker，此时就成为伪分布模式。

2.3.8 验证启动

使用 JPS 命令查看 Tachyon 进程，分别为：TachyonWorker 和 TachyonMaster

可以在浏览器内打开 Tachyon 的 WebUI，如 http://hadoop1:19999，查看整个 Tachyon 的状态，各个 TachyonWorker 的运行情况、各项配置信息和浏览文件系统等。



$cd /app/hadoop/tachyon-0.5.0/bin
$./tachyon runTests

2.4 Tachyon 的配置

这里以 0.5.0 版本为例，介绍 Tachyon 中可配置参数的具体含义。Tachyon 中的可配置项分为两类，一种是系统环境变量，用于在不同脚本间共享配置信息；另一种是程序运行参数，通过 -D 选项传入运行 Tachyon 的 JVM 中。程序运行参数又分为：

- 通用配置（Common Configuration）
- TachyonMaster 配置（Master Configuration）
- TachyonWorker 配置（Worker Configuration）
- 用户配置（User Configuration）

要修改或添加这些可配置项，可修改 conf/tachyon-env.sh 文件。

2.4.1 Tachyon 环境变量

配置项	配置项说明
JAVA_HOME	系统中 JDK 的安装路径
TACHYON_RAM_FOLDER	配置 ramfs 挂载的文件目录，默认为/mnt/ramdisk
TACHYON_MASTER_ADDRESS	启动 TachyonMaster 的地址，默认为 localhost，所以在单机模式下不用更改
TACHYON_UNDERFS_ADDRESS	Tachyon 使用的底层文件系统的路径，本地文件系统（单机模式下），如"/tmp/tachyon"，或 HDFS，如"hdfs://ip:port"
TACHYON_WORKER_MEMORY_SIZE	每个 TachyonWorker 使用的 RamFS 大小，默认为 1GB

2.4.2 Tachyon 通用配置

配置项	配置项说明
tachyon.underfs.address	Tachyon 在底层文件系统的的路径，默认为 $TACHYON_UNDERFS_ADDRESS
tachyon.home	Tachyon 的安装路径，启动 Tachyon 时为当前 tachyon 文件夹的路径
tachyon.data.folder	Tachyon 数据在底层文件系统的存放路径，默认为 $TACHYON_UNDERFS_ADDRESS/tmp/tachyon/data
tachyon.workers.folder	TachyonWorkers 在底层文件系统的工作路径，默认为 $TACHYON_UNDERFS_ADDRESS/tmp/tachyon/workers
tachyon.usezookeeper	TachyonMaster 是否使用 ZooKeeper 容错，默认为 false
tachyon.zookeeper.adress	如果启用，ZooKeeper 的地址
tachyon.zookeeper.election.path	如果启用，Zookeeper 的 election 文件夹路径，默认为/election
tachyon.zookeeper.leader.path	如果启用，Zookeeper 的 leader 文件夹路径，默认为/leader

tachyon.underfs.hdfs.impl	实现HDFS的类，默认org.apache.hadoop.hdfs,DistributedFileSystem
tachyon.max.columns	Tachyon中RawTable允许的最大列数，默认为1000
tachyon.table.metadata.byte	Tachyon中RawTable元数据允许存储的最大字节数，默认为5242880，即5MB
tachyon.underfs.glusterfs.impl	如果使用GlusterFS为底层文件系统，实现GlusterFS的类，默认为org.apache.hadoop.fs.glusterfs.GlusterFileSystem
tachyon.underfs.glusterfs.mounts	如果使用GlusterFS为底层文件系统，GlusterFS卷的挂载目录
tachyon.underfs.glusterfs.volumes	如果使用GlusterFS为底层文件系统，GlusterFS的卷名
tachyon.underfs.glusterfs.mapred.system.dir	如果使用GlusterFS为底层文件系统，GlusterFS用于存放MapReduce中间数据的可选子目录，默认为glusterfs:///mapred/system
tachyon.web.resources	Tachyon WebUI可用的资源，默认为$tachyon.home/core/src/main/webapp
tachyon.async.enabled	是否启用异步模式，默认为false
tachyon.underfs.hadoop.prefixes	底层使用hadoop文件系统的前缀列表，默认为"hdfs://"，"s3://"，"s3n://"，"glusterfs:///"
tachyon.test.mode	是否启用测试模式，默认为false
tachyon.master.retry	连接重试次数，默认为29

2.4.3 TachyonMaster 配置

配置项	配置项说明
tachyon.master.worker.timeout.ms	TachyonMaster和TachyonWorker心跳包失效时长，默认为60000ms
tachyon.master.journal.folder	TachyonMaster的journal日志存放路径，默认为$TACHYON_HOME/journal/
tachyon.master.hostname	TachyonMaster的主机名
tachyon.master.port	TachyonMaster的远程调用通讯端口，默认为19998
tachyon.master.web.port	TachyonMaster的WebUI端口，默认为19999

tachyon.master.web.threads	TachyonMaster 的 WebUI 线程数，默认为 9
tachyon.master.whitelist	可缓存的路径前缀列表，列表以逗号隔开，表示该路径下的文件能够被缓存至内存，默认为/，即根目录
tachyon.master.temporary.folder	TachyonMaster 的临时文件夹，默认为/tmp
tachyon.master.heartbeat.interval.ms	TachyonMaster 心跳包间隔时间，默认为 1000ms
tachyon.master.selector.threads	TachyonMaster 的 thrift 监听线程数，默认为 3
tachyon.master.queue.size.per.selector	TachyonMaster 的 thrift 消息队列长度，默认为 3000
tachyon.master.server.threads	TachyonMaster 节点的 thrift 服务线程数 默认为 CPU 核数的 2 倍
tachyon.master.pinlist	常驻内存的文件列表，以逗号隔开，表示该路径下的文件不会从内存中剔除，默认为 null

2.4.4 TachyonWorker 配置

配置项	配置项说明
tachyon.worker.data.folder	TachyonWorker 在 RamFS 中的工作路径，默认为 $TACHYON_RAM_FOLDER/tachyonworker/
tachyon.work.port	TachyonWorker 的远程调用通讯端口，默认为 29998
tachyon.worker.data.port	TachyonWorker 的数据传输服务的端口，默认为 29999
tachyon.worker.memory.size	TachyonWorker 所使用的 RamFS 大小，默认为 $TACHYON_WORKER_MEMORY_SIZE
tachyon.worker.heartbeat.timeout.ms	TachyonWorker 心跳包失效的时长，默认为 10000ms
tachyon.worker.to.master.heartbeat.interval.ms	TachyonWorker 向 TachyonMaster 发送心跳包的时间间隔，默认为 1000ms
tachyon.worker.selector.threads	TachyonWorker 的 thrift 监听线程数，默认为 3
tachyon.worker.queue.size.per.selector	TachyonWorker 的 thrift 消息队列长度，默认为 3000

tachyon.worker.server.threads	TachyonWorker 的 thrift 服务线程数，默认为 CPU 核数
tachyon.worker.user.timeout.ms	TachyonWorker 和用户之间心跳包失效时长，默认为 10000ms
tachyon.worker.checkpoint.threads	TachyonWorker 的 checkpoint 线程数，默认为 1
tachyon.worker.per.thread.checkpoint.cap.mb.sec	TachyonWorker 的 checkpoint 的速度，默认为 1000MB/s
tachyon.worker.network.type	TachyonWorker 在传输文件数据时使用的传输方式，默认为 NETTY，可选为 NIO 或 NETTY

