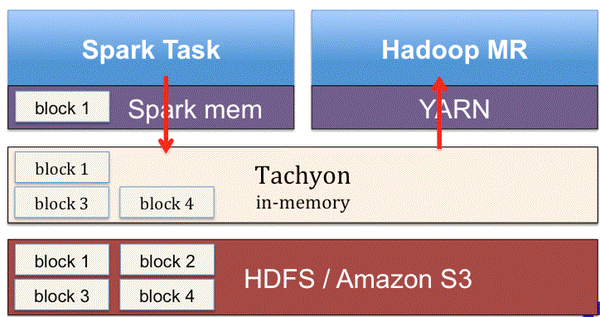
**1、Tachyon介绍**

**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：

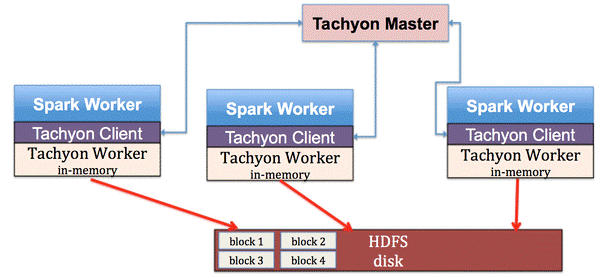
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124238310-180321718.gif)

 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**

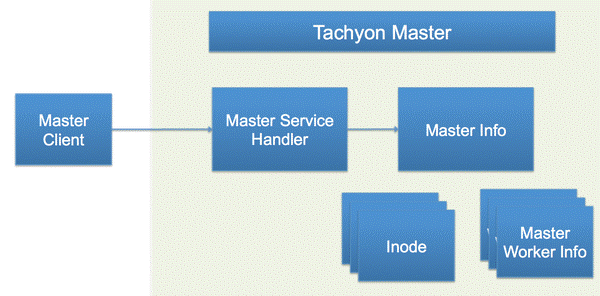
**1.2.1 系统架构**

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。

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124240216-736913646.gif)

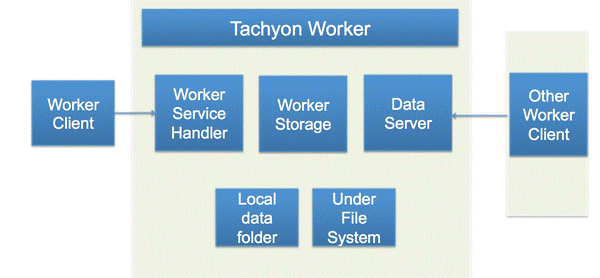
**1.2.2 Tachyon Master结构**

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。

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124241372-881684657.gif)

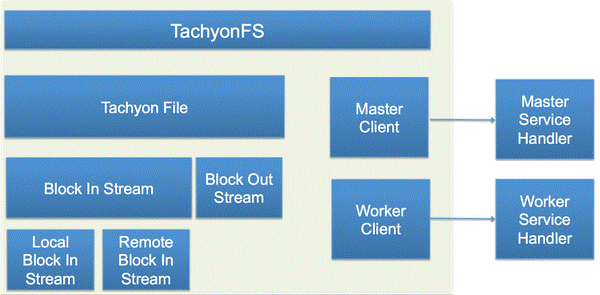
**1.2.3 Tachyon Worker结构**

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。

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124242606-308964567.gif)

**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。

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124254138-661803176.gif)

**1.2.5 场景说明**

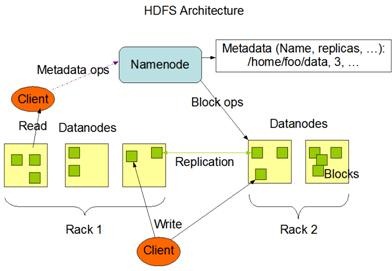
现在我们通过一个简单的场景把各个部件都串起来：假设一个Spark作业发起了一个读请求，它首先会通过Tachyon Client去Tachyon Master查询所需要的Block所在的位置。如果所在的Block不在本地的Tachyon Worker里，此Client则会通过Remote Block In Stream向别的Tachyon Worker发出读请求，同时在Block读入的过程中，Client也会通过Block Out Stream把Block写入到本地的内存存储里，这样就可以保证下次同样的请求可以由本机完成。

