



hadoop 官网帮助手册[第二版]

日期: 20160225

版本: hadoop2.7.1/hadoop2.7.2

目录

hadoop 入门-第一章 General: 第一节单节点伪分布.....	5
hadoop 入门-第一章 General: 第二节集群配置.....	16
hadoop 入门-第一章 General: 第三节 Hadoop 初级入门之命令指南.....	29
hadoop 入门-第一章 General: 第四节文件系统 shell.....	33
hadoop 入门-第一章 General: 第五节 hadoop 的兼容性说明	47
hadoop 入门-第一章 General: 第六节开发人员和用户接口指南: hadoop 接口分类 ...	56
hadoop 入门-第一章 General: 第七节 Hadoop 文件系统 API : 概述.....	61
hadoop 入门-第二章 common: 第一节 hadoop 本地库 指南.....	61
hadoop 入门-第二章 common: 第二节 hadoop 代理用户 -超级用户代理其它用户.....	65
hadoop 入门-第二章 common: 第三节机架智能感知	68
hadoop 入门-第二章 common: 第四节安全模式说明	73

hadoop 入门-第二章 common: 第五节 服务级别授权指南	86
hadoop 入门-第二章 common: 第六节 Hadoop HTTP web-consoles 认证机制	90
hadoop 入门-第二章 common: 第七节 Hadoop Key 管理服务器(KMS) - 文档集	94
hadoop 入门:第三章 HDFS 文档概述	116
hadoop 入门:第四章 mapreduce 文档概述	135
hadoop 入门:第五章 MapReduce REST APIs 文档概述	138
hadoop 入门:第六章 YARN 文档概述	140
hadoop 入门:第九章 hadoop 认证	151
hadoop 入门:第十章 hadoop 工具	153
hadoop 入门:第十一章 hadoop 配置	158
捐助: 大数据、云技术视频	163
hadoop 生态系统零基础入门及大数据实战【后续不断更新】	163

about 云零基础开发、部署+理论 openstack 入门视频【J 版及 K 版】165

hadoop 学习的方式有很多种，比如视频，文档，书籍等，上面的知识都是经过别人消化，传播给大家。那么最本质，基础的知识，还是官网的内容。这个手册是根据官网翻译的。

这个手册的目的，

一是学习，也可以后期备用查询，

二是如果你也想从官网学习，那么这个手册相信可以起到一定得作用。

后面第三章开始，没有对每篇文章进行翻译，只是对文档进行了简单的介绍，我们知道了文档的作用，在用到的时候，知道去哪找，这个是很重要的。相信如果你想翻译的话，无论是英文水平还是个人技能都会有一定的提高。

前两章是基于 hadoop2.7.1，第三章开始是基于 hadoop2.7.2，版本是不断更新的，但是文档的作用是不变的，内容变化也不是很大。对于提供的英文链接，可能会随着版本的更新而失效，大家只要知道官网 <http://hadoop.apache.org/>，然后到上面找到 document，相信就可以找到对应版或则最新版的文档说明了。

里面如有理解不对的地方，希望大家批评指正，共同学习和进步。

hadoop 入门-第一章 General: 第一节单节点伪分布

问题导读

- 1.从本文部署实际部署，总结本地模式、伪分布、分布式的区别是什么？
- 2.单机是否是伪分布？
- 3.本地模式是否可以运行 **mapreduce**？



hadoop2.7 发布，这一版不太适合用于生产环境，但是并不影响学习：由于 hadoop 安装方式有三种，并且三种安装方式都可以在前面的基础上继续配置，分别是：

- 本地模式
- 伪分布
- 分布式

#####

1.准备

安装 **jdk1.7** 参考

[linux\(ubuntu\)安装 Java jdk 环境变量设置及小程序测试](#)

测试:

Java -version

```
aboutyun@ubuntu: ~/hadoop-2.7.0/etc/hadoop$ cd
aboutyun@ubuntu:~$ java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
```

安装 ssh

```
sudo apt-get install ssh
```

```
$ ssh-keygen -t dsa -P "" -f ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
$ export HADOOP_PREFIX=/usr/local/hadoop
```

最后达到无密码登录

```
ssh localhost
```

```
aboutyun@ubuntu:~$ ssh localhost
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-24-generic x86_64)

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

734 packages can be updated.
316 updates are security updates.

New release '14.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Apr 27 00:25:43 2015 from localhost
aboutyun@ubuntu:~$
```

安装 rsync

```
sudo apt-get install rsync
```

修改网卡:

注释掉 127.0.1.1 ubuntu

添加新的映射

10.0.0.81 ubuntu

```
127.0.0.1    localhost
#127.0.1.1  ubuntu
10.0.0.81   ubuntu
# The following lines are desirable for IPv6 capable hosts
::1        ip6-localhost ip6-loopback
fe00::0    ip6-localnet
ff00::0    ip6-mcastprefix
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

这里必须修改，否则后面会遇到连接拒绝等问题

2. 安装

进入配置文件目录

我这里是

~/hadoop-2.7.0/etc/hadoop

```
aboutyun@ubuntu:~/hadoop-2.7.0/etc/hadoop$ ls
capacity-scheduler.xml  httpfs-env.sh          mapred-env.sh
configuration.xml      httpfs-log4j.properties mapred-queues.xml.template
container-executor.cfg httpfs-signature.secret mapred-site.xml.template
core-site.xml          httpfs-site.xml       slaves
hadoop-env.cmd         kms-acls.xml          ssl-client.xml.example
hadoop-env.sh         kms-env.sh            ssl-server.xml.example
hadoop-metrics2.properties kms-log4j.properties yarn-env.cmd
hadoop-metrics.properties kms-site.xml          yarn-env.sh
hadoop-policy.xml     log4j.properties     yarn-site.xml
hdfs-site.xml         mapred-env.cmd
```

修改配置文件:

etc/hadoop/hadoop-env.sh

添加 JAVA_HOME、HADOOP_COMMON_HOME

```
export JAVA_HOME=/usr/jdk
export HADOOP_COMMON_HOME=~/.hadoop-2.7.0
```

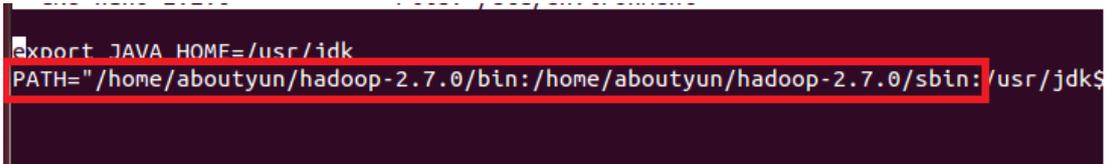
配置环境变量

sudo nano /etc/environment

增加 hadoop 配置

将下面添加到变量 PATH 中

```
/home/aboutyun/hadoop-2.7.0/bin:/home/aboutyun/hadoop-2.7.0/sbin:
```

A terminal window with a dark background. The prompt is root@ubuntu:~#. Two lines of code are entered: export JAVA_HOME=/usr/jdk and PATH="/home/aboutyun/hadoop-2.7.0/bin:/home/aboutyun/hadoop-2.7.0/sbin:/usr/jdk\$. The second line is highlighted with a red rectangular box.

#####

3.本地模式验证[可忽略]

所谓的本地模式：在运行程序的时候，比如 wordcount 是在本地磁盘运行的

上面已经配置完毕，我们对其测试，分别执行面命令：

注意： bin/hadoop 的执行条件是在 hadoop_home 中，我这里是

```
$ mkdir input
```

```
$ cp etc/hadoop/*.xml input
```

```
bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.0.jar grep input output 'dfs[a-z.]+'
```

```
aboutyun@ubuntu:~/hadoop-2.7.0$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.0.jar grep input output 'dfs[a-z.]+'
1 dfsadmin
```

```
$ cat output/*
```

```
aboutyun@ubuntu:~/hadoop-2.7.0$ cat output/*
1 dfsadmin
```

#####

上面本地模式，我们知道就可以，我们下面继续配置伪分布模式

4.伪分布模式

我这里的全路径：/home/aboutyun/hadoop-2.7.0/etc/hadoop

修改文件 etc/hadoop/core-site.xml

添加如下内容：

含义：接收 Client 连接的 RPC 端口，用于获取文件系统 metadata 信息。

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

修改 etc/hadoop/hdfs-site.xml:

添加如下内容:

含义: 备份只有一份

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```



```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

#####

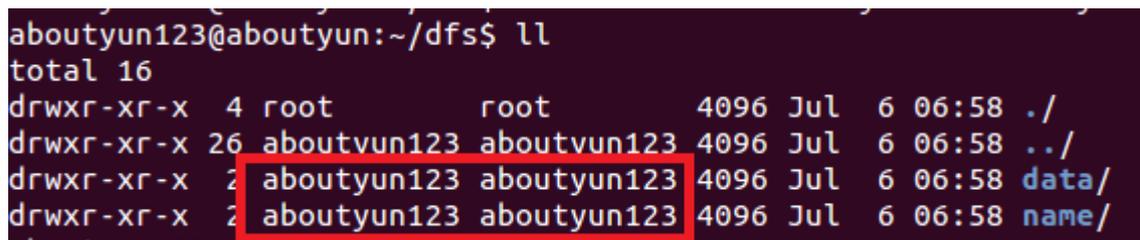
补充重要:

由于系统重启后, 找不到 namenode 进程, 这是因为系统在重启后被删除, 所以加入下面配置

```
[mw_shl_code=xml,true]<property>
  <name>dfs.namenode.name.dir</name>
  <value>file:/home/aboutyun123/dfs/name</value>
</property>
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:/home/aboutyun123/dfs/data</value>
</property>
```

[/mw_shl_code]

同时, 注意权限



```
aboutyun123@aboutyun:~/dfs$ ll
total 16
drwxr-xr-x  4 root      root      4096 Jul  6 06:58 ./
drwxr-xr-x 26 aboutyun123 aboutyun123 4096 Jul  6 06:58 ../
drwxr-xr-x  2 aboutyun123 aboutyun123 4096 Jul  6 06:58 data/
drwxr-xr-x  2 aboutyun123 aboutyun123 4096 Jul  6 06:58 name/
```