2.4.5 用户配置

配置项	配置项说明
tachyon.user.failed.space.request.limits	用户向文件系统请求空间失败时的最大重试次数,默认为 3
tachyon.user.quota.unit.bytes	客用户一次向 TachyonWorker 请求的最少字节数，默认为 8388608，即 8MB
tachyon.user.file.buffer.byte	用户读写文件时的缓存区大小，默认为 1048576，即 1MB
tachyon.user.default.block.size.byte	用户创建文件时的默认块大小，默认为 1073741824，即 1GB
tachyon.user.remote.read.buffer.size.byte	用户读远程文件时的缓冲区大小，默认为 1048576，即 1MB
tachyon.user.heartbeat.interval.ms	用户心跳包时间间隔，默认为 1000ms
tachyon.user.file.writetype.default	用户在使用 tachyon.hadoop.TFS 时的默认写类型，默认为 CACHE_THROUGH

3 Tachyon 命令行使用

Tachyon 的命令行界面让用户可以对文件系统进行基本的操作。调用命令行工具使用以下脚本：

$./tachyon tfs

文件系统访问的路径格式如下：

tachyon://<master node address>:<master node port>/<path>

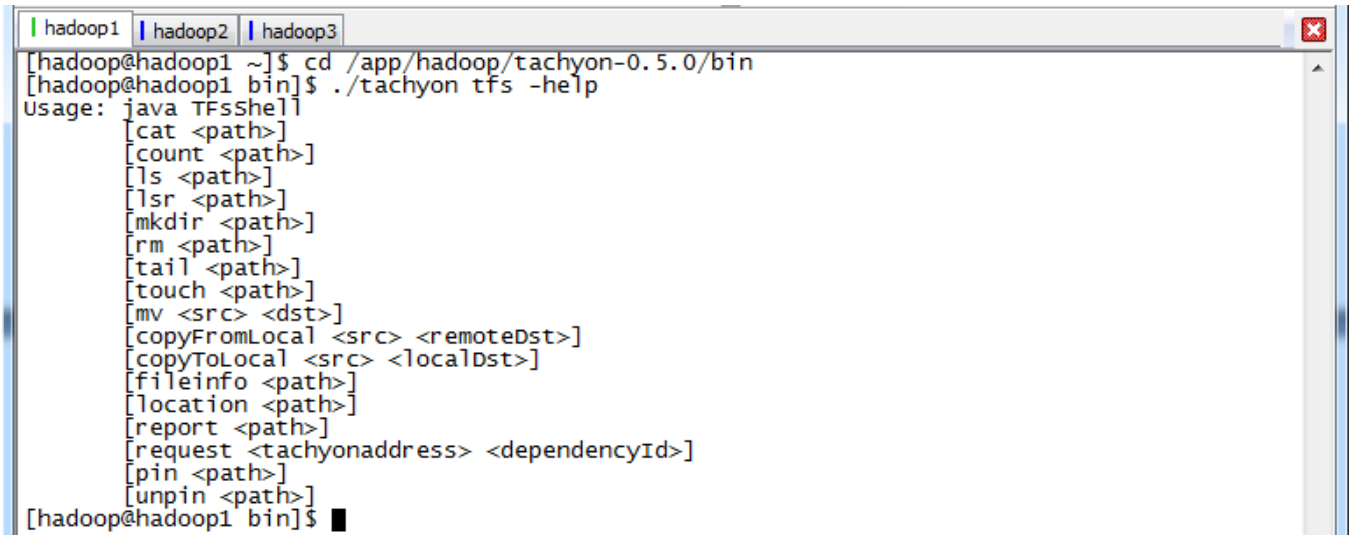
在 Tachyon 命令行使用中 tachyon://<master node address>:<master node port>前缀可以省略，该信息从配置文件中读取。

3.1 接口说明

可以通过如下命令查看 Tachyon 所有接口命令

$cd /app/hadoop/tachyon-0.5.0/bin

$./tachyon tfs -help



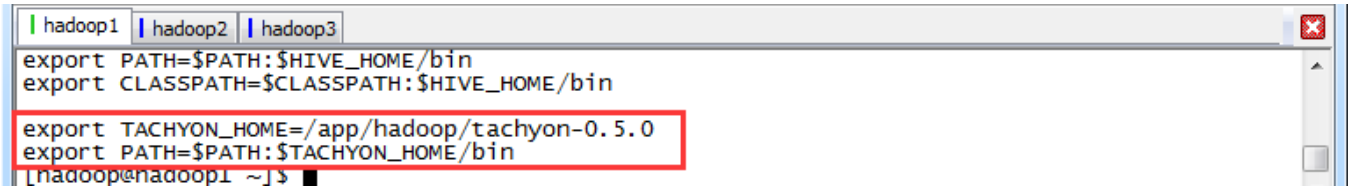
其中大部分的命令含义可以参考 Linux 下同名命令，命令含义：

命令	含义
cat	将文件内容输出到控制台
count	显示匹配指定的前缀“路径”的文件夹和文件的数量。
ls	列出指定路径下所有的文件和目录信息，如大小等。
lsr	递归地列出指定路径下所有的文件和目录信息，如大小等。
mkdir	在给定的路径创建一个目录,以及任何必要的父目录。如果路径已经存在将会失败。
rm	删除一个文件。如果是一个目录的路径将会失败。
rmr（0.5.0 版本不包含）	删除一个文件或目录，以及该目录下的所有文件夹和文件
tail	输出指定文件的最后 1 kb 到控制台。
touch	在指定的路径创建一个 0 字节的文件。
mv	移动指定的源文件或源目录到一个目的路径。如果目的路径已经存在将会失败。

copyFromLocal	将本地指定的路径复制到 Tachyon 中指定的路径。如果 Tachyon 中指定的路径已经存在将会失败。
copyToLocal	从 Tachyon 中指定的路径复制本地指定的路径。
fileinfo	输出指定文件的块信息。
location	输出存放指定文件的所在节点列表信息。
report	向 master 报告文件丢失
request	根据指定的 dependency ID，请求文件。
pin	将指定的路径常驻在内存中。如果指定的是一个文件夹，会递归地包含所有文件以及任何在这个文件夹中新创建的文件。
unpin	撤销指定路径的常驻内存状态。如果指定的是一个文件夹，会递归地包含所有文件以及任何在这个文件夹中新创建的文件。
Free（0.5.0 版本不包含）	释放一个文件或一个文件夹下的所有文件的内存。文件/文件夹在 underfs 仍然是可用的。