**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。

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124255106-1202835634.jpg)

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**

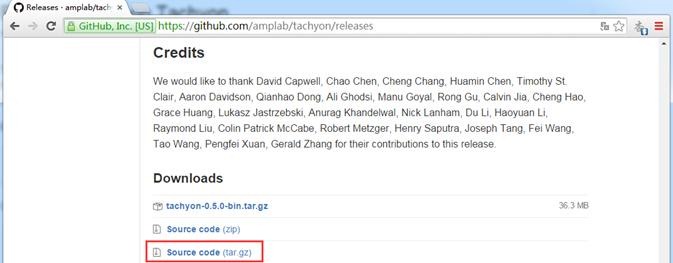
**2.1.1 下载并上传源代码**

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：

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124256247-1535674444.jpg)

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：

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124257216-1141235856.jpg)

第二步   在主节点上解压缩

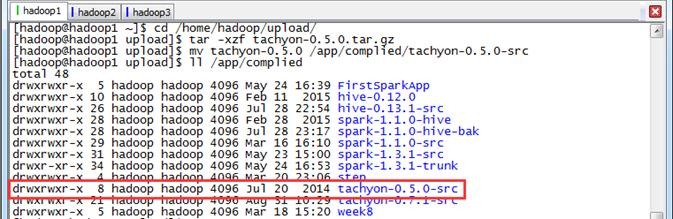
*$cd /home/hadoop/upload/*

*$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*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124258122-413075156.jpg)

**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：

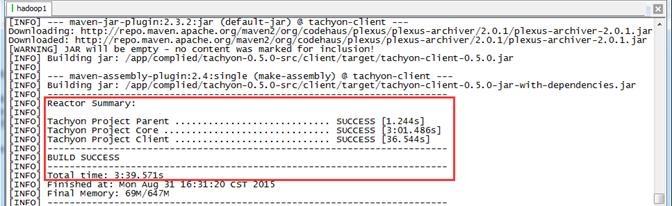
*$cd /app/complied/tachyon-0.5.0-src*

*$export MAVEN\_OPTS="-Xmx2g -XX:MaxPermSize=512M -XX:ReservedCodeCacheSize=512m"*

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

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124259169-1765142528.jpg)

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。

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124300310-1294158160.jpg)

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*

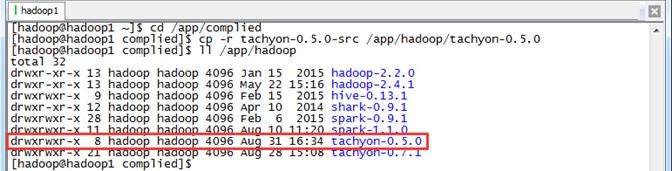
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124301403-114898130.jpg)

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*

*[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124302356-421026748.jpg)*

**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**

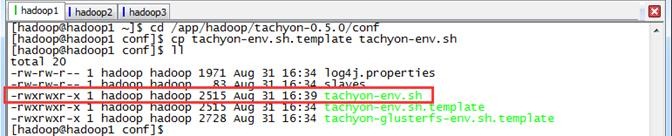
Tachyon相关配置文件在$TACHYON\_HOME/conf目录下，在workers文件中配置需要启动TachyonWorker的节点，默认是localhost，所以在单机模式下不用更改（在Tachyon-0.5.0版本中，该文件为slaves）。在这里需要修改tachyon-env.sh配置文件，具体操作是将tachyon-env.sh.template复制为tachyon-env.sh：

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

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

*$ll*

*$vi tachyon-env.sh*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124303231-2061802901.jpg)