#####

5. 伪分布模式

1. 格式化 namenode

```
hdfs namenode -format
```

有的地方使用

```
bin/hdfs namenode -format
```

如果配置的环境变量直接使用 `hdfs namenode -format` 即可

2. 启动集群

```
start-dfs.sh
```

```
aboutyun@ubuntu:~$ start-dfs.sh
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/aboutyun/hadoop-2.7.0/logs/hadoop-
aboutyun-namenode-ubuntu.out
localhost: starting datanode, logging to /home/aboutyun/hadoop-2.7.0/logs/hadoop-
aboutyun-datanode-ubuntu.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/aboutyun/hadoop-2.7.0/logs
/hadoop-aboutyun-secondarynamenode-ubuntu.out
aboutyun@ubuntu:~$
```

这时候单节点伪分布就已经安装成功了

#####

验证[可忽略]

输入下面

```
http://localhost:50070/
```

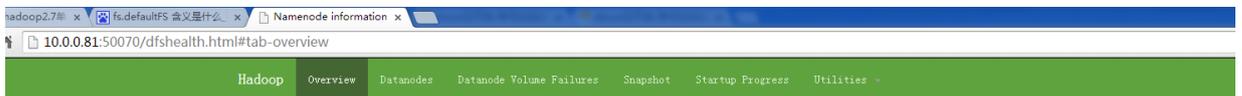
如果是在虚拟机中安装，但是在宿主主机中访问，需要输入虚拟机 ip 地址

这里虚拟机 ip 地址是 10.0.0.81

```
aboutyun@ubuntu:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:50:56:31:35:02 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.81/24 brd 10.0.0.255 scope global eth0
    inet6 fe80::250:56ff:fe31:3502/64 scope link
        valid_lft forever preferred_lft forever
```

所以，我这里是

<http://10.0.0.81:50070/>



Overview 'localhost:9000' (active)

Started:	Mon Apr 27 03:47:08 PDT 2015
Version:	2.7.0, rUnknown
Compiled:	2015-04-22T14:32Z by root from Unknown
Cluster ID:	CID-adabf762-f2f4-43b9-a807-7501f83a9176
Block Pool ID:	BP-919453455-127.0.1.1-1430131312149

Summary

Security is off.
Safemode is off.
1 files and directories, 0 blocks = 1 total filesystem object(s).
Heap Memory used 29.9 MB of 51.02 MB Heap Memory. Max Heap Memory is 966.69 MB.
Non Heap Memory used 27.29 MB of 28.6 MB Committed Non Heap Memory. Max Non Heap Memory is 130 MB.

Configured Capacity:	0 B
DFS Used:	0 B (100%)
Non DFS Used:	0 B
DFS Remaining:	0 B (0%)
Block Pool Used:	0 B (100%)

配置到这里也是可以的，我们同样可以运行 wordcount，也就是我们的 mapreduce 不运行在 yarn 上。如果想让程序运行在 yarn 上,继续下面配置

#####

6.配置 Yarn

1.修改配置文件

修改配置文件 **mapred-site.xml**

编辑文件 `etc/hadoop/mapred-site.xml`，添加下面内容由于 `etc/hadoop` 中没有 `mapred-site.xml`，所以对 `mapred-queues.xml.template` 复制一份

```
cp mapred-site.xml.template mapred-site.xml
```

然后编辑文件 `mapred-site.xml`

添加

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

最后形式：

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

修改配置文件 **yarn-site.xml**

添加如下内容：

```
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
```

```
contributions under the License. See accompanying LICENSE file for details.
-->
<configuration>

<!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

2.启动 yarn

start-yarn.sh

```
aboutyun@ubuntu:~/hadoop-2.7.0/etc/hadoop$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/aboutyun/hadoop-2.7.0/logs/yarn-aboutyun-resourcemanager-ubuntu.out
localhost: starting nodemanager, logging to /home/aboutyun/hadoop-2.7.0/logs/yarn-aboutyun-nodemanager-ubuntu.out
aboutyun@ubuntu:~/hadoop-2.7.0/etc/hadoop$
```

(由于我这里已经配置了环境变量那个，所以在哪个地方都可以运行 **start-yarn.sh**)

如果你没有配置环境变量，则需要进入 `hadoop_home`,执行下面命令

sbin/start-yarn.sh

3.验证

启动 yarn 之后，输入

`http://localhost:8088/`

即可看到下面界面

The screenshot shows the Hadoop cluster management interface. At the top left is the Hadoop logo. The main title is 'All Applications'. On the left is a navigation menu with options like 'Cluster', 'About Nodes', 'Node Labels', 'Applications', and 'Scheduler'. The main content area displays 'Cluster Metrics' and 'Scheduler Metrics'. The 'Cluster Metrics' table shows various resource usage statistics. The 'Scheduler Metrics' table shows scheduler types and resource types. Below these is a table for application entries, which is currently empty, showing 'Showing 0 to 0 of 0 entries'.

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unk
0	0	0	0	0	0 B	8 GB	0 B	0	8	0	1	0	0	0

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocat
Capacity Scheduler	[MEMORY]	<memory:1004, vCores:1>	<memory:8192, vCores:8>

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress
No data available in table									

下一篇 [hadoop2.7 运行 wordcount](#)

遇到问题

问题 1:

Error: Could not find or load main class

```
org.apache.hadoop.hdfs.server.namenode.NameNode
```

解决办法:

在~/hadoop-2.7.0/etc/hadoop/hadoop-env.sh 中添加

```
export HADOOP_COMMON_HOME=~/hadoop-2.7.0
```

重启生效

问题 2:

格式化 Java_home not found

```
bin/hdfs namenode -format
```

在/etc/environment 中添加

```
export JAVA_HOME=/usr/jdk
```

生效

```
source /etc/environment
```

重启[如还不行, 重启]

```
sudo init 6
```

hadoop 入门-第一章 General: 第二节集群配置

问题导读

- 1.说说你对集群配置的认识?
- 2.集群配置的配置项你了解多少?
- 3.下面内容让你对集群的配置有了什么新的认识?



目的

目的 1:

这个文档描述了如何安装配置 hadoop 集群, 从几个节点到上千节点。为了学习 hadoop, 你可能先从单节点入手 (查看 [Single Node Setup](#)). 这里有中文版 [hadoop2.7【单节点】单机、伪分布、分布式安装指导](#)

<http://www.aboutyun.com/thread-12798-1-1.html>

这个文档不包括: hadoop 在安全模式下配置和 HA【高可用配置】, 后面在更新

目的 2:

我们看了很多集群配置文档，你是否静下心来，想集群配置到底是怎么一回事。

准备

1. 安装 Java

2. 下载 hadoop 包

#####

包集合:

[hadoop 家族、strom、spark、Linux、flume 等 jar 包、安装包汇总下载\(持续更新\)](#)

<http://www.aboutyun.com/thread-8178-1-1.html>

#####

安装

安装 hadoop 集群包括：解压包，配置 hadoop，划分主节点和子节点。

集群中可以将 namenode 和 ResourceManager 分布在不同的机器上, 这些称之为 **master**。其它服务例如: Web App Proxy Server 和 MapReduce Job History server，根据负载可以共享设施或则使用专用的机器。

集群其它机器作为 DataNode 和 NodeManager. 这些是 slaves

配置 hadoop 【非安全模式】

hadoop 配置文件被分为两类:

1. 只读默认配置，有下列配置文件

core-default.xml, hdfs-default.xml, yarn-default.xml and mapred-default.xml.

2. 定制配置，有下列配置文件

etc/hadoop/core-site.xml, etc/hadoop/hdfs-site.xml, etc/hadoop/yarn-site.xml and etc/hadoop/mapred-site.xml.

另外你可以配置 hadoop 脚本，在 hadoop 的 bin 目录下，通过 etc/hadoop/hadoop-env.sh 和 etc/hadoop/yarn-env.sh 来指定值。

配置 hadoop 集群需要配置环境变量，Hadoop 守护进程执行以及 Hadoop 守护进程的配置参数。如果没有配置过集群，可能对这个了解不多，具体参考

[hadoop \(2.x\) 以 hadoop2.2 为例完全分布式最新高可靠安装文档](#)

<http://www.aboutyun.com/thread-7684-1-1.html>

hdfs 守护进程:

NameNode, SecondaryNameNode, 和 DataNode

YARN 守护进程:

ResourceManager, NodeManager, 和 WebAppProxy

如果运行 MapReduce , MapReduce Job History Server 也会运行。前提需要配置并开启。对于比较大的集群安装, 他们分别运行在不同的客户端。

配置守护进程环境变量

管理员使用 `etc/hadoop/hadoop-env.sh` , `etc/hadoop/mapred-env.sh` 和 `etc/hadoop/yarn-env.sh` 脚本来定制 hadoop 守护进程变量。

至少需要指定 `JAVA_HOME` , 每个节点都必须指定。

管理员可以配置单独的守护进程, 使用下面的选项。

Daemon	Environment Variable
NameNode	HADOOP_NAMENODE_OPTS
DataNode	HADOOP_DATANODE_OPTS
Secondary NameNode	HADOOP_SECONDARYNAMENODE_OPTS
ResourceManager	YARN_RESOURCEMANAGER_OPTS
NodeManager	YARN_NODEMANAGER_OPTS
WebAppProxy	YARN_PROXYSERVER_OPTS
Map Reduce Job History Server	HADOOP_JOB_HISTORYSERVER_OPTS

例如: 配置 Namenode 使用 `parallelGC`,在 `hadoop-env.sh` 声明如下:

[Bash shell] [纯文本查看](#) [复制代码](#)

[2](#)