3.2 接口操作示例

在操作之前需要把$TACHYON_HOME/bin 配置到/etc/profile 配置文件的 PATH 中，并通过 *source /etc/profile* 生效



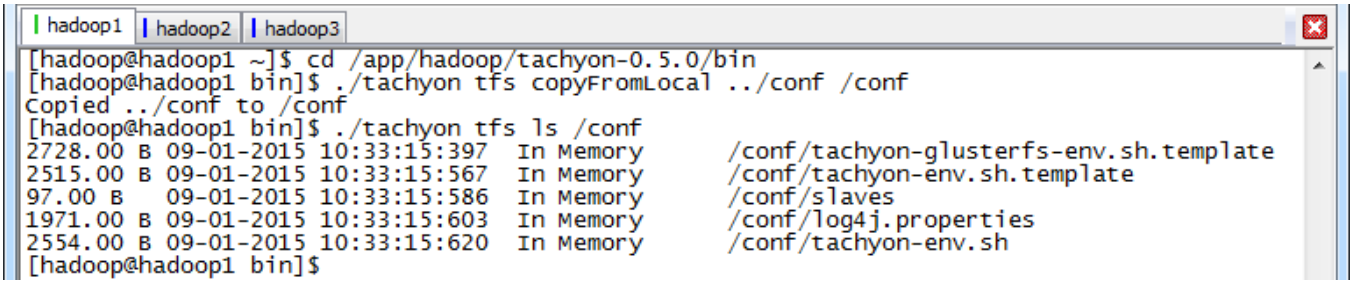
3.2.1 copyFromLocal

将本地$TACHYON_HOME/conf 目录拷贝到 Tachyon 文件系统的根目录下的 conf 子目录

$cd /app/hadoop/tachyon-0.5.0/bin

$./tachyon tfs copyFromLocal ../conf /conf

$./tachyon tfs ls /conf



3.2.2 copyToLocal

把 Tachyon 文件系统文件复制到本地，需要注意的是命令中的 src 必须是 Tachyon 文件系统中的文件不支持目录拷贝，否则报错无法复制

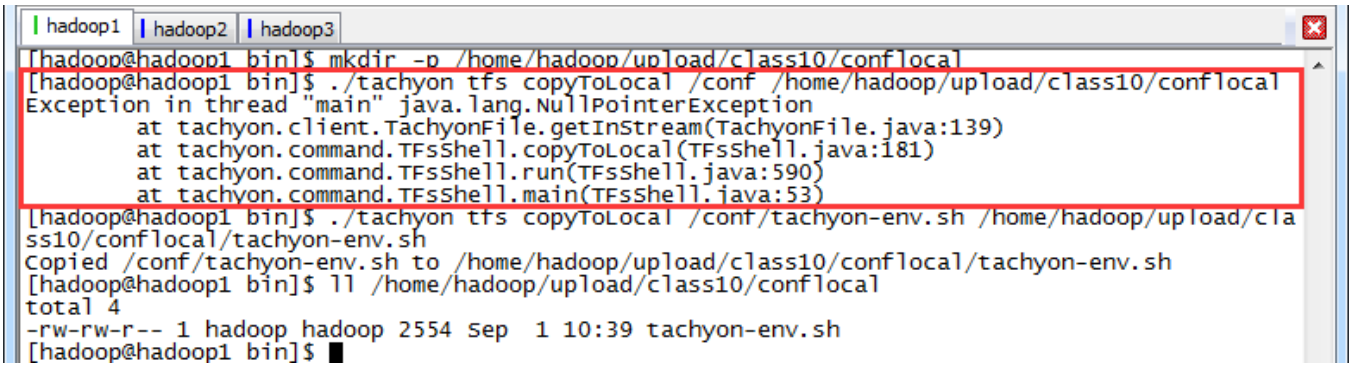
$mkdir -p /home/hadoop/upload/class10/conflocal

