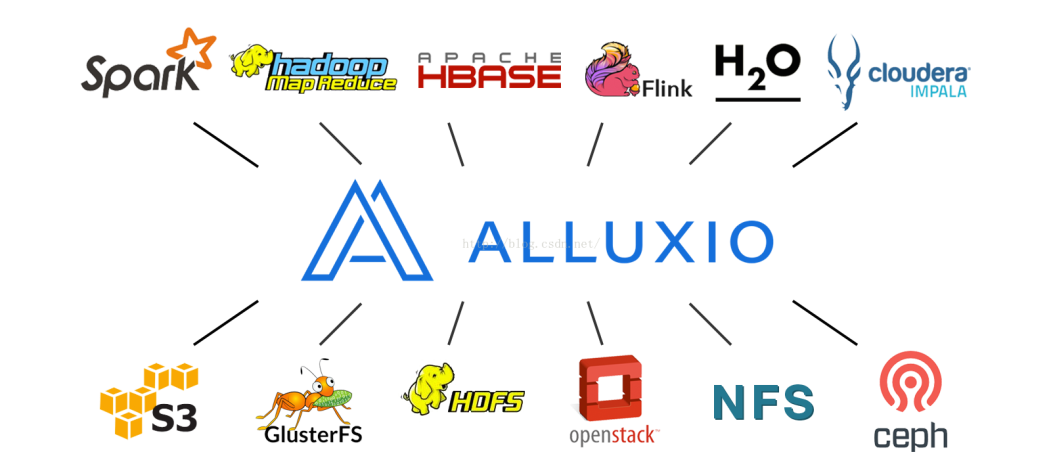
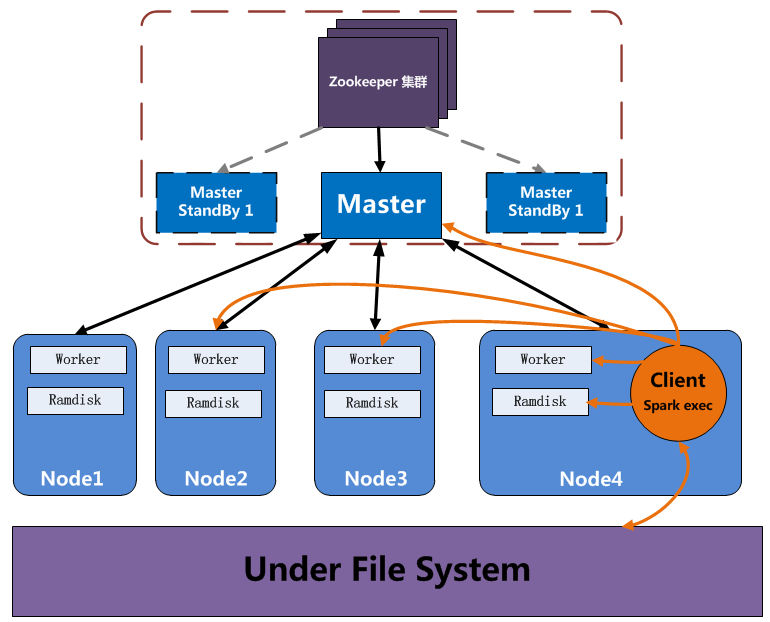
Alluxio简介

Alluxio是一个基于内存的分布式文件系统，架构在底层分布式文件系统和上层分布式计算框架之间的一个中间件，主要职责是以文件形式在内存或者它存储的设施中提供数据的存储服务，如下图所示：



在大数据领域，最底层的是分布式文件系统，如Amazon S3、Apache HDFS等，而较高层的应用则是一些分布式计算框架，如Spark,MapReduce,Flink等，这些分布式框架，往往都是直接从分布式文件系统中读写数据，效率比较低，性能消耗比较大。如果将其架构搭建底层分布式文件系统和上层分布式计算框架之间，以文件的形式在内存中对外提供读写访问服务，那么Alluxio可以为大数据应用提供一个数量级加速，而且它只要提供通用的数据访问接口，就能很方便的切换底层分布式文件系统。