```
1 export HADOOP_NAMENODE_OPTS="-XX:+UseParallelGC"
```

查看 `etc/hadoop/hadoop-env.sh` 其它例子

其它可定义的有用参数包括

- HADOOP_PID_DIR - 存储守护进程 id 的文件目录.
- HADOOP_LOG_DIR - 存储守护进程日志文件目录. 日志文件自动创建如果不存在。
- HADOOP_HEAPSIZE / YARN_HEAPSIZE -heapsize 内存最大配置.如果变量为 1000, heap 是 1000MB. 这是为守护进程配置 heap. 默认值 1000M.

在大多数情况下, 应该指定 HADOOP_PID_DIR 和 HADOOP_LOG_DIR, 他们由用户运行 `hadoop` 守护进程所写, 否则可能是潜在攻击。

传统配置 HADOOP_PREFIX 在系统级环境变量配置, 例如一个简单的脚本 `/etc/profile.d:`

[Bash shell] 纯文本查看 复制代码

[?](#)

- 1 HADOOP_PREFIX=/path/to/hadoop
- 2 export HADOOP_PREFIX

Daemon	Environment Variable
ResourceManager	YARN_RESOURCEMANAGER_HEAPSIZE
NodeManager	YARN_NODEMANAGER_HEAPSIZE
WebAppProxy	YARN_PROXYSERVER_HEAPSIZE
Map Reduce Job History Server	HADOOP_JOB_HISTORYSERVER_HEAPSIZE

配置 `hadoop` 守护进程

本节涉及在给定的配置文件中指定的重要参数:

`etc/hadoop/core-site.xml`

Parameter	Value	Notes
<code>fs.defaultFS</code>	NameNode URI	hdfs://host:port/
<code>io.file.buffer.size</code>	131072	读写序列文件缓存大小

- `etc/hadoop/hdfs-site.xml`

- 配置 NameNode:

[彻底了解 namenode](#)

Parameter	Value	Notes
dfs.namenode.name.dir	本地文件系统存储着命令空间和操作日志	如果含有多个目录,是冗余的【多个目录以逗号隔开】
dfs.hosts / dfs.hosts.exclude	列出排除 DataNodes.	如何需要使用这一功能来控制 datanode 的访问
dfs.blocksize	268435456	大文件系统 HDFS blocksize 256MB
dfs.namenode.handler.count	100	更多 NameNode server 线程来处理来自 datanode 的 RPCs 请求

- 配置 DataNode:

Parameter	Value	Notes
dfs.datanode.data.dir	存储 blocks 的本地路径列表,用逗号隔开	这是一个逗号分隔的目录列表,数据将被存储在所有被命名的目录中,通常在不同的设备上。

- etc/hadoop/yarn-site.xml
- 配置 ResourceManager 和 NodeManager:

Parameter	Value	Notes
yarn.acl.enable	true / false	是否启用 ACLs, 默认为 false 不启用
yarn.admin.acl	Admin ACL	admin.acl 设置 YARN 的集群管理员,. 值为*表示任何人都可以. 仅指定的可以访问
yarn.log-aggregation-enable	false	配置启用或则禁用日志

- 配置 ResourceManager:

Parameter	Value	Notes
yarn.resourcemanager.address	ResourceManager host:port 为客户端提交 job.	如果配置 host:port , 会覆盖在 yarn.resourcemanager.hostname 设置的 hostname
yarn.resourcemanager.scheduler.address	ResourceManager 对 ApplicationMaster 暴露的访问地址。 ApplicationMaster 通过该地址向 RM 申请资源、释放资源等。	如果配置 host:port , 会覆盖在 yarn.resourcemanager.hostname 设置的 hostname
yarn.resourcemanager.resource-tracker.address	ResourceManager 对 NodeManager 暴露的地址。NodeManager 通过该地址向 RM 汇报心跳, 领取任务等。	如果配置 host:port , 会覆盖在 yarn.resourcemanager.hostname 设置的 hostname
yarn.resourcemanager.admin.address	ResourceManager 对管理员暴露的访问地址。管理员通过该地址向 RM 发送管理命令等	如果配置 host:port , 会覆盖在 yarn.resourcemanager.hostname 设置的 hostname
yarn.resourcemanager.webapp.address	ResourceManager web-ui host:port (ResourceManager 访问端口)	如果配置 host:port , 会覆盖在 yarn.resourcemanager.hostname 设置的 hostname
yarn.resourcemanager.hostname	ResourceManager 客户端.	host Single hostname that can be set in place of setting all yarn.resourcemanager*address resources. Results in default ports for ResourceManager components.
yarn.resourcemanager.scheduler.class	ResourceManager 调度类 .	CapacityScheduler (recommended), FairScheduler (also recommended), or FifoSchedular
yarn.scheduler.minimum-allocation-mb	单个 container 可申请的最小内存资源量。比如设置为 1024, 则运行 MapRedce 作业时, 每个 Task 最少可申请 1024MB 内存	In MBs
yarn.scheduler.maximum-allocation-mb	单个 container 可申请的最大内存资源量。比如	In MBs

	设置为 3072，则运行 MapReduce 作业时，每个 Task 最多可申请 3072MB 内存。	
yarn.resourcemanager.nodes.include-path /yarn.resourcemanager.nodes.exclude-path	NodeManager 黑白名单。	NodeManager 黑白名单。如果发现若干个 NodeManager 存在问题，比如故障率很高，任务运行失败率高，则可以将之加入黑名单中。注意，这两个配置参数可以动态生效。（调用一个 refresh 命令即可） 默认值：“”

- 配置 NodeManager:

Parameter	Value	Notes
yarn.nodemanager.resource.memory-mb	NodeManager 总的可用物理内存。	定义了资源总量的 nodemanager 可用运行 containers
yarn.nodemanager.vmem-pmem-ratio	每使用 1MB 物理内存，最多可用的虚拟内存数。	每个任务的虚拟内存的使用可能会超过其物理内存的限制，这个比例。通过对 nodemanager 任务使用的虚拟内存总量可能超过物理内存使用的比率。
yarn.nodemanager.local-dirs	中间结果存放位置	这个参数通常会配置多个目录，分摊磁盘 IO 负载。
yarn.nodemanager.log-dirs	日志存放路径	这个参数通常会配置多个目录，分摊磁盘 IO 负载。
yarn.nodemanager.log.retain-seconds	10800	NodeManager 上日志最多存放时间（不启用日志聚集功能时有效）。
yarn.nodemanager.remote-app-log-dir	/logs	当应用程序运行结束后，日志被转移到 HDFS 目录（启用日志聚集功能时有效）。
yarn.nodemanager.remote-app-log-dir-suffix	logs	远程日志目录子目录名称（启用日志聚集功能时有效）。
yarn.nodemanager.aux-services	mapreduce_shuffle	NodeManager 上运行的附属服务。需配置成 mapreduce_shuffle，才可运行 MapReduce 程序。

- 配置 History Server (Needs to be moved elsewhere):

Parameter	Value	Notes
yarn.log-aggregation.retain-seconds	-1	参数解释：在 HDFS 上聚集的日志最多保存多长时间。

		默认值: -1
yarn.log-aggregation.retain-check-interval-seconds	-1	参数解释: 多长时间检查一次日志, 并将满足条件的删除, 如果是 0 或者负数, 则为上一个值的 1/10。 默认值: -1

- etc/hadoop/mapred-site.xml
- 配置 MapReduce Applications:

mapreduce.map.memory.mb

mapreduce.reduce.memory.mb

说明: 这两个参数指定用于 MapReduce 的两个任务 (Map and Reduce task) 的内存大小, 其值应该在 RM 中的最大最小 container 之间。如果没有配置则通过如下简单公式获得:

$\max(\text{MIN_CONTAINER_SIZE}, (\text{Total Available RAM}) / \text{containers})$

一般的 reduce 应该是 map 的 2 倍。注: 这两个值可以在应用启动时通过参数改变;

mapreduce.map.java.opts

mapreduce.reduce.java.opts

说明: 这两个参数主要是为需要运行 JVM 程序 (java、scala 等) 准备的, 通过这两个设置可以向 JVM 中传递参数的, 与内存有关的是, -Xmx, -Xms 等选项。此数值大小, 应该在 AM 中的 map.mb 和 reduce.mb 之间。

Parameter	Value	Notes
mapreduce.framework.name	yarn	执行框架设置为 Hadoop YARN.
mapreduce.map.memory.mb	1536	maps 资源限制
mapreduce.map.java.opts	-Xmx1024M	maps 的 child jvms heap-size
mapreduce.reduce.memory.mb	3072	reduces 资源限制
mapreduce.reduce.java.opts	-Xmx2560M	reduces 的 child jvms heap-size
mapreduce.task.io.sort.mb	512	任务内部排序缓冲区大小
mapreduce.task.io.sort.factor	100	排序文件的时候一次同时最多可并流的个数, 这里设置 100。
mapreduce.reduce.shuffle.parallelcopies	50	reudeuce shuffle 阶段并行传输数据的数量。

- 配置 MapReduce JobHistory Server:

Parameter	Value	Notes
-----------	-------	-------

mapreduce.jobhistory.address	MapReduce JobHistory Server 地址 【host:port】	默认端口号 10020.
mapreduce.jobhistory.webapp.address	MapReduce JobHistory Server Web UI 地址【 host:port】	默认端口号 19888.
mapreduce.jobhistory.intermediate-done-dir	/mr-history/tmp	MapReduce 作业产生的日志存放位置。
mapreduce.jobhistory.done-dir	/mr-history/done	MR JobHistory Server 管理的日志的存放位置。

监控 NodeManagers 健康

hadoop 提供检测一个节点健康状态的机制，管理员可以配置 **NodeManager** 去周期性执行一个脚本来决定一个节点是否健康

管理员可以在这个脚本中做任何的状态监控从而决定此节点是否健康.如果脚本检测节点处于非健康状态，它必须打印必须打印 **error** 开头的标准输出。**NodeManager** 周期性检测，和检测输出，如果脚本输出包含错误（**ERROR**），如上所述，**node** 的状态报告为 **unhealthy**，该节点被资源管理器列为黑名单。该节点不会被分配任务。

尽管如此，如果资源管理器恢复健康状态，脚本继续运行，并且会自动被移除黑名单。节点脚本输出，提供给管理，在界面上节点不健康，以及在界面上显示节点健康时间。

下面参数可以被用来控制节点是否健康监视脚本在 `etc/hadoop/yarn-site.xml`。

参数	值	说明
yarn.nodemanager.health-checker.script.path	Node health script	节点健康状态监测脚本
yarn.nodemanager.health-checker.script.opts	Node health script options	脚本的选项来检查节点的健康状态
yarn.nodemanager.health-checker.script.interval-ms	Node health script interval	运行健康脚本的时间间隔。
yarn.nodemanager.health-checker.script.timeout-ms	Node health script timeout interval	健康脚本执行超时

如果仅本地磁盘故障，检测脚本不会给出 **error**。**NodeManager** 有能力定期检测磁盘本地磁盘健康状态（特别是 `nodemanager-local-dirs` 和 `nodemanager-log-dirs`）。当目录损坏数达到配置的阈值（`yarn.nodemanager.disk-health-checker.min-healthy-disks` 配置的）之后整个节点就会被标记为不正常的。同时这些信息也会上报给资源管理器(resource manager),检测脚本也会检测启动盘。

Slaves File

列出所有 slave hostnames or IP 地址在 `etc/hadoop/slaves` 文件, 一行一个。Helper 脚本 (described below) 使用 `etc/hadoop/slaves` 文件运行命令在许多客户端。它不需要任何基于 Java 的 hadoop 配置, 为了使用此功能, ssh 必须连接信任(`passphraseless` 或则其它方法,比如 Kerberos)的账户运行 hadoop

Hadoop 机架感知

许多 hadoop 具有感知功能和利用网络拓扑结构的性能和安全性。hadoop 守护进程调用管理员配置模块获取集群 slaves 信息。更多信息查看 [Rack Awareness](#) , 开始启动 hdfs 之前, 推荐配置集群机架感应。

日志

hadoop 通过 Apache Commons 的日志框架使用 `Apache log4j` 作为日志。编辑 `etc/hadoop/log4j.properties` 文件定制 hadoop 守护进程日志配置比如 `log` 格式等

hadoop 集群操作

所有的配置完成, 分部署文件 `HADOOP_CONF_DIR` 目录分发到所有机器, 他们在所有机器上路径是相同的。建议 hdfs 和 yarn 使用独立的用户。hdfs 执行 hdfs 用户, yarn 使用 yarn 账户。

hadoop 启动

启动 hadoop 集群, 需要启动 hdfs 和 yarn

第一次启动 hdfs, 必须要格式化, 格式化分布式文件系统作为 HDFS:

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [hdfs]$ $HADOOP_PREFIX/bin/hdfs namenode -format <cluster_name>
```