$./tachyon tfs copyToLocal /conf /home/hadoop/upload/class10/conflocal

$./tachyon tfs copyToLocal /conf/tachyon-env.sh

/home/hadoop/upload/class10/conflocal/tachyon-env.sh

$ll /home/hadoop/upload/class10/conflocal



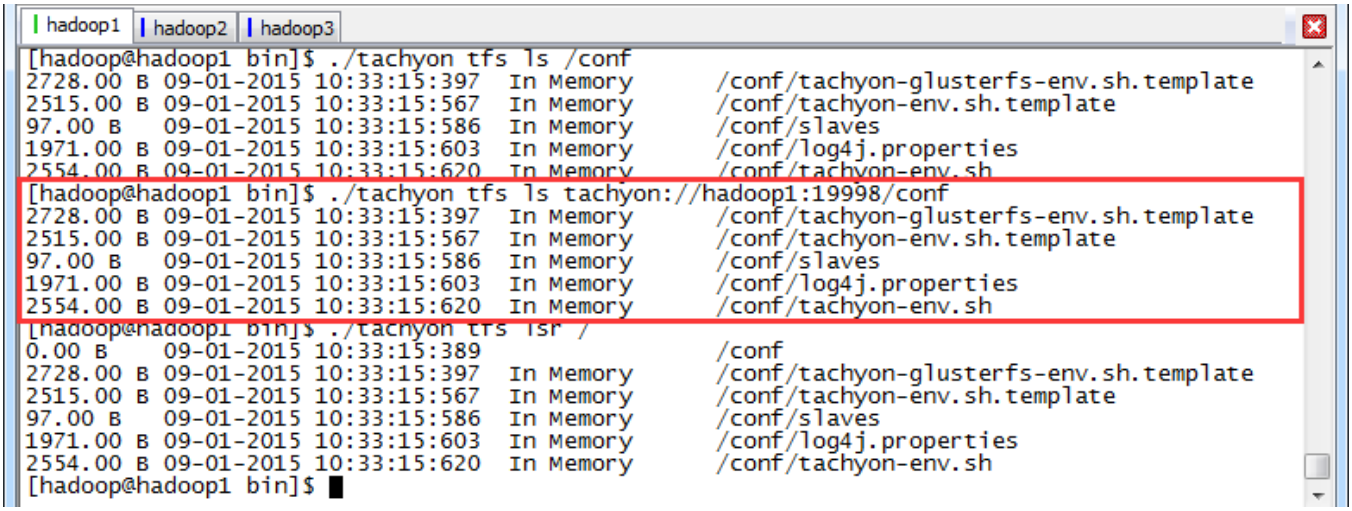
3.2.3 ls 和 lsr

使用 ls 和 lsr 命令查看 Tachyon 文件系统下的文件信息，其中 lsr 命令可以递归地查看子目录。

$./tachyon tfs ls /conf

$./tachyon tfs ls tachyon://hadoop1:19998/conf

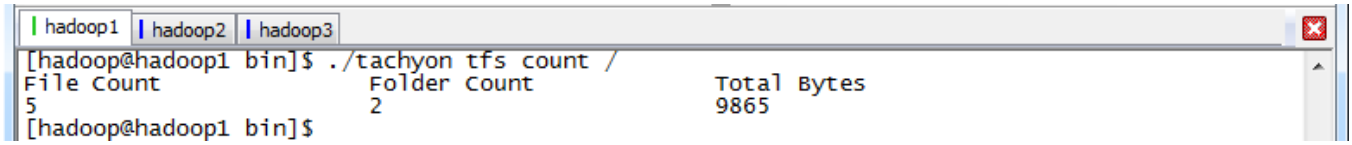
$./tachyon tfs lsr /



3.2.4 count

统计当前路径下的目录、文件信息，包括文件数、目录树以及总的大小

$./tachyon tfs count /

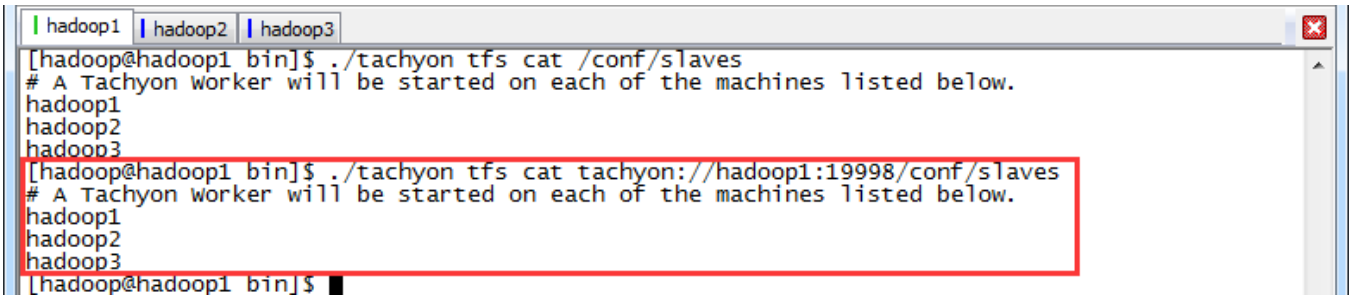


3.2.5 cat

查看指定文件的内容

$./tachyon tfs cat /conf/slaves

$./tachyon tfs cat tachyon://hadoop1:19998/conf/slaves



3.2.6 mkdir、rm、rmr 和 touch

（1） **mkdir**：创建目录，支持自动创建不存在的父目录；

（2） **rm**：删除文件，不能删除目录，注意，递归删除根目录是无效的

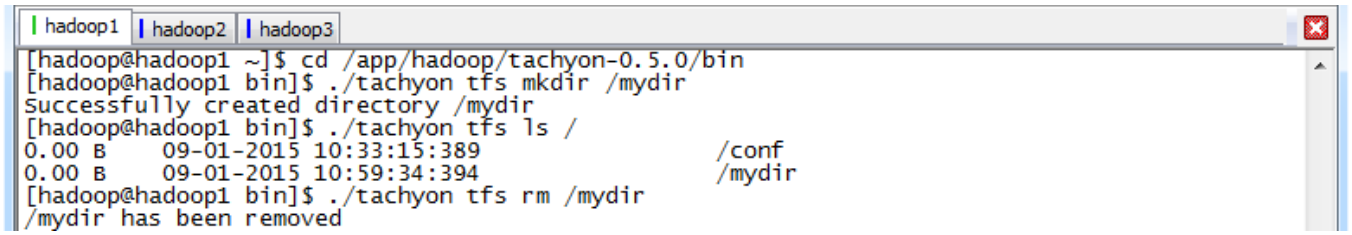
（3） **rmr**：删除目录，支持递归，包含子目录和文件，其中 0.5.0 版本不提供该命令

（4） **touch**：创建文件，不能创建已经存在的文件。

$./tachyon tfs mkdir /mydir

$./tachyon tfs ls /

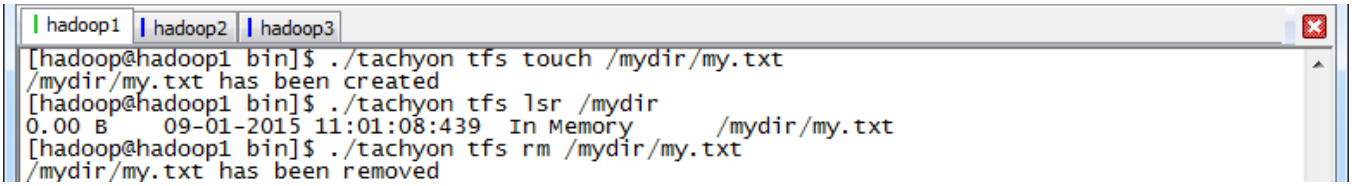
$./tachyon tfs rm /mydir



$./tachyon tfs touch /mydir/my.txt

$./tachyon tfs lsr /mydir

$./tachyon tfs rm /mydir/my.txt



$./tachyon tfs touch /mydir2/2/2/my.txt

$./tachyon tfs lsr /mydir2

$./tachyon tfs rm /mydir2

$./tachyon tfs rm /

$./tachyon tfs ls /

```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 bin]$ ./tachyon tfs touch /mydir2/2/2/my.txt
/mydir2/2/2/my.txt has been created
[hadoop@hadoop1 bin]$ ./tachyon tfs lsr /mydir2
0.00 B     09-01-2015 11:03:56:543                  /mydir2/2
0.00 B     09-01-2015 11:03:56:543                  /mydir2/2/2
0.00 B     09-01-2015 11:03:56:543  In Memory       /mydir2/2/2/my.txt
[hadoop@hadoop1 bin]$ ./tachyon tfs rm /mydir2
/mydir2 has been removed
[hadoop@hadoop1 bin]$ ./tachyon tfs rm /
[hadoop@hadoop1 bin]$ ./tachyon tfs ls /
0.00 B     09-01-2015 10:33:15:389                  /conf
0.00 B     09-01-2015 11:00:11:478                  /mydir
```

3.2.7 pin 和 unpin

pin 命令将指定的路径常驻在内存中，如果指定的是一个文件夹会递归地包含所有文件以及任何在这个文件夹中新创建的文件。unpin 命令撤销指定路径的常驻内存状态。

```
hadoop1 | hadoop2 | hadoop3
[hadoop@hadoop1 bin]$ ./tachyon tfs pin /conf/log4j.properties
File '/conf/log4j.properties' was successfully pinned.
[hadoop@hadoop1 bin]$ ./tachyon tfs unpin /conf/log4j.properties
File '/conf/log4j.properties' was successfully unpinned.
[hadoop@hadoop1 bin]$
```

pin 执行前或 unpin 执行后的 Web Interface 界面

$./tachyon tfs pin /conf/log4j.properties

TACHYON | Overview | System Configuration | **Browse File System** | In Memory Files

root / conf

File Name	Size	Block Size	In-Memory	Pin	Creation Time
log4j.properties	1971.00 B	1024.00 MB	100%	YES	09-01-2015 10:33:15:603
slaves	97.00 B	1024.00 MB	100%	NO	09-01-2015 10:33:15:586
tachyon-env.sh	2554.00 B	1024.00 MB	100%	NO	09-01-2015 10:33:15:620
tachyon-env.sh.template	2515.00 B	1024.00 MB	100%	NO	09-01-2015 10:33:15:567
tachyon-glusterfs-env.sh.template	2728.00 B	1024.00 MB	100%	NO	09-01-2015 10:33:15:397