Alluxio是主从架构，有主节点（AlluxioMaster）和多个从节点Worker(AlluxioWorker)构成，如下图所示：



当Alluxio的Worker节点启动之后，Worker会向Master注册。注册成功之后，Master会与Worker之间维护一个心跳。Alluxio会将内存映射为Block设备，就是RamDisk。然后Alluxio会将数据存放在每个Worker的RamDisk（内存）中。Alluxio可以以内存的速度去读写Ramdisk中存放的数据。Alluxio的各组件如下：

* Master，负责处理与存储系统的元数据，元数据通过树形结构来管理，树形结构的每一个叶子节点代表文件或者空目录。客户端或其他程序可以对元数据信息的访问或修改。对内（所有Worker）负责统计收集每个Worker周期心跳发送来的数据块信息与每个Worker的运行状态。Master不会主动与其他组件通讯，只以请求应答的方式与其他组件通讯。
* Worker，负责管理本地资源，如本地内存、SSD、磁盘等。提供Client读写数据请求服务，客户端或其他程序发来的数据将会以block的方式进行管理（一个文件的多个block会分散存储在多个Worker节点上）。每个Worker缓存数据时使用Ramfs，是Linux下的一种基于Ram做存储的文件系统，上层使用访问内存一样的速度访问类似磁盘空间。
* Client，使用数据的程序，例如Spark Executor、MR Task等其他程序。Alluxio允许Client通过与Master通讯来执行元数据操作，与Worker通讯来读写数据。同时客户端还可以直接绕过Alluxio访问底层存储系统。
* Under File System，真实的数据存储系统，可以是S3、HDFS、GlusterFS等存储系统。