启动 hdfs

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [hdfs]$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR
1 --script hdfs start namenode
```

启动 datanode

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [hdfs]$ $HADOOP_PREFIX/sbin/hadoop-daemons.sh --config $HADOOP_CONF_DIR  
1 --script hdfs start datanode
```

如果 `etc/hadoop/slaves` 和 `ssh` 配置了相互访问 (see [Single Node Setup](#)), 所有 `hdfs` 可以使用下面命令

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [hdfs]$ $HADOOP_PREFIX/sbin/start-dfs.sh
```

启动 `yarn` 用下面命令, 在指定的 `ResourceManager` 作为 `yarn`

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [yarn]$ $HADOOP_YARN_HOME/sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR  
1 start resourcemanager
```

运行下面命令在指定的客户端作为 `yarn` 启动 `NodeManager`

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [yarn]$ $HADOOP_YARN_HOME/sbin/yarn-daemons.sh --config $HADOOP_CONF_DIR  
1 start nodemanager
```

启动独立 `WebAppProxy server`. 运行 `WebAppProxy server` 作为 `yarn`. 如果多台服务器使用负载均衡, 则每台都需运行

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [yarn]$ $HADOOP_YARN_HOME/sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR  
1 start proxyserver
```

如果 `etc/hadoop/slaves` 和 `ssh` 配置, 启动脚本如下:

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [yarn]$ $HADOOP_PREFIX/sbin/start-yarn.sh
```

启动 MapReduce JobHistory Server

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [mapred]$ $HADOOP_PREFIX/sbin/mr-jobhistory-daemon.sh --config  
1 $HADOOP_CONF_DIR start historyserver
```

hadoop 停止

停止 namenode，用下面命令

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [hdfs]$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR  
1 --script hdfs stop namenode
```

停止 datanode，用下面命令

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [hdfs]$ $HADOOP_PREFIX/sbin/hadoop-daemons.sh --config $HADOOP_CONF_DIR  
1 --script hdfs stop datanode
```

如果 etc/hadoop/slaves 和 ssh 配置相互访问，用下面脚本停止

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [hdfs]$ $HADOOP_PREFIX/sbin/stop-dfs.sh
```

停止 ResourceManager 用下面命令

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 [yarn]$ $HADOOP_YARN_HOME/sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR  
1 stop resourcemanager
```

运行脚本停止 NodeManager

[Bash shell] [纯文本查看](#) [复制代码](#)

2

```
1 [yarn]$ $HADOOP_YARN_HOME/sbin/yarn-daemons.sh --config $HADOOP_CONF_DIR  
1 stop nodemanager
```

如果 `etc/hadoop/slaves` 和 `ssh` 配置相互访问，则运行下面脚本

[Bash shell] [纯文本查看](#) [复制代码](#)

2

```
1 [yarn]$ $HADOOP_PREFIX/sbin/stop-yarn.sh
```

停止 WebAppProxy server。运行 WebAppProxy，如果多台服务器使用负载均衡，则每一个都运行。

[Bash shell] [纯文本查看](#) [复制代码](#)

2

```
1 [yarn]$ $HADOOP_YARN_HOME/sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR  
1 stop proxyserver
```

停止 MapReduce JobHistory Server，使用下面命令

[Bash shell] [纯文本查看](#) [复制代码](#)

2

```
1 [mapred]$ $HADOOP_PREFIX/sbin/mr-jobhistory-daemon.sh --config  
1 $HADOOP_CONF_DIR stop historyserver
```

Web Interfaces

Daemon	Web Interface	Notes
NameNode	http://nn_host:port/	Default HTTP port is 50070.
ResourceManager	http://rm_host:port/	Default HTTP port is 8088.
MapReduce JobHistory Server	http://jhs_host:port/	Default HTTP port is 19888.

hadoop 入门-第一章 General: 第三节 Hadoop 初级入门之命令指南

问题导读

- 1.hadoop daemonlog 管理员命令的作用是什么?
- 2.hadoop 如何运行一个类, 如何运行一个 jar 包?
- 3.hadoop archive 的作用是什么?



概述

hadoop 命令被 bin/hadoop 脚本调用. 运行 hadoop 脚本不带任何命令将打印命令相关描述。

Usage: hadoop [--config confdir] [--loglevel loglevel] [COMMAND] [GENERIC_OPTIONS] [COMMAND_OPTIONS]

FIELD	Description
--config confdir	覆盖默认配置文件目录。默认路径\${HADOOP_HOME}/conf.
--loglevel loglevel	覆盖日志级别. 有效日志级别为 FATAL, ERROR, WARN, INFO, DEBUG, 和 TRACE. 默认是 INFO.
GENERIC_OPTIONS	多项命令共同支持的选项
COMMAND_OPTIONS	hadoop 子项目文档描述了 hadoop 命令的选项。hdfs 和 YARN 在其它文档那个

Hadoop 通用选项

Many subcommands honor a common set of configuration options to alter their behavior:

GENERIC_OPTION	Description
-archives <comma separated list of	Specify comma separated archives to be unarchived on the compute

archives>	machines. Applies only to job.
-conf <configuration file>	指定应用程序配置文件
-D <property>=<value>	使用一个给定的属性值
-files <comma separated list of files>	指定文件复制到 mapredue 集群。仅适用于 job.
-jt <local> or <resourcemanager:port>	指定 ResourceManager. 仅适用于 job.
-libjars <comma seperated list of jars>	指定 jar 文件包括所在 classpath. 仅适用于 job.

Hadoop 通用命令

所有这些命令都是从 Hadoop 的 shell 命令执行。他们被分成用户命令和管理员命令【 User Commands 和 Administration Commands.】

用户命令

用于 Hadoop 集群用户命令。

archive

创建一个 Hadoop 档案，更多信息查看 [Hadoop Archives Guide](#).

checknative

用法: `hadoop checknative [-a] [-h]`

COMMAND_OPTION	Description
-a	检查所有库
-h	打印帮助

这个命令检查 Hadoop 本地代码的可用性。更多信息查看 [#NativeLibraries.html](#)。默认，此命令只检查 libhadoop 的可用性。

classpath

用法: `hadoop classpath [--glob |--jar <path> |-h |--help]`

COMMAND_OPTION	Description
<code>--glob</code>	通配符
<code>--jar path</code>	jar 路径
<code>-h, --help</code>	打印帮助

打印 `classpath` 需要 Hadoop 的 `jar` 和需要的库类路径。如果不带参数调用，然后打印的命令脚本设置 `classpath`。这可能包含通配符

credential

用法: `hadoop credential <subcommand> [options]`

COMMAND_OPTION	Description
<code>create alias [-provider provider-path]</code>	提示证书被存储为指定别名的用户。如果没有 <code>-provider</code> 选项的话，那么将会默认使用 <code>core-site.xml</code> 文件中 <code>hadoop.security.credential.provider.path</code> 项对应的值。
<code>delete alias [-provider provider-path] [-f]</code>	删除与所提供的别名对应的证书文件。如果没有 <code>-provider</code> 选项的话，那么将会默认使用 <code>core-site.xml</code> 文件中 <code>hadoop.security.credential.provider.path</code> 项对应的值。这项操作需要通过用户的确认，除非使用了 <code>-f</code> 选项。
<code>list [-provider provider-path]</code>	列出所有的证书别名。如果没有 <code>-provider</code> 选项的话，那么将会默认使用 <code>core-site.xml</code> 文件中 <code>hadoop.security.credential.provider.path</code> 项对应的值。

该命令在凭证提供者内部管理凭证（`credentials`），密码（`passwords`）和秘密（`secrets`）。

Hadoop 的 `CredentialProvider` API 支持应用程序拆分，并且要求拆分后的应用如何储存所需的密码（`passwords`）和秘密（`secrets`）。为了指明一个 `Provider` 的位置和类型，需要在 `core-site.xml` 添加 `hadoop.security.credential.provider.path` 配置项，或者通过指令中 `-provider` 命令选项进行设置。`Provider` 路径是一串以逗号分割的 URL 字符串。这些字符串会说明 `Provider` 的类型和位置，举个例子：

[Bash shell] 纯文本查看 复制代码

[?](#)

```
luser:/// , jceks://file/tmp/test.jceks, jceks://hdfs@nn1.example.com/my/path/test.jceks
```

指示当前用户的凭证，需要通过 `User Provider` 咨询。存储在本地文件系统的文件 `/tmp/test.jceks` 是一个 `Java Keystore Provider`，相应的存储在 `hdfs` 上的文件 `nn1.example.com/my/path/test.jcek` 也是一个 `Java Keystore Provider`。

当使用 `credential` 命令时，它通常要提供密码（`password`）或秘密（`secret`）给一个特定的凭证存储 `provider`。为了清晰的表明要显示使用哪个 `provider` 存储，可以在命令中使用 `-provider` 选项。否则，给定多个 `provider` 的时候，则使用的哥非持久的 `provider`，这可能不是你预期的

例如：

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
hadoop credential list -provider jceks://file/tmp/test.jceks
```

distcp

递归的拷贝文件或者目录。更多信息查看 [Hadoop DistCp Guide](#).

fs

这个命令在文档 [File System Shell Guide](#) 。 和 `hdfs` 脚本的 `dfs` 类似

jar

用法: `hadoop jar <jar> [mainClass] args...`

运行一个 `jar` 文件使用 `yarn` 启动 `yarn` 应用程序

key

通过 `KeyProvider` 管理密钥

trace

查看和修改 Hadoop 跟踪（`tracing`）设置。查看：[跟踪（tracing）指南](#)。

[Tracing Guide](#).

version

用法: `hadoop version`

打印版本

CLASSNAME

用法: `hadoop CLASSNAME`

运行一个类

管理员命令

集群管理员命令

daemonlog

用法:

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1hadoop daemonlog -getlevel <host:httpport> <classname>
```

```
2hadoop daemonlog -setlevel <host:httpport> <classname> <level>
```