$./tachyon tfs unpin /conf/log4j.properties

TACHYON | Overview | System Configuration | **Browse File System** | In Memory Files

root / conf

File Name	Size	Block Size	In-Memory	Pin	Creation Time
log4j.properties	1971.00 B	1024.00 MB	100%	NO	09-01-2015 10:33:15:603
slaves	97.00 B	1024.00 MB	100%	NO	09-01-2015 10:33:15:586
tachyon-env.sh	2554.00 B	1024.00 MB	100%	NO	09-01-2015 10:33:15:620
tachyon-env.sh.template	2515.00 B	1024.00 MB	100%	NO	09-01-2015 10:33:15:567
tachyon-glusterfs-env.sh.template	2728.00 B	1024.00 MB	100%	NO	09-01-2015 10:33:15:397

4 Tachyon 实战应用

4.1 配置及启动环境

4.1.1 修改 spark-env.sh

修改$SPARK_HOME/conf 目录下 spark-env.sh 文件：

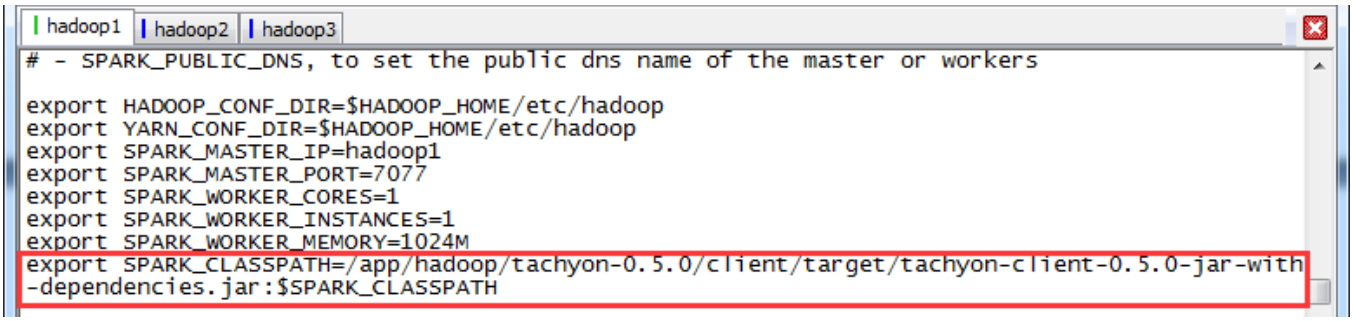
$cd /app/hadoop/spark-1.1.0/conf

$vi spark-env.sh

在该配置文件中添加如下内容：

export

SPARK_CLASSPATH=/app/hadoop/tachyon-0.5.0/client/target/tachyon-client-0.5.0-jar-with-dependencies.jar:$SPARK_CLASSPATH



4.1.2 启动 HDFS

$cd /app/hadoop/hadoop-2.2.0/sbin

$./start-dfs.sh

4.1.3 启动 Tachyon

在这里使用 SudoMout 参数，需要在启动过程中输入 hadoop 的密码，具体过程如下：

$cd /app/hadoop/tachyon-0.5.0/bin

$./tachyon-start.sh all SudoMount

4.2 Tachyon 上运行 Spark

4.2.1 添加 core-site.xml

在 Tachyon 的官方文档说 Hadoop1.X 集群需要添加该配置文件（参见 http://tachyon-project.org/documentation/Running-Spark-on-Tachyon.html），实际在

Hadoop2.2.0 集群测试的过程中发现也需要添加如下配置文件,否则无法识别以 tachyon://开头的文件系统，具体操作是在$SPARK_HOME/conf 目录下创建 core-site.xml 文件

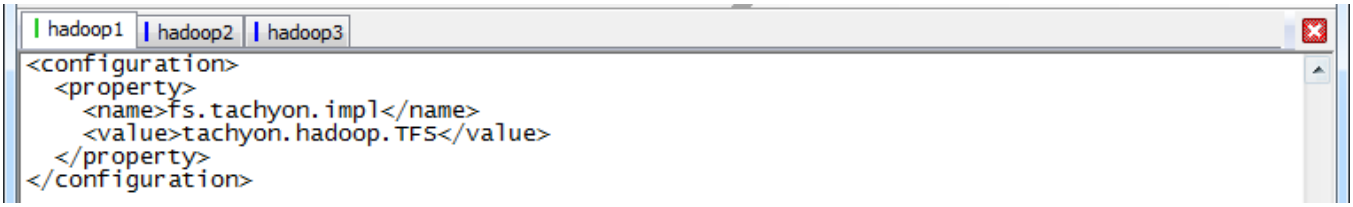
$cd /app/hadoop/spark-1.1.0/conf

$touch core-site.xml

$vi core-site.xml

在该配置文件中添加如下内容：

```
<configuration>
  <property>
    <name>fs.tachyon.impl</name>
    <value>tachyon.hadoop.TFS</value>
  </property>
</configuration>
```