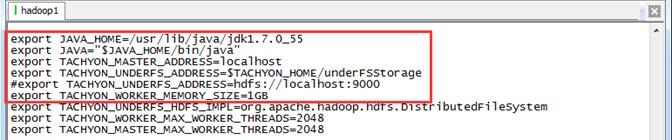
并在tachyon-env.sh中修改具体配置，下面列举了一些重要的配置项：

**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

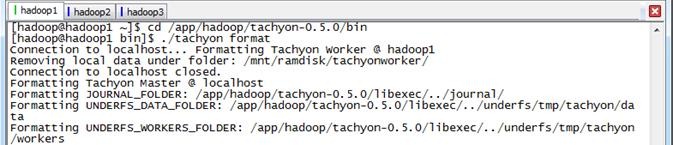
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124304497-1244721044.jpg)

**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*

*[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124305435-1842123988.jpg)*

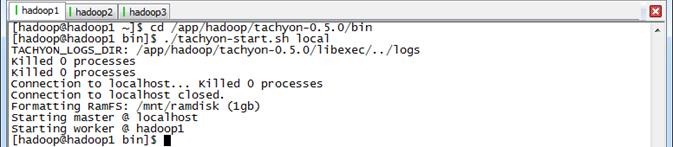
存储文件为$TACHYON\_HOME/underfs/tmp/tachyon目录下

**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*

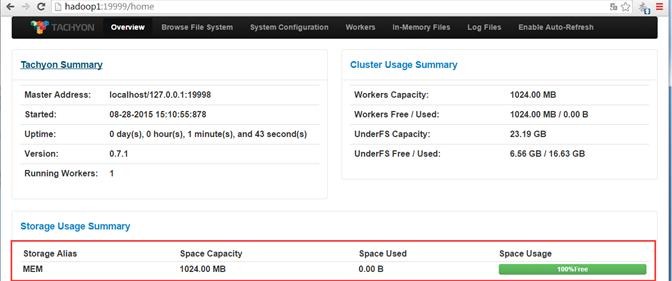
*[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124306419-1058736758.jpg)*

**2.2.4 验证启动**

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

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124307325-657431280.jpg)

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

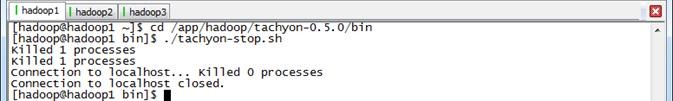
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124308372-1274324662.jpg)

**2.2.5 停止Tachyon**

停止Tachyon的命令为：

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

*$./tachyon-stop.sh*

*[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124309153-1716861317.jpg)*

**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：

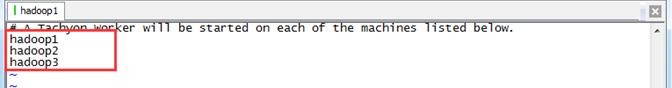
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **IP地址** | **机器名** | **运行进程** | **核数/内存** | **用户名** | **目录** |
| 1 | 192.168.0.61 | hadoop1 | TachyonMaster  TachyonWorker | 1核/3G | hadoop | /app  /app/hadoop  /app/hadoop/Tach... |
| 2 | 192.168.0.62 | hadoop2 | TachyonWorker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

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124309981-1091895741.jpg)

**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*

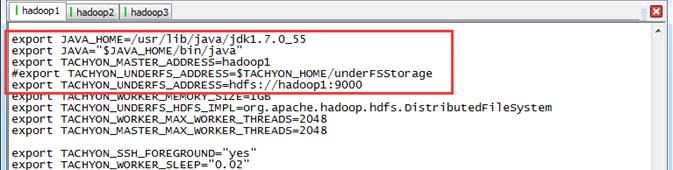
*$cp tachyon-env.sh.template tachyon-env.sh*

*$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*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124310841-459074245.jpg)

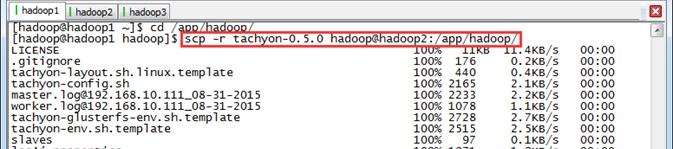
**2.3.4 向各个节点分发Tachyon**

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/*

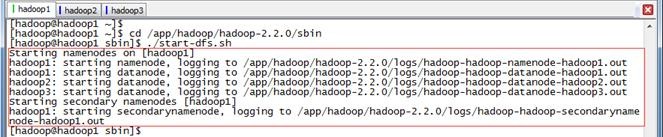
*$scp -r tachyon-0.5.0 hadoop@hadoop3:/app/hadoop/*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124311997-1257359994.jpg)