COMMAND_OPTION	Description
<code>-getlevel host:httpportclassname</code>	打印运行在<host:port>的守护进程的日志级别。这个命令内部会连接 <code>http://<host:port>/logLevel?log=<name></code>
<code>-setlevel host:httpportclassname level</code>	设置运行在<host:port>的守护进程的日志级别。这个命令内部会连接 <code>http://<host:port>/logLevel?log=<name></code>

设置或获取指定后台进程的日志级别

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1Example: $ bin/hadoop daemonlog -setlevel 127.0.0.1:50070  
1org.apache.hadoop.hdfs.server.namenode.NameNode DEBUG
```

hadoop 入门-第一章 General: 第四节文件系统 shell

问题导读

1.Hadoop 文件系统 shell 与 Linux shell 有哪些相似之处?

2.如何改变文件所属组?

3.如何改变 hdfs 的文件权限?

4.如何查找 hdfs 文件，并且不区分大小写？



概述

文件系统 (FS) shell 包括各种类似的命令直接与 Hadoop Distributed File System (HDFS)交互。hadoop 也支持其它文件系统,比如 Local FS, HFTP FS, S3 FS, 和 其它的。FS shell 被下面调用:

[Bash shell] 纯文本查看 复制代码

[?](#)

```
1 bin/hadoop fs <args>
```

所有的 FS shell 命令带有 URIs 路径参数。The URI 格式是://authority/path。对 HDFS 文件系统, scheme 是 hdfs。其中 scheme 和 authority 参数都是可选的

如果没有指定, 在文件中使用默认 scheme.一个 hdfs 文件或则目录比如 /parent/child, 可以是

hdfs://namenodehost/parent/child 或则简化为/parent/child (默认配置设置成指向 hdfs://namenodehost) .大多数 FS shell 命令对应 Unix 命令.每个命令都有不同的描述。将错误信息发送到标准错误输出和输出发送到 stdout。

appendToFile 【添加文件】

用法: `hadoop fs -appendToFile <localsrc> ... <dst>`添加单个 src,或则多个 srcs 从本地文件系统到目标文件系统。从标准输入读取并追加到目标文件系统。

- `hadoop fs -appendToFile localfile /user/hadoop/hadoopfile`
- `hadoop fs -appendToFile localfile1 localfile2 /user/hadoop/hadoopfile`
- `hadoop fs -appendToFile localfile hdfs://nn.example.com/hadoop/hadoopfile`
- `hadoop fs -appendToFile - hdfs://nn.example.com/hadoop/hadoopfile` Reads the input from stdin.

返回代码:

返回 0 成功返回 1 错误

cat

用法: `hadoop fs -cat URI [URI ...]`

将路径指定文件的内容输出到 stdout

例子:

- `hadoop fs -cat hdfs://nn1.example.com/file1 hdfs://nn2.example.com/file2`
- `hadoop fs -cat file:///file3 /user/hadoop/file4`

返回代码:

返回 0 成功返回 1 错误

checksum

用法: `hadoop fs -checksum URI`

返回 checksum 文件信息

例子:

- `hadoop fs -checksum hdfs://nn1.example.com/file1`
- `hadoop fs -checksum file:///etc/hosts`

chgrp

用法: `hadoop fs -chgrp [-R] GROUP URI [URI ...]`

改变文件所属组, 必须是文件所有者或则超级用户, 更多信息在 [Permissions Guide](#).

选项

- 使用-R 将使改变在目录结构下递归进行

chmod

用法: `hadoop fs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI ...]`

更改文件的权限, 使用-R 将使改变在目录结构下递归进行。必须是文件所有者或则超级用户, 更多信息在 [Permissions Guide](#).

选项

- 使用-R 将使改变在目录结构下递归进行。

chown

用法: `hadoop fs -chown [-R] [OWNER][:[GROUP]] URI [URI]`

更改文件的所有者, 使用-R 将使改变在目录结构下递归进行。必须是文件所有者或则超级用户, 更多信息在 [Permissions Guide](#).

选项

- 使用-R 将使改变在目录结构下递归进行。

copyFromLocal

用法: `hadoop fs -copyFromLocal <localsrc> URI`

类似 `put` 命令, 需要指出的是这个限制是本地文件

选项:

- -f 选项会重写已存在的目标文件

copyToLocal

用法: `hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localdst>`

与 `get` 命令类似, 除了限定目标路径是一个本地文件外。

count

用法: `hadoop fs -count [-q] [-h] [-v] <paths>` 统计目录个数, 文件和目录下文件的大小。输出列: DIR_COUNT, FILE_COUNT, CONTENT_SIZE, PATHNAME

【目录个数, 文件个数, 总大小, 路径名称】

输出列带有 `-count -q` 是: QUOTA, REMAINING_QUOTA, SPACE_QUOTA, REMAINING_SPACE_QUOTA, DIR_COUNT, FILE_COUNT, CONTENT_SIZE, PATHNAME

【配置, 其余指标, 空间配额, 剩余空间定额, 目录个数, 文件个数, 总大小, 路径名称】

The `-h` 选项, `size` 可读模式。

The `-v` 选项显示一个标题行。

Example:

- `hadoop fs -count hdfs://nn1.example.com/file1 hdfs://nn2.example.com/file2`
- `hadoop fs -count -q hdfs://nn1.example.com/file1`
- `hadoop fs -count -q -h hdfs://nn1.example.com/file1`
- `hdfs dfs -count -q -h -v hdfs://nn1.example.com/file1`

返回代码:

返回 0 成功返回 1 错误

```
aboutyun123@aboutyun:~$ hadoop fs -count -q -h /user/aboutyun123/input/
none      inf      none     inf      1        30      76.7 K /user/aboutyun123/input
aboutyun123@aboutyun:~$ hadoop fs -count -q -h /user/aboutyun123
none      inf      none     inf      3        32      77.0 K /user/aboutyun123
aboutyun123@aboutyun:~$ hadoop fs -ls /user/aboutyun123
Found 2 items
drwxr-xr-x - aboutyun123 supergroup 0 2015-07-08 00:31 /user/aboutyun123/input
drwxr-xr-x - aboutyun123 supergroup 0 2015-07-08 00:39 /user/aboutyun123/output
aboutyun123@aboutyun:~$ hadoop fs -ls /user
Found 1 items
drwxr-xr-x - aboutyun123 supergroup 0 2015-07-08 00:39 /user/aboutyun123
aboutyun123@aboutyun:~$ hadoop fs -count -q -h /user
none      inf      none     inf      4        32      77.0 K /user
aboutyun123@aboutyun:~$
```

cp

用法: `hadoop fs -cp [-f] [-p | -p[topax]] URI [URI ...] <dest>` 复制文件, 这个命令允许复制多个文件到一个目录。

``raw.*'` 命名空间扩展属性被保留

(1) 源文件和目标文件支持他们 (仅 `hdfs`)

(2) 所有的源文件和目标文件路径在 `/.reserved/raw` 目录结构下。

决定是否使用 `raw.*`命名空间扩展属性依赖于 `-P` 选项

选项:

- `-f` 选项如果文件已经存在将会被重写.
- `-p` 选项保存文件属性 [`topx`] (`timestamps, ownership, permission, ACL, XAttr`). 如果指定 `-p` 没有参数, 保存 `timestamps, ownership, permission`. 如果指定 `-pa`, 保留权限 因为 `ACL` 是一个权限的超级组。确定是否保存 `raw` 命名空间属性取决于是否使用 `-p` 决定

例子:

- `hadoop fs -cp /user/hadoop/file1 /user/hadoop/file2`
- `hadoop fs -cp /user/hadoop/file1 /user/hadoop/file2 /user/hadoop/dir`

返回代码:

返回 0 成功返回 1 错误

createSnapshot

查看 [HDFS Snapshots Guide](#).

deleteSnapshot

查看 [HDFS Snapshots Guide](#).

df 【查看还剩多少 hdfs 空间】

用法: `hadoop fs -df [-h] URI [URI ...]`

显示剩余空间

选项:

- `-h` 选项会让人更加易读 (比如 `64.0m` 代替 `67108864`)

Example:

- `hadoop dfs -df /user/hadoop/dir1`

```
aboutyun123@aboutyun:~$ hadoop fs -df /
Filesystem              Size      Used    Available  Use%
hdfs://localhost:9000  20334034944  1019904  12705988608    0%
aboutyun123@aboutyun:~$ hadoop fs -df -h /
Filesystem              Size      Used    Available  Use%
hdfs://localhost:9000  18.9 G   996 K    11.8 G     0%
```

du

用法: `hadoop fs -du [-s] [-h] URI [URI ...]`显示给定目录的文件大小及包含的目录, 如果只有文件只显示文件的大小
选项:

- `-s` 选项汇总文件的长度, 而不是现实单个文件.
- `-h` 选项显示格式更加易读 (例如 `64.0m` 代替 `67108864`)

例子:

- `hadoop fs -du /user/hadoop/dir1 /user/hadoop/file1 hdfs://nn.example.com/user/hadoop/dir1`

返回代码:

返回 `0` 成功返回 `1` 错误

```
aboutyun123@aboutyun:~$ hadoop fs -du -s /
702582 /
aboutyun123@aboutyun:~$ hadoop fs -du -s /user
78822 /user
aboutyun123@aboutyun:~$ hadoop fs -du -s -h /user
77.0 K /user
```

du

用法: `hadoop fs -dus <args>`

显示统计文件长度

注意:这个命令已被启用, `hadoop fs -du -s` 即可

expunge

用法: `hadoop fs -expunge`

清空垃圾回收站, 涉及 [HDFS Architecture Guide](#) 更多信息查看回收站特点

find

用法: `hadoop fs -find <path> ... <expression> ...`查找与指定表达式匹配的所有文件, 并将选定的操作应用于它们。如果没有指定路径, 则默认查找当前目录。如果没有指定表达式默认`-print`

下面主要表达式:

- `-name` 模式
• `-iname` 模式

如果

值为 `TRUE` 如果文件基本名匹配模式使用标准的文件系统组合。如果使用`-iname` 匹配不区分大小写。

- `-print`
• `-print0Always`
值为 `TRUE`. 当前路径被写至标准输出。如果使用 `-print0` 表达式, `ASCII NULL` 字符是追加的.

下面操作:

- expression -a expression
- expression -and expression
- expression expression

and 运算符连接两个表达式,如果两个字表达式返回 **true**,则返回 **true**.由两个表达式的并置暗示,所以不需要明确指定。如果第一个失败,则不会应用第二个表达式。

例子:

```
hadoop fs -find / -name test -print
```

返回代码:

返回 **0** 成功返回 **1** 错误

```
aboutyun123@aboutyun:~$ hadoop fs -find /user/aboutyun123/input -iname yarn* -print0
/user/aboutyun123/input/yarn-env.cmd/user/aboutyun123/input/yarn-env.sh/user/aboutyun123/input/yarn-site.xmlaboutyun123@aboutyun:~$ h
adoop fs -find /user/aboutyun123/input -iname yarn* -print
/user/aboutyun123/input/yarn-env.cmd
/user/aboutyun123/input/yarn-env.sh
/user/aboutyun123/input/yarn-site.xml
```

get

用法: `hadoop fs -get [-ignorecrc] [-crc] <src> <localdst>`复制文件到本地文件。

复制文件到本地文件系统。【CRC 校验失败的文件复制带有 `-ignorecrc` 选项（如翻译有误欢迎指正）】

Files that fail the CRC check may be copied with the `-ignorecrc` option.

文件 CRC 可以复制使用 CRC 选项。

例子:

- `hadoop fs -get /user/hadoop/file localfile`
- `hadoop fs -get hdfs://nn.example.com/user/hadoop/file localfile`

返回代码:

返回 **0** 成功返回 **1** 错误

getfacl

用法: `hadoop fs -getfacl [-R] <path>`

显示访问控制列表 (ACL) 的文件和目录。如果一个目录有默认的 ACL, `getfacl` 也显示默认的 ACL。

选项:

- `-R`: 递归目录和列出所有文件的 ACLs.
- `path`: 文件或目录列表。

例子:

- `hadoop fs -getfacl /file`
- `hadoop fs -getfacl -R /dir`

返回代码:

返回 0 成功返回 非 0 错误

```
aboutyun123@aboutyun:/usr/hadoop-2.7.0/sbin$ hadoop fs -getfacl /test
# file: /test
# owner: aboutyun123
# group: supergroup
user::rw-
group::r--
other::r--
```

getfattr

用法: `hadoop fs -getfattr [-R] -n name | -d [-e en] <path>`

显示文件和目录扩展属性名字和值[如果有的话]

选项:

- `-R`: 递归显示文件和目录属性.
- `-n name`: Dump the named extended attribute value.
- `-d`: Dump all extended attribute values associated with pathname.
- `-e encoding`: 检索后的值进行编码。有效的编码是 "text", "hex", and "base64". 值编码作为文本字符串是用双引号括起来的 (""),

值编码作为 16 进制和 64 进制, 前缀分别为 0x 和 0s

- `path`: 文件或则目录

例子:

- `hadoop fs -getfattr -d /file`
- `hadoop fs -getfattr -R -n user.myAttr /dir`

返回代码:

返回 0 成功返回 非 0 错误

getmerge

用法: `hadoop fs -getmerge <src> <localdst> [addnl]`

源目录和目标文件作为输入和连接文件合并到本地目标文件。`addnl` 选项可以设置在文件末尾添加一个换行符。

help

用法: `hadoop fs -help`

返回使用输出。

ls

用法: `hadoop fs -ls [-d] [-h] [-R] [-t] [-S] [-r] [-u] <args>`

选项:

- `-d`: 目录被列为纯文件。
- `-h`: 文件格式变为易读 (例如 `67108864` 显示 `64.0m`)。
- `-R`: 递归子目录列表中。
- `-t`: 按修改时间排序输出 (最近一次)。
- `-S`: 按文件大小排序输出。
- `-r`: 倒序排序
- `-u`: 对使用时间显示和排序而不是修改时间

文件返回下面信息:

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

1	permissions	number_of_replicas	userid	groupid	filesize
2	权限		副本数		
	期		修改时间		文件名

目录返回下面信息

权限	用户	所属组	修改日期	修改时间	目录名
----	----	-----	------	------	-----

目录内的文件默认按文件名排序

例子:

- `hadoop fs -ls /user/hadoop/file1`

退出代码:

返回 `0` 成功, 返回 `-1` 错误

lsr

用法: `hadoop fs -lsr <args>`

ls 递归

注意: 这个命令被启用的, 替换为 `hadoop fs -ls -R`

mkdir

用法: `hadoop fs -mkdir [-p] <paths>`

以 URI 的路径作为参数并创建目录。

选项:

- `-p` 选项与 Linux `-p` 功能一样, 会创建父目录

例子:

- `hadoop fs -mkdir /user/hadoop/dir1 /user/hadoop/dir2`
- `hadoop fs -mkdir hdfs://nn1.example.com/user/hadoop/dir hdfs://nn2.example.com/user/hadoop/dir`

退出代码:

返回 0 成功, -1 错误

moveFromLocal

用法: `hadoop fs -moveFromLocal <localsrc> <dst>`

类似 `put` 命令, 但是它是本地源文件复制后被删除

moveToLocal

用法: `hadoop fs -moveToLocal [-crc] <src> <dst>`

显示 "Not implemented yet" 消息

mv

用法: `hadoop fs -mv URI [URI ...] <dest>` 移动文件, 这个命令允许移动多个文件到某个目录

例子:

- `hadoop fs -mv /user/hadoop/file1 /user/hadoop/file2`
- `hadoop fs -mv hdfs://nn.example.com/file1 hdfs://nn.example.com/file2 hdfs://nn.example.com/file3 hdfs://nn.example.com/dir1`

退出代码:

返回 0 成功, -1 错误

put

用法: `hadoop fs -put <localsrc> ... <dst>`

复制单个或则多个源文件到目标系统文件。从 `stdin` 读取输入并写入到目标文件系统。

- `hadoop fs -put localfile /user/hadoop/hadoopfile`
- `hadoop fs -put localfile1 localfile2 /user/hadoop/hadoopdir`
- `hadoop fs -put localfile hdfs://nn.example.com/hadoop/hadoopfile`
- `hadoop fs -put - hdfs://nn.example.com/hadoop/hadoopfile` 从 `stdin` 读取输入。

退出代码:

返回 0 成功, -1 错误

renameSnapshot

See HDFS Snapshots Guide.

rm

用法: `hadoop fs -rm [-f] [-r |-R] [-skipTrash] URI [URI ...]`

删除指定的参数文件。

选项:

- `-f` 选项 如果该文件不存在, 则该选项将不显示诊断信息或修改退出状态以反映错误。
- `-R` 选项递归删除目录下任何内容
- `-r` 与 `-R` 效果一样
- `-skipTrash` 选项绕过垃圾回收器, 如果启用, 将会立即删除指定文件。这是非常有用对于超过配额的目录

例子:

- `hadoop fs -rm hdfs://nn.example.com/file /user/hadoop/emptydir`

退出代码:

返回 0 成功, -1 错误

rmdir

用法: `hadoop fs -rmdir [--ignore-fail-on-non-empty] URI [URI ...]`

删除目录

选项:

- `--ignore-fail-on-non-empty`: 当使用通配符, 一个目录还包含文件, 不会失败.

例子:

- `hadoop fs -rmdir /user/hadoop/emptydir`

rmr

用法: `hadoop fs -rmr [-skipTrash] URI [URI ...]`

递归删除

说明:这个命令被弃用了, 而是使用 `hadoop fs -rm -r`

setfacl

用法: `hadoop fs -setfacl [-R] [-b | -k -m | -x <acl_spec> <path>] [--set <acl_spec> <path>]`

设置访问控制列表 (ACL) 的文件和目录。

选项:

- `-b`: 移除所有除了基本的 ACL 条目。用户、组和其他的条目被保留为与权限位的兼容性。
- `-k`: 删除默认的 ACL。
- `-R`: 递归应用于所有文件和目录的操作。
- `-m`: 修改 ACL。新的项目添加到 ACL, 并保留现有的条目。
- `-x`: 删除指定的 ACL 条目。其他保留 ACL 条目。
- `--set`: 完全替换 ACL, 丢弃所有现有的条目。`acl_spec` 必须包括用户, 组, 和其他有权限位的兼容性。
- `acl_spec`: 逗号分隔的 ACL 条目列表。
- `path`: 修改文件或目录。

例子:

- `hadoop fs -setfacl -m user:hadoop:rw- /file`
- `hadoop fs -setfacl -x user:hadoop /file`
- `hadoop fs -setfacl -b /file`
- `hadoop fs -setfacl -k /dir`
- `hadoop fs -setfacl --set user::rw-,user:hadoop:rw-,group::r--,other::r-- /file`
- `hadoop fs -setfacl -R -m user:hadoop:r-x /dir`
- `hadoop fs -setfacl -m default:user:hadoop:r-x /dir`

退出代码:

返回 0 成功, 非 0 错误

以上需要开启 `acl`:

开启 `acls`, 配置 `hdfs-site.xml`

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 vi etc/hadoop/hdfs-site.xml
2 <property>
3     <name>dfs.namenode.acls.enabled</name>
4     <value>>true</value>
5 </property>
```

setfattr

用法: `hadoop fs -setfattr -n name [-v value] | -x name <path>`

设置一个文件或目录的扩展属性名和值。

选项:

-b: 移除所有的条目除了基本的 ACL 条目。用户、组和其他的条目被保留为与权限位的兼容性。

-n name: 扩展属性名。

-v value: 扩展属性值。有三种不同编码值, 如果该参数是用双引号括起来的, 则该值是引号内的字符串。如果参数是前缀 0x 或 0X, 然后作为一个十六进制数。如果参数从 0 或 0, 然后作为一个 base64 编码。

-x name: 移除所有属性值

path: 文件或则路径

例子:

- `hadoop fs -setfattr -n user.myAttr -v myValue /file`
- `hadoop fs -setfattr -n user.noValue /file`
- `hadoop fs -setfattr -x user.myAttr /file`

退出代码:

返回 0 成功, 非 0 错误

setrep

用法: `hadoop fs -setrep [-R] [-w] <numReplicas> <path>`

更改文件的备份。如果是一个目录, 会递归改变目录下文件的备份。

选项:

-w 标识, 要求备份完成, 这可能需要很长时间。

-R 标识, 是为了兼容, 没有实际效果

例子:

- `hadoop fs -setrep -w 3 /user/hadoop/dir1`

退出代码:

返回 0 成功, 非 0 错误

stat

用法: `hadoop fs -stat [format] <path> ...`按指定格式打印文件/目录的打印统计。

格式接受文件块 (%b), 类型 (%F), group 拥有者 (%g), 名字 (%n), block size (%o), replication (%r), 用户拥有者 (%u), 修改日期 (%y, %Y). %y 显示 UTC 日期如 "yyyy-MM-dd HH:mm:ss" 和 %Y 1970 年 1 月 1 日以来显示毫秒 UTC. 如果没有指定, 默认使用 %y.