# Alluxio安装

## 2.1 Running Alluxio Locally

Alluxio的执行依赖JDK，要确保Linux OS上已经安装好了

1）下载Alluxio 1.7.0的安装文件

*$wget http://alluxio.org/downloads/files/1.7.0/alluxio-1.7.0-bin.tar.gz*

*$tar xvfz alluxio-1.7.0-bin.tar.gz*

*$cd alluxio-1.7.0*

2）Standalone模式的而配置

*$cp conf/alluxio-site.properties.template conf/alluxio-site.properties*

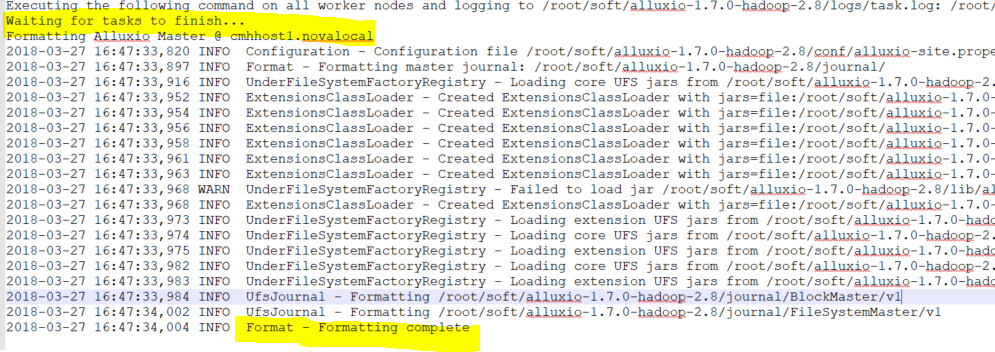
配置参数：

*alluxio.master.hostname=localhost*

*alluxio.underfs.address=/tmp*

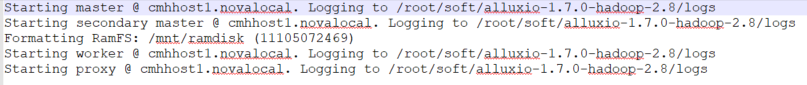
3）格式化Alluxio Filesystem（第一次启动）

*$bin/alluxio format*



4）启动Alluxio FileSystem

*$bin/alluxio-start.sh local*



5）查看进程及UI

*# jps |grep Alluxio\**

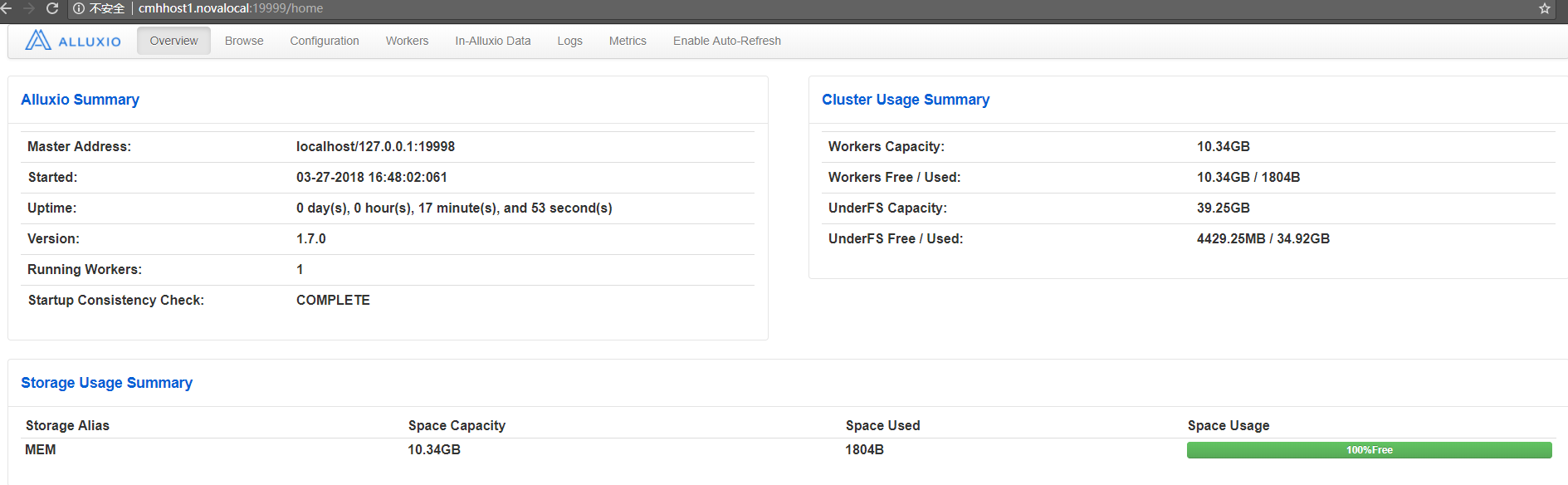
*9920 AlluxioWorker*

*9922 AlluxioProxy*

*9688 AlluxioMaster*

*9690 AlluxioSecondaryMaster*

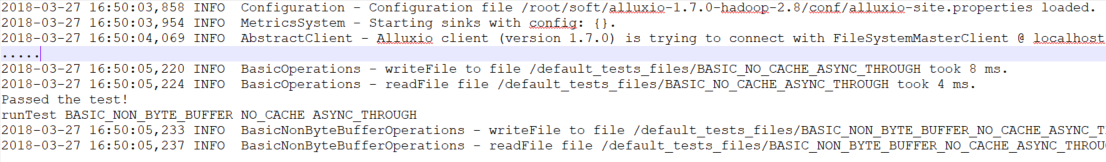
AlluxioMaster UI的默认端口为19999，打开后如下所示：



6）执行测试程序

*# bin/alluxio runTests*

输出如下：



https://www.alluxio.org/docs/1.7/en/Running-Alluxio-Locally.html

## Running Alluxio on a Cluster

JDK和ssh无密码登录的安装及设置不再介绍

1）下载安装包

*$wget http://alluxio.org/downloads/files/1.7.0/alluxio-1.7.0-bin.tar.gz*

*$tar xvfz alluxio-1.7.0-bin.tar.gz*

*$cd alluxio-1.7.0*

2）配置Alluxio，在Cluster模式下Alluxio不能使用Local FS作为Alluxio Under Storage System，因此使用HDFS作为Shared Storage