**2.3.5 启动HDFS**

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

*$./start-dfs.sh*

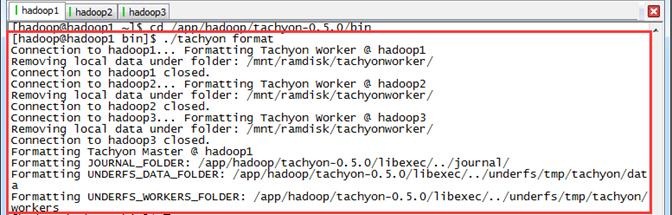
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124312856-1356786617.jpg)

**2.3.6 格式化Tachyon**

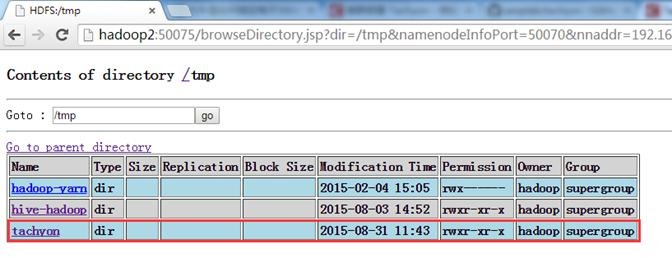
启动前需要格式化存储文件，格式化命令为：

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

*$./tachyon format*

*[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124313872-2031786773.jpg)*

可以看到在HDFS的/tmp创建了tachyon文件夹

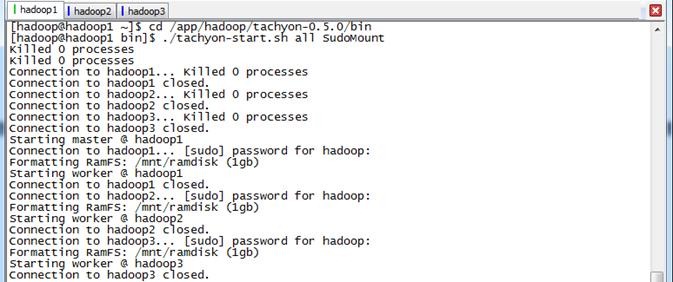
*[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124315356-117970974.jpg)*

**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*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124316888-973634877.jpg)

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。

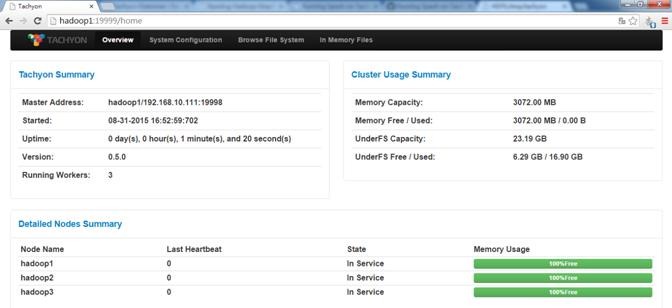
单机和集群式模式的区别就在于节点配置和启动步骤，事实上，也可以在集群模式下只设置一个TachyonWorker，此时就成为伪分布模式。

**2.3.8 验证启动**

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

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124318091-144619686.jpg)

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。

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124319247-953571567.jpg)

*$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>*

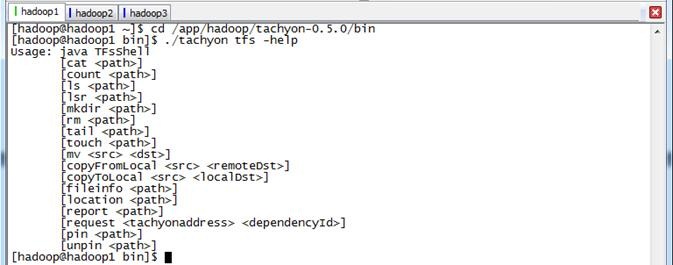
在Tachyon命令行使用中tachyon://<master node address>:<master node port>前缀可以省略，该信息从配置文件中读取。