例子:

- `hadoop fs -stat "%F %u:%g %b %y %n" /file`

退出代码:

返回 0 成功

返回-1 错误

tail

用法: `hadoop fs -tail [-f] URI`

显示文件内容，最后千字节的文件发送到 `stdout`，

选项:

-f 选项将输出附加数据随着文件的生长，如同 Unix

例子:

- `hadoop fs -tail pathname`

退出代码:

返回 0 成功

返回-1 错误

test

用法: `hadoop fs -test -[defsz] URI`

选项:

-d:如果路径是一个目录，返回 0

-e:如果路径已经存在，返回 0

-f: 如果路径是一个文件，返回 0

-s:如果路径不是空，返回 0

-z:如果文件长度为 0，返回 0

例子:

- `hadoop fs -test -e filename`

text

用法: `hadoop fs -text <src>`

一个源文件，以文本格式输出文件。允许的格式是 zip 和 `textrecordinputstream`。

touchz

用法: `hadoop fs -touchz URI [URI ...]`

创建一个零长度的文件。

例子:

- `hadoop fs -touchz pathname`

退出代码: 返回 0 成功, -1error

truncate

用法: `hadoop fs -truncate [-w] <length> <paths>`

截断指定文件模式指定的长度匹配的所有文件。

选项:

`-w` 选项需要等待命令完成块恢复。如果没有 `-w` 选项, 在恢复的过程中可能是未闭合的

例子:

- `hadoop fs -truncate 55 /user/hadoop/file1 /user/hadoop/file2`
- `hadoop fs -truncate -w 127 hdfs://nn1.example.com/user/hadoop/file1`

usage

用法: `hadoop fs -usage command`

返回单个命令的帮助。

hadoop 入门-第一章 General: 第五节 hadoop 的兼容性说明

问题导读

- 1.根据下文 hadoop2.1.1 客户端与 hadoop2.4.0 集群是否可以通信?**
- 2.hadoop2.4.0 客户端与 hadoop2.3.0 集群【服务器】是否可以通信?**
- 3.升级后集群, hdfs,mapreduce, yarn 程序是否需要修改?**
- 4.hadoop 单独组建是否可以升级?**
- 5.hadoop 主版本升级后, 如 flume 是否受影响?**



目的

这个文档介绍了 Apache Hadoop 项目的兼容性，在 hadoop 发布版之间的兼容性，影响了 hadoop 开发者，hadoop 相关项目，终端用户。

描述了 hadoop 终端用户及相关项目的影响

Hadoop 开发人员采用的政策为了适用，不兼容的改变是允许的

兼容类型

Java API

- hadoop 接口和类都是有注释的，描述了使用者和稳定性，来保持与以前的版本兼容。更多细节查看 [Hadoop Interface Classification](#)
- InterfaceAudience: 描述了什么类型是允许改变的，可能的值是 Stable, Evolving, Unstable, 和 Deprecated.

用例

- Public-Stable API 兼容性，从而确保最终用户的方案和下游项目继续不加修改工作。
- LimitedPrivate-Stable API 兼容性，需要允许跨越小版本单个组件升级。
- Private-Stable API 兼容性，需要滚动升级

Policy

Public-Stable APIs 在一个主要版本发布前，从版本中移除，必须被弃用。

LimitedPrivate-Stable APIs 在主版本中可以改变，但是仅在发布的主版本之内。

Private-Stable APIs 在主版本中可以改变，但是仅在发布的主版本之内。

没有注解的类为"Private".类成员没有注释继承封闭类的注释。

注意：从原始文件生成 API 需要兼容滚动升级，更多细节 查看 [wire-compatibility](#) 部分.

语义兼容

Hadoop 努力确保 API behavior 版本保持稳定，尽管改变可能导致 behavior 改变

测试和 Javadoc 指定 API 的 behavior。hadoop 社区指定 API 的过程更加严格，加强测试套件，以验证符合规范。有效地产生正式规范行为，可以容易地测试的子集。

Policy

API behavior 可以改变，以解决不正确的行为，比如伴随着升级已存在的 bug 修改等。

Wire 兼容性

Wire 兼容性相关数据通过 wire 传播在 hadoop 进程之间。hadoop 使用协议缓存区【大多数使用 RPC 通信】

保护兼容性需要禁止修改，如下描述。非 RPC 通信也应考虑。比如使用 HTTP 传输 HDFS 镜像作为快照或则传输 MapTask 输出。潜在的通信可以分类如下：

- Client-Server: hadoop 客户端和 server 通信（等 HDFS 客户端与 NameNode 协议，或则 YARN 客户端与 ResourceManager 协议）
- Client-Server (Admin):它是值得区分的一个子集的 Client-Server 协议仅由管理员命令使用(等 HAAdmin 协议)。由于这些协议只影响管理员（可以容忍的变化），最终用户（使用一般的 Client-Server 协议）不能。
- Server-Server:服务器之间的通信

Use Cases 【用例】

- Client-Server 兼容性需要允许用户继续使用旧版本客户端，甚至是升级后的集群到新的版本【反之亦然】。比如 hadoop2.1.0 客户端与 hadoop2.3.0 集群通信【这里只是整理自官网，升级时许谨慎】这里附上原文【Client-Server compatibility is required to allow users to continue using the old clients even after upgrading the server (cluster) to a later version (or vice versa). For example, a Hadoop 2.1.0 client talking to a Hadoop 2.3.0 cluster.】
- Client-Server 兼容性也需要允许用户在升级服务器（集群）之前升级客户端。例如，hadoop2.4.0 客户端与 hadoop2.3.0 集群【服务器】通信。这允许客户端在集群升级之前修复 bug。**请注意**，新的客户端的 API 或 shell 命令调用新的集群功能将无法使用。YARN 应用程序尝试使用新的 APIs(包括数据结构的新领域),APIs 尚未部署到集群可以预期 link exceptions.

- Client-Server 兼容性需要允许单独组件升级，其它组件不升级。例如 hdfs 从版本 2.1.0 到 2.2.0, MapReduce 不需要升级
- Server-Server 兼容性需要允许在一个集群中存在混合版本，因此集群需要在不停止的情况下滚动升级

Policy

- 在一个主要版本中保留了 Client-Server 和 Server-Server 的兼容性。（不同的类型不同的 Policy 尚待考虑）
- 在主要版本中可以打破兼容性，但是打破兼容性会有严重后果，因此需要在 hadoop 社区讨论
- Hadoop 的协议在 .proto (ProtocolBuffers) 文件中定义。Client-Server 和 Server-protocol .proto 文件被标记为稳定的。当一个 .proto 标记为稳定这意味着变化应兼容的方式如下所述：

下列变化是兼容的，并且在任何时间都可以被允许：

- 添加一个可选字段，期望代码处理由于与旧版本通信丢失的字段
- 添加一个新的 RPC/方法到服务
- 向消息添加新的可选请求
- 重命名字段
- 重命名 .proto 文件
- 改变 .proto 注释影响代码生成（如 Java 包名称）

下面改变是不兼容的，但仅在主版本中考虑

- 改变 rpc/method 名
- 改变 rpc/method 参数类型和返回类型
- 移除 rpc/method
- 改变 service 名称
- 改变消息名称
- 以不兼容的方式修改字段类型（递归定义）
- 改变一个可选的必须字段
- 改变或删除一个必须字段
- 删除一个可选的字段,只要可选字段允许删除合理的默认值

下面改变不兼容，因此不允许

- 改变一个字段的 id
- 重新使用以前删除的字段
- 字段是非常便宜的，改变和重用是不好的

Java 二进制兼容性最终用户应用程序即 Apache Hadoop 的 ABI

作为 hadoop 修订升级终端用户期望它们的程序不做任何修改继续工作。为满足这一期望，因此支持 API 兼容, Semantic 兼容 和 Wire 兼容.

尽管如此，Hadoop 是非常复杂的,分布式系统和服务各种类型的用户案例。特别是 map/reduce 是一个非常广泛的 API; 在某种意义上终端用户有非常大的空间，比如当运行 mapreduce 任务，本地磁盘的分配，任务的环境变量【配置】等。在这种情况下，它变得非常难以完全指定，支持，绝对兼容性。

用例

- 现有的 mapreduce 程序，包括现有 jars 最终用户应用程序和项目比如 Apache Pig, Apache Hive, Cascading 等当指向在一个主版本中升级的 Hadoop 集群，可以无修改的工作。
- 现有的 Yarn 程序，包括现有 jars 最终用户应用程序和项目比如 Apache Tez 等当指向在一个主版本中升级的 Hadoop 集群，可以无修改的工作。
- 现有的传输数据（输入/输出 hdfs 文件）应用程序，包括已存在的终端用户 jar 包应用程序比如 Apache Flume, 在主版本内当指向升级的 Apache Hadoop 集群应该无修改的工作。

Policy

- 现在有的 MapReduce, YARN & HDFS 应用程序和框架应该在一个主版本内，不做任何修改就可以工作。Apache Hadoop ABI 是支持的。

应用程序很小的一部分可能影响到磁盘的分布等。开发者社区将努力减少这些变化，不会使他们在一个小版本。更糟糕的情况，我们将考虑强还原这些重大更改，如果有必要将取消版本发布。

- 尤其是 MapReduce 应用，开发者社区将尽力支持提供二进制兼容性在各主要版本如应用 `org.apache.hadoop.mapred`。

APIs 支持兼容性 hadoop-1.x 和 hadoop-2.x. 查看在 hadoop-1.x 和 hadoop-2.x MapReduce 应用程序，更多细节点此

REST APIs

REST API 兼容性对应于两个请求（URL）的和响应于每个请求（内容，其可含有其它 URL）。

- [WebHDFS- Stable](#)
- [ResourceManager](#)
- [NodeManager](#)
- [MR Application Master](#)
- [History Server](#)
- [Timeline Server v1 REST API](#)

Policy

API 注释稳定，至少在一个主版本中保存兼容，可能在主版本的新版本的 REST API 被弃用

Metrics/JMX

尽管 Metrics API 由 Java API 兼容管理，实际 metrics 暴露给 hadoop 为用户实现自动化使用它们（脚本等）。添加额外的 metrics 是兼容的。修改（例如，改变单位或测量）或则移除已存在的 metrics 破坏兼容性，类似的改变 JMX MBean 对象名称也破坏兼容性。