*$cp conf/alluxio-site.properties.template conf/alluxio-site.properties*

配置

*alluxio.master.hostname=cmhhost1.novalocal*

*alluxio.underfs.address=hdfs://nn/*

将hdfs的配置文件hdfs-site.xml和core-site.xml复制到conf目录下

*$cp /etc/hadoop/conf/hdfs-site.xml conf/*

*$cp /etc/hadoop/conf/core-site.xml conf/*

配置workers

*cmhhost1.novalocal*

*cmhhost2.novalocal*

*cmhhost3.novalocal*

3）格式化alluxio

*$bin/alluxio format*

4）启动alluxio

*# bin/alluxio-start.sh all*

5) 查看Master/Worker上的进程，UI等

Master上的进程

*# jps |grep Alluxio\**

*15346 AlluxioWorker*

*15077 AlluxioMaster*

*15593 AlluxioProxy*

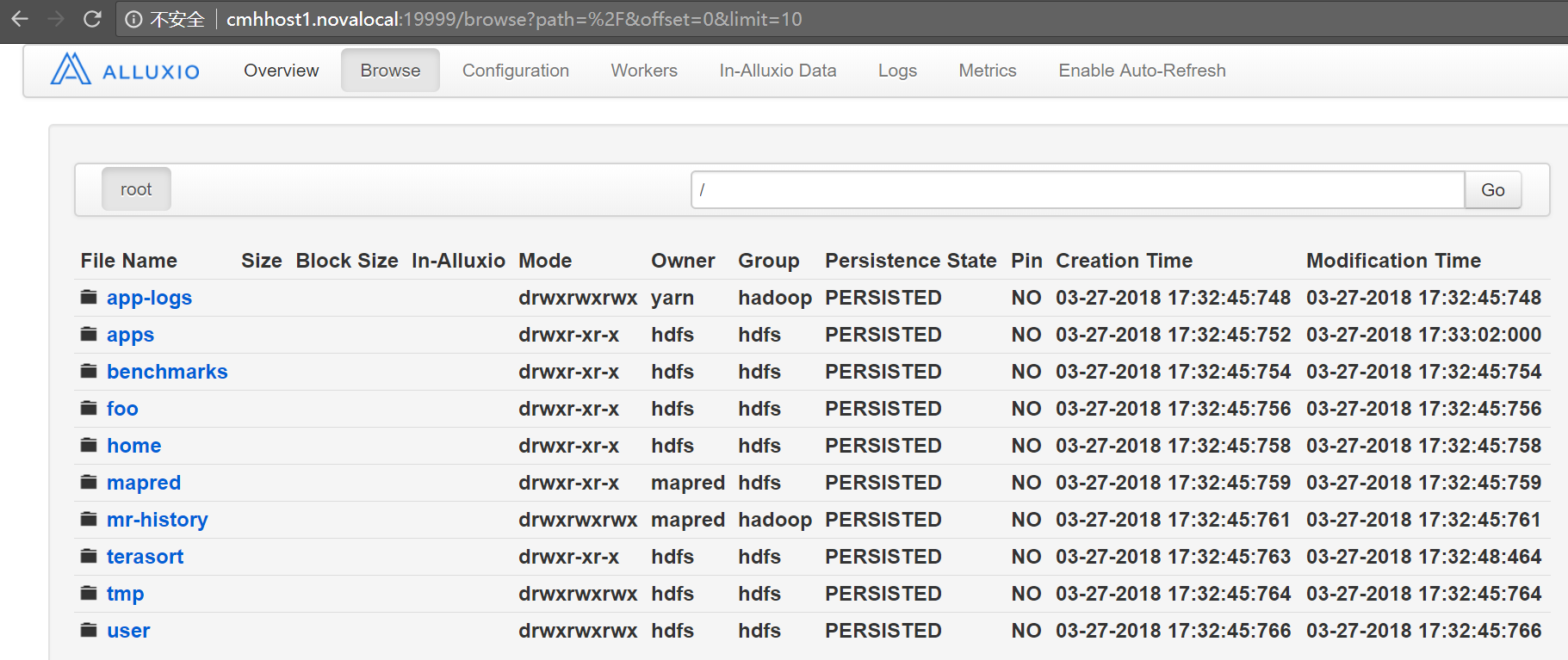
Worker上的进程

*# jps|grep Alluxio*

*5415 AlluxioProxy*

*5343 AlluxioWorker*

UI界面如下：



查看到的文件是HDFS上的文件。

6）执行test

*# bin/alluxio runTests*

执行成功

# Alluxio使用

## 2.1 Command Line接口

Alluxio命令行接口为用户提供了基本的文件系统操作，可以使用以下命令来得到所有子命令:

*# bin/alluxio fs*

*Usage: alluxio fs [generic options]*

*[cat <path>]*

*[checkConsistency [-r] <Alluxio path>]*

*……*

对于用Alluxio URI(如ls,mkdir)作为参数的fs子命令来说，参数应该是完全的Alluxio URI:

*alluxio://<master-hostname>:<master-port>/path*

或者是没有头部信息的/<path>，主机名和端口在conf/alluxio-site.properties中配置。

在alluxio fs命令中，大多数需要路径参数的命令可以使用通配符简化使用，例如：

*./bin/alluxio fs rm '/data/2014\*'*

该示例命令将data文件夹下以2014为文件名前缀的所有文件删除。

### 2.1.1 Alluxio FS Shell操作

下表是操作列表

|  |  |  |
| --- | --- | --- |
| 操作 | 语法 | 描述 |
| cat | cat “path” | 将Alluxio中一个文件内容打印到控制台上 |
| checkConsistency | checkConsistency “path” | 检查Alluxio与底层存储系统的元数据一致性 |
| checksum | checksum "path" | 计算一个文件的md5校验码 |
| chgrp | chgrp "group" "path" | 修改Alluxio中的文件或文件夹的所属组 |
| chmod | chmod "permission" "path" | 修改Alluxio中文件或文件夹的访问权限 |
| chown | chown "owner" "path" | 修改Alluxio中文件或文件夹的所有者 |
| copyFromLocal | copyFromLocal "source path" "remote path" | 将“source path”指定的本地文件系统中的文件拷贝到Alluxio中"remote path"指定的路径 如果"remote path"已经存在该命令会失败 |
| copyToLocal | copyToLocal "remote path" "local path" | 将"remote path"指定的Alluxio中的文件复制到本地文件系统中 |
| count | count "path" | 输出"path"中所有名称匹配一个给定前缀的文件及文件夹的总数 |
| cp | cp "src" "dst" | 在Alluxio文件系统中复制一个文件或目录 |
| du | du "path" | 输出一个指定的文件或文件夹的大小 |
| fileInfo | fileInfo "path" | 输出指定的文件的数据块信息 |
| free | free "path" | 将Alluxio中的文件或文件夹移除，如果该文件或文件夹存在于底层存储中，那么仍然可以在那访问 |
| getCapacityBytes | getCapacityBytes | 获取Alluxio文件系统的容量 |
| help | help "cmd" | 打印给定命令的帮助信息，如果没有给定命令，打印所有支持的命令的帮助信息 |
| leader | leader | 打印当前Alluxio leader master节点主机名 |
| load | load "path" | 将底层文件系统的文件或者目录加载到Alluxio中 |
| loadMetadata | loadMetadata "path" | 将底层文件系统的文件或者目录的元数据加载到Alluxio中 |
| location | location "path" | 输出包含某个文件数据的主机 |
| ls | ls "path" | 列出给定路径下的所有直接文件和目录的信息，例如大小 |
| masterInfo | masterInfo | 打印Alluxio master容错相关的信息，例如leader的地址、所有master的地址列表以及配置的Zookeeper地址 |
| mkdir | mkdir "path1" ... "pathn" | 列出给定路径下的所有直接文件和目录的信息，例如大小 |
| mount | mount "path" "uri" | 将底层文件系统的"uri"路径挂载到Alluxio命名空间中的"path"路径下，"path"路径事先不能存在并由该命令生成。 没有任何数据或者元数据从底层文件系统加载。当挂载完成后，对该挂载路径下的操作会同时作用于底层文件系统的挂载点。 |
| mv | mv "source" "destination" | 将"source"指定的文件或文件夹移动到"destination"指定的新路径，如果"destination"已经存在该命令失败。 |
| persist | persist "path1" ... "pathn" | 将仅存在于Alluxio中的文件或文件夹持久化到底层文件系统中 |
| pin | pin "path" | 将给定文件锁定到内容中以防止剔除。如果是目录，递归作用于其子文件以及里面新创建的文件 |
| report | report "path" | 向master报告一个文件已经丢失 |
| rm | rm "path" | 删除一个文件，如果输入路径是一个目录该命令失败 |
| setTtl | setTtl "path" "time" | 设置一个文件的TTL时间，单位毫秒 |
| stat | stat "path" | 显示文件和目录指定路径的信息 |
| tail | tail "path" | 将指定文件的最后1KB内容输出到控制台 |
| test | test "path" | 测试路径的属性，如果属性正确，返回0，否则返回1 |
| touch | touch "path" | 在指定路径创建一个空文件 |
| unmount | unmount "path" | 卸载挂载在Alluxio中"path"指定路径上的底层文件路径，Alluxio中该挂载点的所有对象都会被删除，但底层文件系统会将其保留。 |
| unpin | unpin "path" | 将一个文件解除锁定从而可以对其剔除，如果是目录则递归作用 |
| unsetTtl | unsetTtl "path" | 删除文件的ttl值 |