**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*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124320325-1288269229.jpg)

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

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124321481-876856359.jpg)

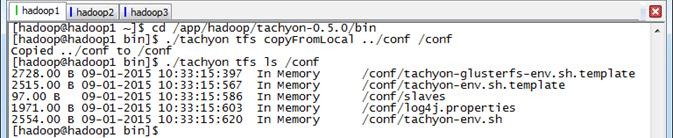
**3.2.1 copyFromLocal**

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*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124322310-423387339.jpg)

**3.2.2 copyToLocal**

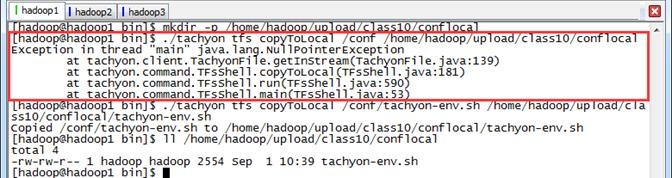
把Tachyon文件系统文件复制到本地，需要注意的是命令中的src必须是Tachyon文件系统中的文件不支持目录拷贝，否则报错无法复制

*$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*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124323216-779763289.jpg)

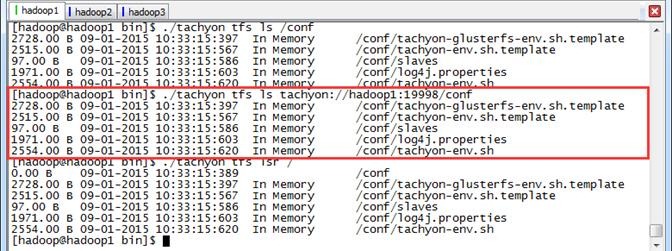
**3.2.3 ls和lsr**

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 /*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124324575-628741812.jpg)

**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 /*

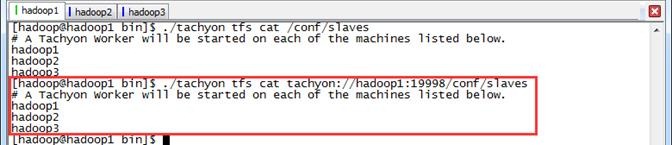
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124326169-957185852.jpg)

**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*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124327544-2143036329.jpg)

**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

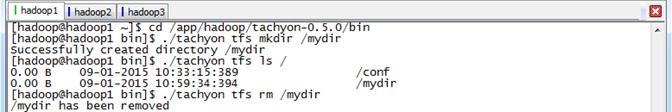
（3）**rmr**：删除目录，支持递归，包含子目录和文件，其中0.5.0版本不提供该命令

（4）**touch**：创建文件，不能创建已经存在的文件。

*$./tachyon tfs mkdir /mydir*

*$./tachyon tfs ls /*

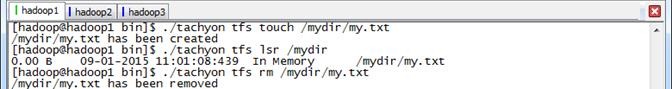
*$./tachyon tfs rm /mydir*

*[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124328481-990823749.jpg)*

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

*$./tachyon tfs lsr /mydir*

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

*[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124329216-233183636.jpg)*

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

*$./tachyon tfs lsr /mydir2*

*$./tachyon tfs rm /mydir2*

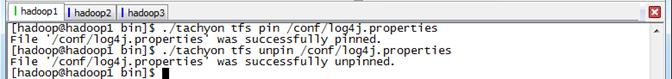
*$./tachyon tfs rm /*

*$./tachyon tfs ls /*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124330372-435475865.jpg)