4.2.2 启动 Spark 集群

$cd /app/hadoop/spark-1.1.0/sbin

$./start-all.sh

4.2.3 读取文件并保存

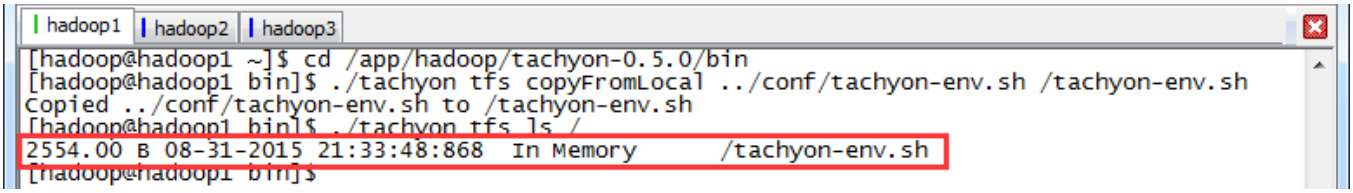
第一步 准备测试数据文件

使用 Tachyon 命令行准备测试数据文件

$cd /app/hadoop/tachyon-0.5.0/bin

$./tachyon tfs copyFromLocal ../conf/tachyon-env.sh /tachyon-env.sh

$./tachyon tfs ls /



第二步 启动 Spark-Shell

$cd /app/hadoop/spark-1.1.0/bin

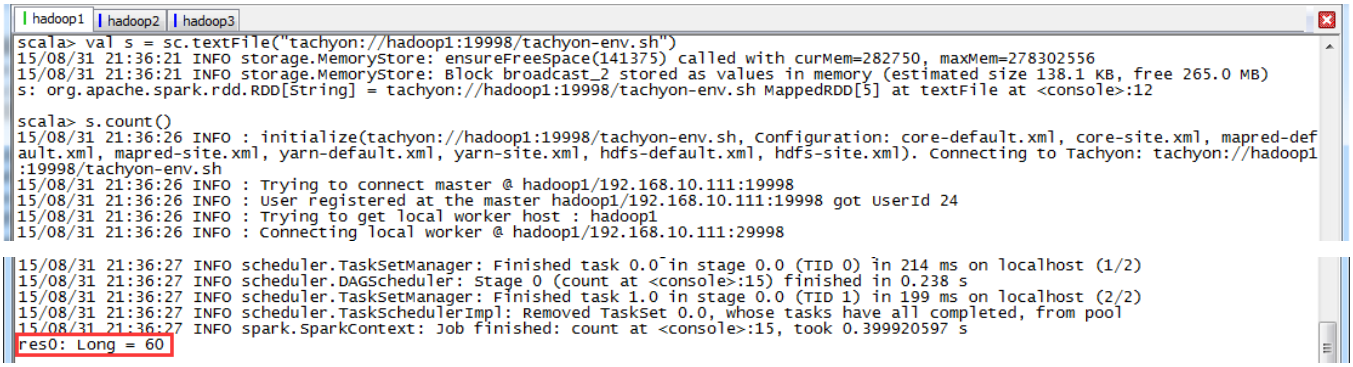
$./spark-shell

第三步 对测试数据文件进行计数并另存

对前面放入到 Tachyon 文件系统的文件进行计数

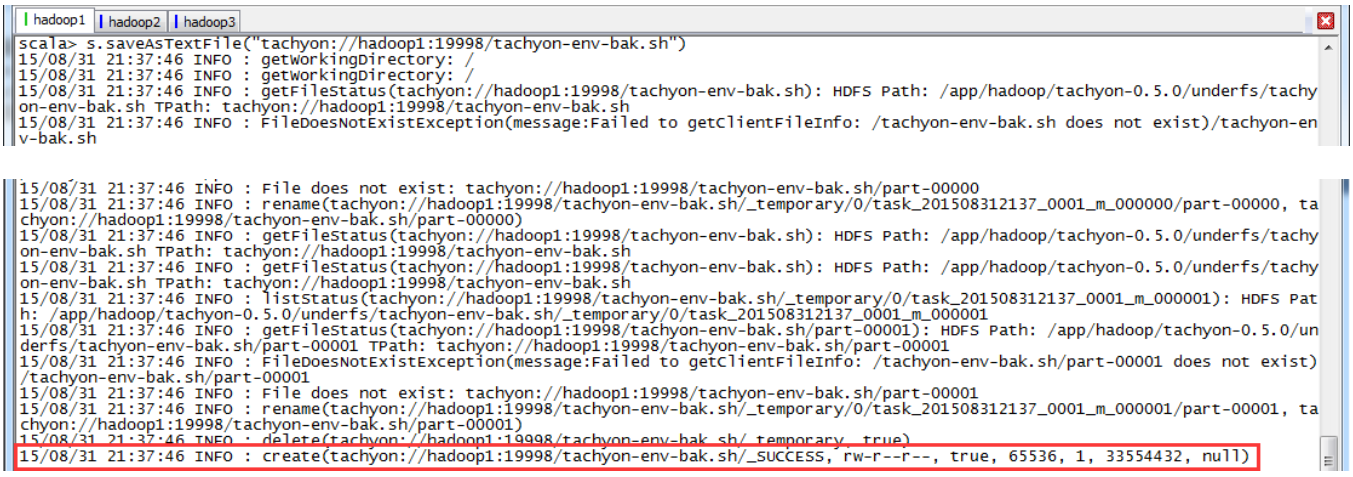
scala>val s = sc.textFile("tachyon://hadoop1:19998/tachyon-env.sh")

scala>s.count()



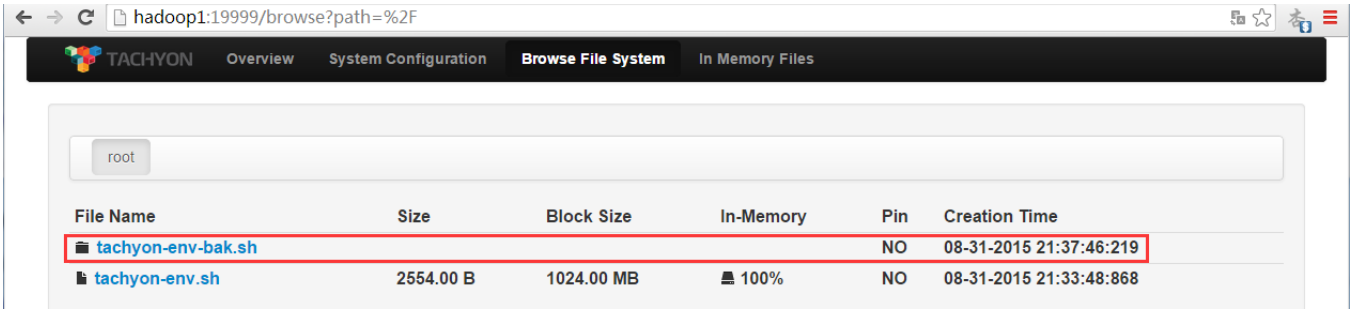
把前面的测试文件另存为 tachyon-env-bak.sh 文件

scala>s.saveAsTextFile("tachyon://hadoop1:19998/tachyon-env-bak.sh")