### 2.1.2 Alluxio Shell使用示例

1）copyFromLocal

copyFromLocal命令将本地文件系统中的文件拷贝到Alluxio中，如果你运行该命令的机器上有Alluxio worker，那么数据便会存放在这个worker上，否则，数据将会随机地复制到一个运行Alluxio worker的远程节点上。如果该命令指定的目标是一个文件夹，那么这个文件夹及其所有内容都会被递归复制到Alluxio中。

使用举例：

*./bin/alluxio fs copyFromLocal /local/data /input*

2）load

load 命令将底层文件系统中的数据载入到Alluxio中。如果运行该命令的机器上正在运行一个Alluxio worker，那么数据将移动到该worker上，否则，数据会被随机移动到一个worker上。 如果该文件已经存在在Alluxio中，设置了--local选项，并且有本地worker，则数据将移动到该worker上。 否则该命令不进行任何操作。如果该命令的目标是一个文件夹，那么其子文件和子文件夹会被递归载入。

*./bin/alluxio fs load /data/today*

3）persist

persist命令将Alluxio中的数据持久化到底层文件系统中。该命令是对数据的操作，因而其执行时间取决于该文件的大小。在持久化结束后，该文件即在底层文件系统中有了备份，因而该文件在Alluxio中的数据块被剔除甚至丢失的情况下，仍能够访问。

使用举例：在从一系列临时文件中过滤出包含有用数据的文件后，便可以使用persist命令对其进行持久化。

*./bin/alluxio fs persist /tmp/experimental-logs-2.txt*

https://www.alluxio.org/docs/master/cn/Command-Line-Interface.html

## 2.2 在Alluxio上运行Hadoop MapReduce

1）配置Hadoop的core-site.xml，添加Alluxio相关的配置：

*<property>*

*<name>fs.alluxio.impl</name>*

*<value>alluxio.hadoop.FileSystem</value>*

*<description>The Alluxio FileSystem (Hadoop 1.x and 2.x)</description>*

*</property>*

*<property>*

*<name>fs.AbstractFileSystem.alluxio.impl</name>*

*<value>alluxio.hadoop.AlluxioFileSystem</value>*

*<description>The Alluxio AbstractFileSystem (Hadoop 2.x)</description>*

*</property>*

该配置让MapReduce的输出和输入文件通过Alluxio scheme alluxio://来识别URI。

2）将alluxio-1.7.0-client.jar分发到各节点的/usr/lib/hadoop目录下

3）提交程序，命令如下：

*hadoop jar*

*/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples-2.8.2-bc1.4.0.jar wordcount*

*alluxio://cmhhost1.novalocal:19998/data/input*

*alluxio://cmhhost1.novalocal:19998/data/output*

执行成功，查看结果

*$bin/alluxio fs ls /data/output*

*NOT\_PERSISTED 03-30-2018 17:47:25:183 100% /data/output/part-r-00000*

*NOT\_PERSISTED 03-30-2018 17:47:26:059 100% /data/output/\_SUCCESS*

中文官网：

https://www.alluxio.org/docs/master/cn/Getting-Started.html

https://www.alluxio.org/docs/1.7/en/Running-Alluxio-on-a-Cluster.html