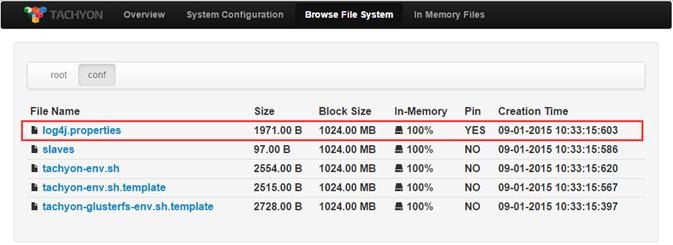
**3.2.7 pin和unpin**

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。

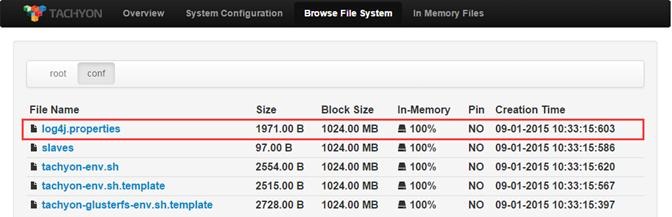
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124331435-2144159577.jpg)

pin执行前或unpin执行后的Web Interface界面

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

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124332403-732600351.jpg)

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

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124333450-1401357095.jpg)

**4、Tachyon实战应用**

**4.1 配置及启动环境**

**4.1.1 修改spark-env.sh**

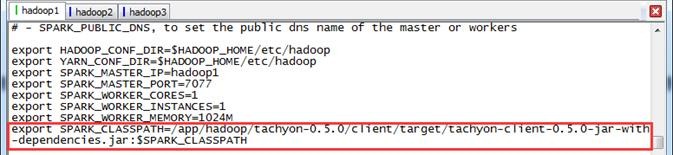
修改$SPARK\_HOME/conf目录下spark-env.sh文件：

*$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*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124334622-724255698.jpg)

**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>*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124335481-787817982.jpg)

**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**

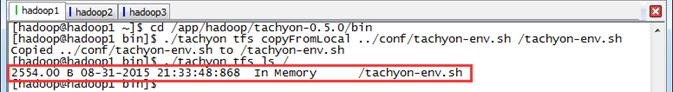
第一步   准备测试数据文件

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 /*

*[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124336325-1275770482.jpg)*

第二步   启动Spark-Shell

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

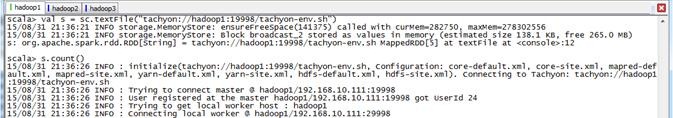
*$./spark-shell*

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")*

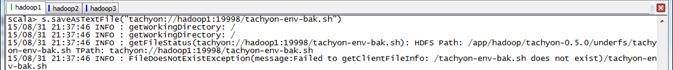
*scala>s.count()*

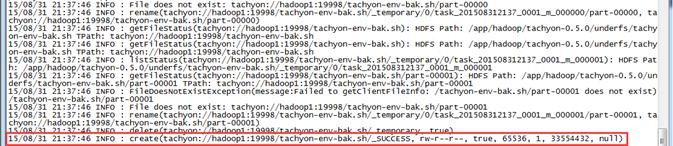
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124337435-1363955822.jpg)

[clip_image094](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124338356-1456266407.jpg)

把前面的测试文件另存为tachyon-env-bak.sh文件

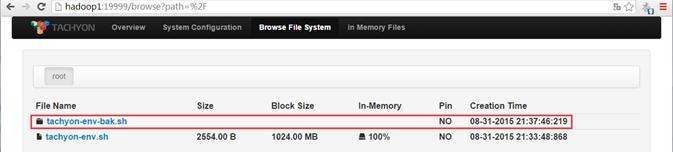
*scala>s.saveAsTextFile("tachyon://hadoop1:19998/tachyon-env-bak.sh")*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124339278-1924997004.jpg)

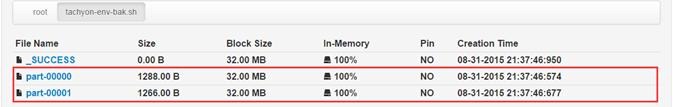
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124340310-1220302073.jpg)