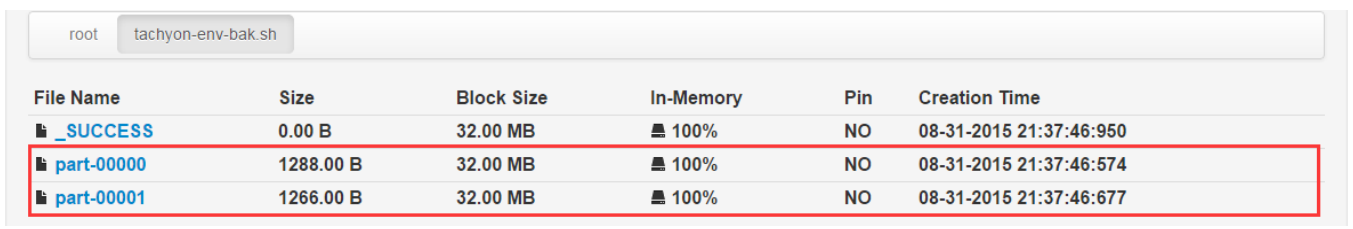


第四步 在 Tachyon 的 UI 界面查看

可以查看到该文件在 Tachyon 文件系统中保存成 tahyon-env-bak.sh 文件夹



该文件夹中包含两个文件，分别为 part-00000 和 part-00001：



其中 tahyon-env-bak.sh/part-0001 文件中内容如下：

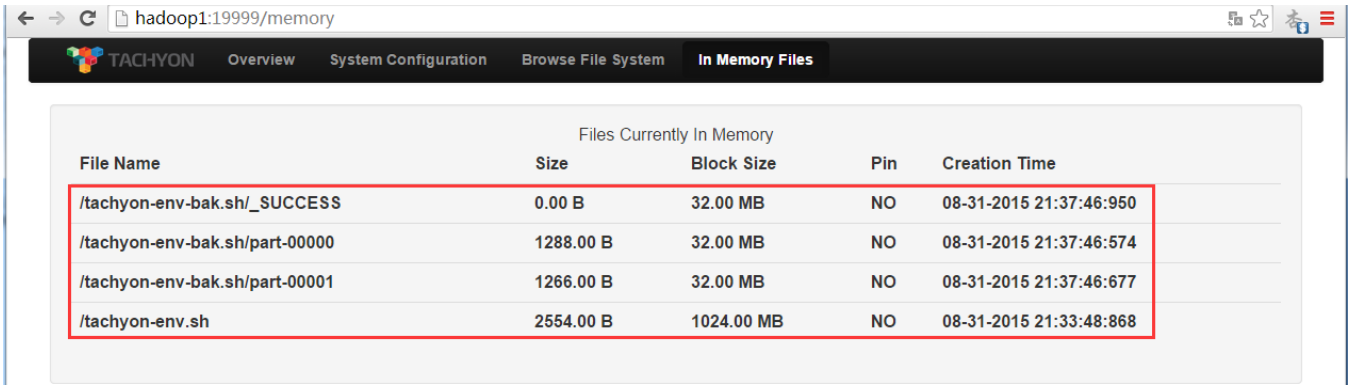
/tachyon-env-bak.sh/part-00001: First 5KB from 0 in ASCII

```
export TACHYON_UNDERFS_ADDRESS=$TACHYON_HOME/underfs
#export TACHYON_UNDERFS_ADDRESS=hdfs://hadoop1:9000
export TACHYON_WORKER_MEMORY_SIZE=1GB
export TACHYON_UNDERFS_HDFS_IMPL=org.apache.hadoop.hdfs.DistributedFileSystem

CONF_DIR="$( cd "$( dirname "${BASH_SOURCE[0]}" )" && pwd )"

export TACHYON_JAVA_OPTS+="
 -Dlog4j.configuration=file:$CONF_DIR/log4j.properties
 -Dtachyon.debug=false
 -Dtachyon.underfs.address=$TACHYON_UNDERFS_ADDRESS
 -Dtachyon.underfs.hdfs.impl=$TACHYON_UNDERFS_HDFS_IMPL
 -Dtachyon.data.folder=$TACHYON_UNDERFS_ADDRESS/tmp/tachyon/data
 -Dtachyon.workers.folder=$TACHYON_UNDERFS_ADDRESS/tmp/tachyon/workers
 -Dtachyon.worker.memory.size=$TACHYON_WORKER_MEMORY_SIZE
```

另外通过内存存在文件的监控页面可以观测到，这几个操作文件在内存中：



4.3 Tachyon 运行 MapReduce

4.3.1 修改 core-site.xml

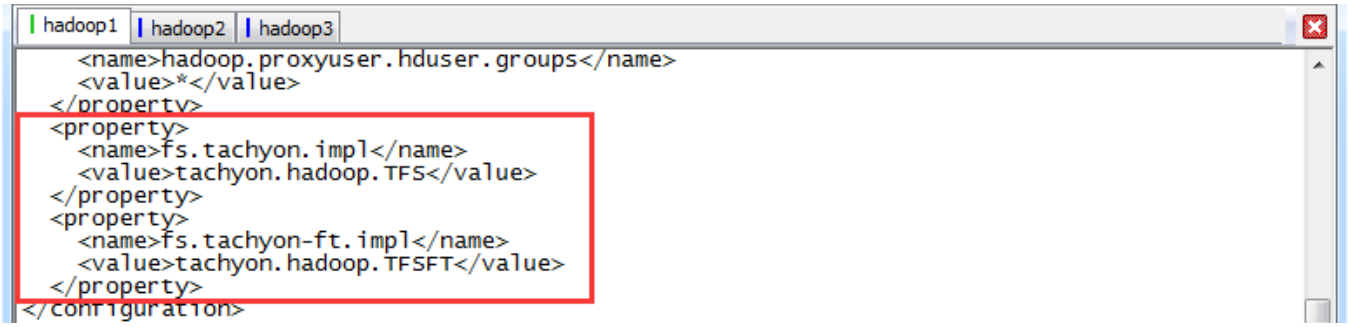
该配置文件为$Hadoop_HOME/conf 目录下的 core-site.xml 文件

$cd /app/hadoop/hadoop-2.2.0/etc/hadoop

$vi core-site.xml

修改 core-site.xml 文件配置，添加如下配置项：

```
<property>
   <name>fs.tachyon.impl</name>
   <value>tachyon.hadoop.TFS</value>
</property>
<property>
   <name>fs.tachyon-ft.impl</name>
   <value>tachyon.hadoop.TFSFT</value>
</property>
```