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

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124341450-969353621.jpg)

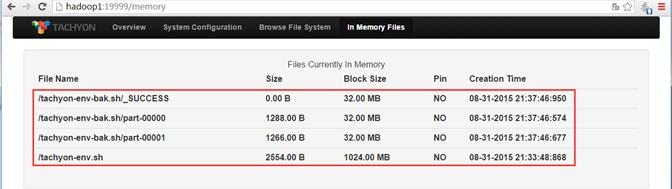
该文件夹中包含两个文件，分别为part-00000和part-00001：

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124342435-380268787.jpg)

其中tahyon-env-bak.sh/part-0001文件中内容如下：

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124343341-1530890189.jpg)

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：

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124344231-1913891309.jpg)

**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>*

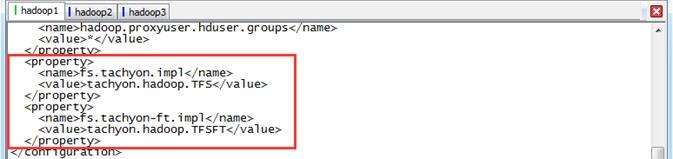
*</property>*

*<property>*

*<name>fs.tachyon-ft.impl</name>*

*<value>tachyon.hadoop.TFSFT</value>*

*</property>*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124345122-99130849.jpg)

**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

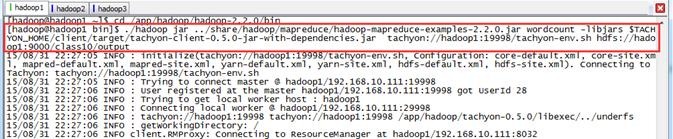
*$cd /app/hadoop/hadoop-2.2.0/bin*

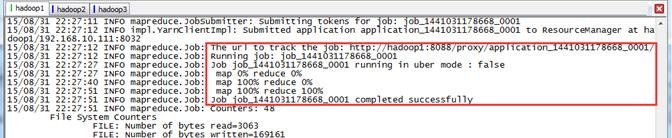
*$./hadoop fs -mkdir /class10*

第二步   运行MapReduce例子

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

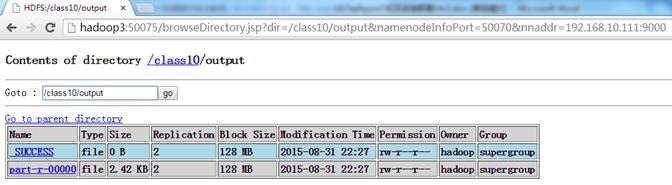
*$./hadoop jar ../share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar wordcount -libjars $TACHYON\_HOME/client/target/tachyon-client-0.5.0-jar-with-dependencies.jar tachyon://hadoop1:19998/tachyon-env.sh hdfs://hadoop1:9000/class10/output*

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124346013-32621043.jpg)

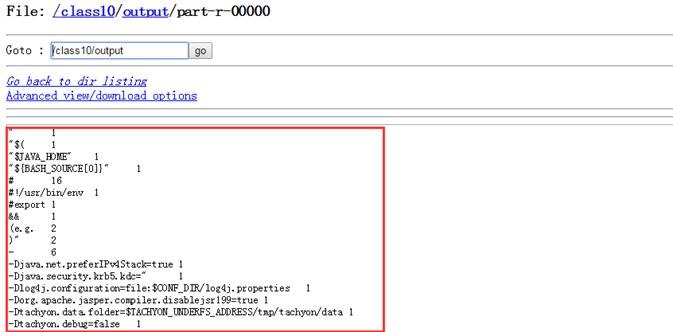
[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124347481-34387093.jpg)

第三步   查看结果

查看HDFS，可以看到在/class10中创建了output目录

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124349263-648005586.jpg)

查看part-r-0000文件内容，为tachyon-env.sh单词计数

[](http://images2015.cnblogs.com/blog/107289/201509/107289-20150901124350419-474230381.jpg)