4.3.2 启动 YARN

$cd /app/hadoop/hadoop-2.2.0/sbin

$./start-yarn.sh

4.3.3 运行 MapReduce 例子

第一步 创建结果保存目录

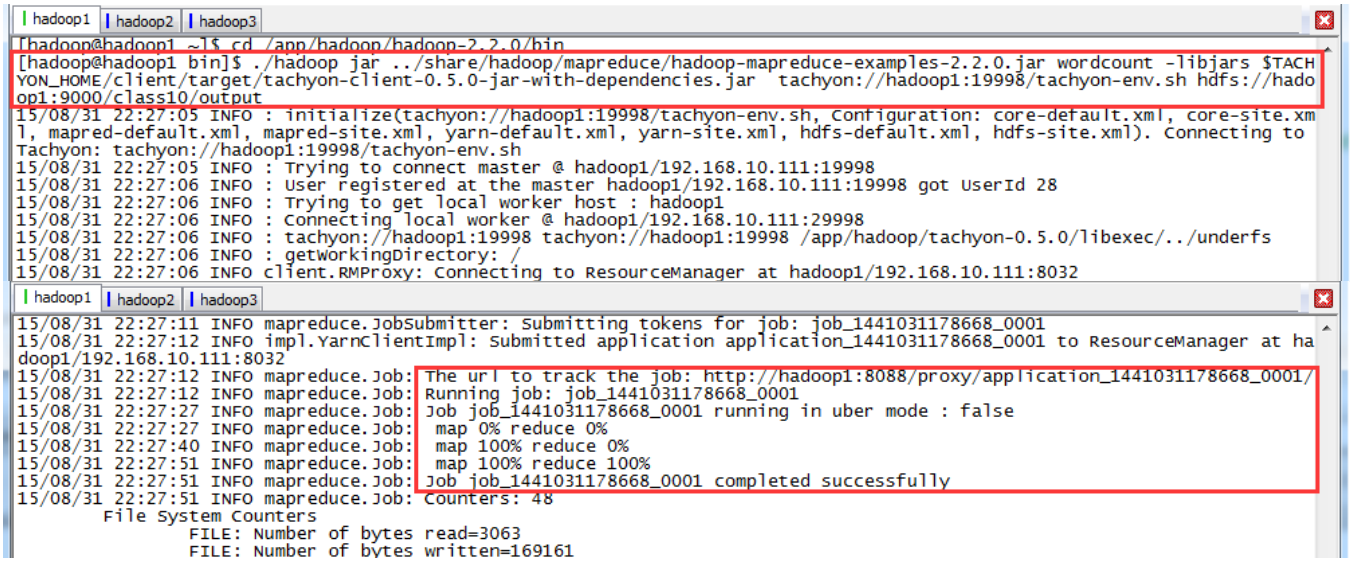
$cd /app/hadoop/hadoop-2.2.0/bin

$./hadoop fs -mkdir /class10

第二步 运行 MapReduce 例子

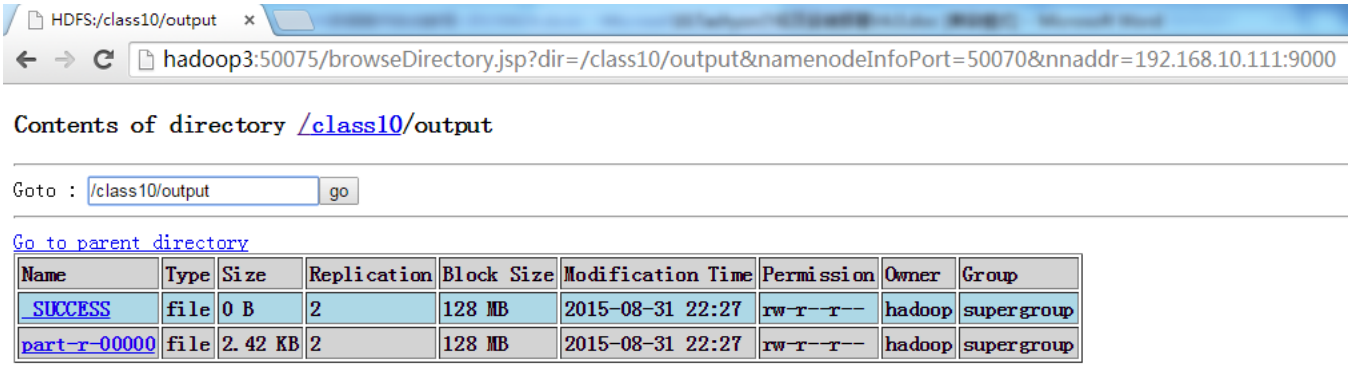
$cd /app/hadoop/hadoop-2.2.0/bin

$./hadoop jar ../share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar wordcount -libjars $TACHYON_HOME/client/target/tachyon-client-0.5.0-jar-with-dependencies.jar tachyon://hadoop1:19998/tachyon-env.sh hdfs://hadoop1:9000/class10/output

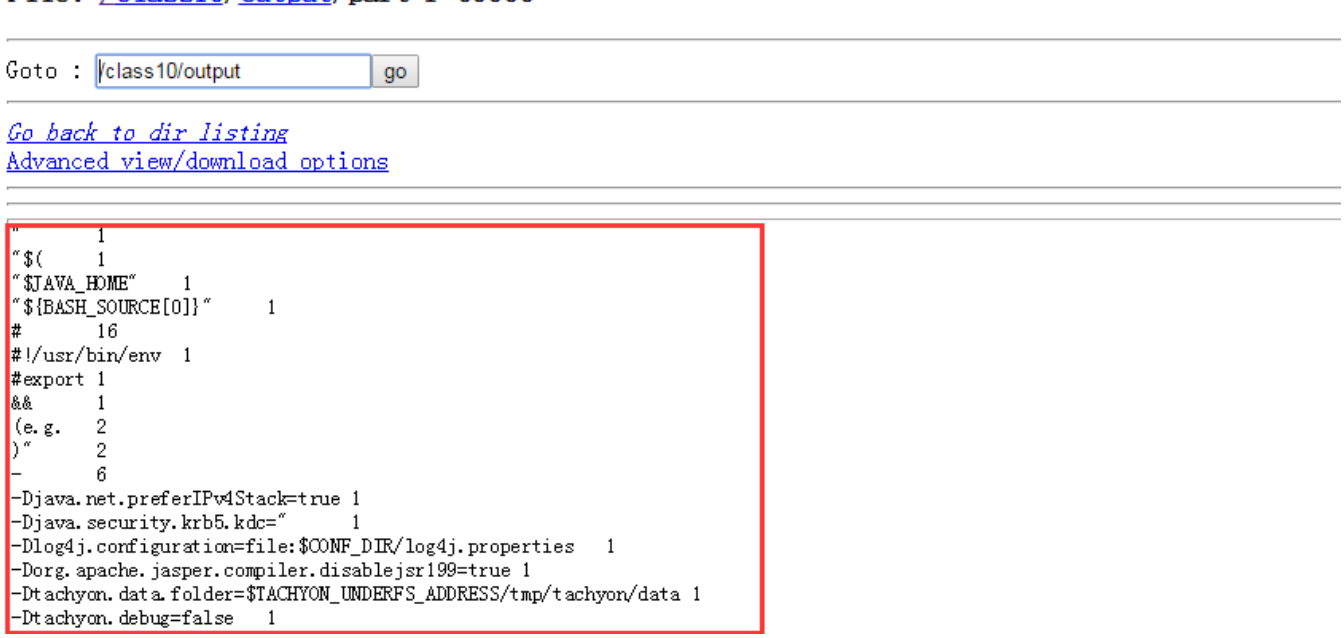


第三步 查看结果

查看 HDFS，可以看到在/class10 中创建了 output 目录



查看 part-r-0000 文件内容，为 tachyon-env.sh 单词计数



5 参考资料

（1）《 Tachyon ： Spark 生 态 系 统 中 的 分 布 式 内 存 文 件 系 统 》 http://www.csdn.net/article/2015-06-25/2825056

（2）《Tachyon 的安装、配置和使用》 http://blog.csdn.net/u014252240/article/details/42238081

（3）Tachyon 官方网站 http://tachyon-project.org/documentation/Running-Spark-on-Tachyon.html