Policy

在一个主版本内容 Metrics 应保持兼容性。

文件格式和元数据

用户和系统级数据（包括元数据）存储在不同格式的文件。改变元数据或则存储数据/元数据的文件格式将导致版本间不兼容。

用户级文件格式

改变终端用户存储数据的格式阻碍了后面版本的访问，因此非常重要保持文件格式的的兼容性。人们总是在现有的格式上添加一个“新”的格式改进。这些格式例子包括 har, war, SequenceFileFormat 等。

Policy

不向前兼容的用户文件格式改变是非常严格的在主版本。新发布版本期望读取已有格式，但写入数据的格式可能与先前版本不兼容。社区更倾向于创建一个新的格式，来代替对现有格式的不兼容的改变。

系统内部文件格式

hadoop 内部数据以文件的方式存储，改变这些格式会导致不兼容。这种改变不像用户级别文件具有毁灭性，兼容性可以被打破的 policy 是重要的。

MapReduce

MapReduce 使用格式如 I-File 存储 MapReduce 指定数据

Policy

MapReduce 内部是格式如 IFile 在一个版本内保持通用性。改变这些格式会引发运行中的 jobs 失败，因此我们应该确保新的客户端以兼容的方式能够取 shuffle 数据从旧的服务端

HDFS Metadata

HDFS 元数据（镜像和 edit logs）在一个指定的格式。无论是格式还是元数据改变会阻止后面版本读取旧版本元数据。这种不兼容的变化可能需要 hdfs 升级转换元数据使它访问。一些改变需要更多的升级。取决于不相容的变化的程度：

自动：image 自动升级，无需显示升级

直接：image 是可升级的，但可能需要一个显式发布“升级”。

间接：image 是可升级的，但是可能首先升级到中间的发布版

不可升级：image 不可升级的

Policy

一个发布版升级必须允许集群回滚到旧版本和他的旧的磁盘格式。回滚需要还原原始数据，但是不需要恢复更新的数据
HDFS 元数据改变必须升级通过任一升级方式--- automatic【自动】，direct【直接】或则 indirect【间接】。
更多的细节 policies 在各种升级方式中还有待考虑

命令行界面 (CLI)

Hadoop 的命令行程序，可以使用可直接通过系统 shell 或通过 shell 脚本。改变一个命令的路径，删除或重命名的命令行选项，参数的顺序，或则命令返回代码和打破输出兼容性，可能对用户造成不利影响。

Policy

在随后的主版本中将被移除或则不兼容的修改之前，在主要版本中，CLI 命令被弃用（当使用时会有警告）

Web UI

Web UI,web 页面的详细内容和页面布局, 改变潜在的接口可能会影响页面信息

Policy

Web pages 在任何时候允许不兼容性改变。用户期望使用 REST APIs 获取任何信息

Hadoop 配置文件

用户使用 (1) hadoop 定义属性配置和提供暗示,以 hadoop (2) 定制属性传递信息给 job.因此配置属性的兼容性是双重的:

- 修改 key-names, values 的单位, 和 hadoop 自定义属性的默认值
- 自定义配置属性 keys 不能与 hadoop 定义属性命名空间冲突。通常情况下, 用户应该避免使用 hadoop 前缀: hadoop, io, ipc, fs, net, file, ftp, s3, kfs, ha, file, dfs, mapred, mapreduce, yarn.

Policy

Hadoop 定义的属性是被弃用的至少一个主要发布之前被删除。修改单位现有属性是不允许的。

在主要版本或则小版本中, hadoop 自定义属性值可以被改变

当前没有明确的 policy 规则, 当新的前缀被添加或则删除。前缀应该避免自定义配置属性。尽管如此, 如上所述, 用户应避免使用使用 Hadoop 的前缀: hadoop, io, ipc, fs, net, file, ftp, s3, kfs, ha, file, dfs, mapred, mapreduce, yarn.

目录结构

源代码,工具(源和测试), 用户日志, 配置文件, output 和 job history, 存储在本地磁盘或则 hdfs.改变。改变用户访问文件的目录结构破坏兼容性, 即使源路径通过 symbolic links 保存。(如果, 例如 servlet 访问路径, 被配置为不遵循符号链接)

Policy

布局的源代码和生成的构件可以随时更改, 特别是跨主要版本。在主要版本中, 开发者尝试(没有保证)保存目录结构; 尽管如此, 独立的文件可以被添加/移除/删除。最好的方式是补丁与提交的 Apache 源码树的源码保持同步。

目录结构, 配置文件, 用户日志, job history 被保存在一个主版本内跨越小版本和发布点(point releases)

Java Classpath

用户应用程序构建可能添加所有的 hadoop jars(包括 hadoop 库依赖)到应用程序 classpath。添加新的依赖或则升级已存在的依赖可能影响这些应用程序的 classpaths

Policy

当前没有策略, hadoop 依赖可以改变。

环境变量

用户和相关项目经常使用导出的环境变量（如 HADOOP_CONF_DIR），因此，删除或重命名的环境变量是一个不相容的改变。

Policy

当环境变量改变，没有 Policy。开发者在版本中尝试限制改变。

构建构件

hadoop 使用 maven 管理项目，更改构件会影响现在的用户 workflow。

Policy

测试构件：测试 jars 产生内部使用非常严格的，不希望在 hadoop 外部使用，类似 APIs 注释 @Private, @Unstable.

构建构件：hadoop 客户端构件 (maven groupId:artifactId) 在主版本中保持兼容性，其它构件可以以不兼容的方式改变。

硬件/软件需求

跟上硬件的最新进展，操作系统, JVMs, 和其他软件, 新的 hadoop 发行版或则他们的一些新的功能需要同样的更高版本。对于特定的环境，升级 Hadoop 可能需要升级相关的软件组件。

Policies

硬件：

- 1.结构：社区有没有计划对 Hadoop 的具体结构，但可以有家族特定的优化。
- 2.最少资源：在 hadoop 守护进程所需的资源没有保证期间，社区尝试不增加需求在发布的小版本内。

操作系统：在小版本中，社区将试图保持相同的操作系统的要求（OS 内核版本）。当前 GNU/Linux 和 Microsoft Windows 是官方支持的，社区 hadoop 工作的相当不错也在其它操作系统比如 Apple MacOSX, Solaris 等。

JVM 不会改变跨点发布的同一个小版本内，除非如果 JVM 版本变的不支持。Minor/major 版本以后可能需要一些/所有的支持的操作系统的 JVM。

其它软件：社区试图保持由 Hadoop 的所需的额外软件的最低版本。例如，SSH, Kerberos 身份等。

参考：

下面是与主题相关的一些相关 JIRAs 和 pages:

Here are some relevant JIRAs and pages related to the topic:

- The evolution of this document - [HADOOP-9517](#)

- Binary compatibility for MapReduce end-user applications between hadoop-1.x and hadoop-2.x - MapReduce Compatibility between [hadoop-1.x and hadoop-2.x](#)
- Annotations for interfaces as per interface classification schedule - HADOOP-7391[Hadoop Interface Classification](#)
- Compatibility for Hadoop 1.x releases - [HADOOP-5071](#)
- The Hadoop [Roadmap page](#) that captures other release policies

版本: hadoop2.7.1

hadoop 入门-第一章 General: 第六节开发人员和用户接口指南: hadoop 接口分类

问题导读

1.hadoop 接口分为哪两类?

2.audience 分为几类，具体有哪几类？

3.Stability 有哪些分类？

4.对于 audience 和 Stability 你是如何理解的？



目的

这里提供的接口分类是为开发人员和用户提供接口的指南。这个分类指导开发者声明目标受众和用户接口及他的稳

定性。

用户接口的好处：知道什么接口可以用，什么接口不可以用及它们的稳定性

开发者好处：防止接口意外改变和因此，意外的影响用户或其他组件或系统。这对于一个大的系统是非常有用的，许多

开发者谁可能不都具有该项目的共享状态/历史记录。

接口分类

hadoop 采用以下接口分类，这个分类来自 OpenSolaris taxonomy，在某种程度上，来自雅虎内部使用的分类。接口有

两个主要属性：Audience 和 Stability

Audience

Audience 指出了接口潜在使用者。许多接口是 internal/private 的实现。其它是 public/external 接口，意味着更广泛的使用被应用程序或则客户端。

例如：in posix, lib 是外部的或则公共接口，大部分内核都是内部或私有接口。某些接口是针对其他特定子系统的。

确认接口的 Audience 可以帮助了解打破它带来的影响。例如，它可能是好的，打破一个接口的兼容性，其受众是一个小数目的特定子系统。另一方面，它可能是不好的，打破数以百万计的互联网用户依赖的协议接口。

hadoop 使用下面种类的 audience

Private:

内部接口项目内部使用（比如 hdfs, mapreduce），不能被应用程序和其它项目使用。它可以做任何的改变。大多数项目接口是私有的，也被称为项目专用。

Limited-Private:这样的接口由特定的项目或则接口使用（通常密切相关的项目）。其它项目或则系统不能使用。接口改变将通知指定项目。例如：`hadoop` 项目一些接口是 `LimitedPrivate{HDFS, MapReduce}`，它们是私有的 `HDFS` 和 `MapReduce` 项目。

Public

这些接口一般被应用程序使用

Stability

`Stability` 指出了什么样的接口是稳定的，当接口做不兼容改变时是允许的。`hadoop api` 有下列类别 `stability`

Stable

版本演进的同时保留与次要版本的相容；换句话说，不兼容的更改 `API` 的标记为稳定只允许在主要版本（即在 `m.0`）。

Evolving

`Evolving`，不兼容的改变在小版本里允许的（即 `m.x`）

Unstable

不通用的改变不稳定的 APIs 任何时候都是允许的。这通常只对私有接口

尽管如此一些人称其为 `public` 接口以强调，它不应该被用作接口，对于 `public` 接口，被标记为非接口比“Unstable”可能

更合适。例子：公共可视化接口是 `unstable`（即不是接口），`GUI`，`clis` 的输出格式将改变

Deprecated

在未来可能被删除的 APIs，不应该使用。

常见问题解答

java 作用域难道不充足吗？

Java 的范围不是很完整. 一个类经常被强制为 `public` 为内部其它组件使用。它没有 `friends` 或则像 C++

`sub-package-private`

我可以很容易地访问私有实现的接口，如果它是 Java 的 `public`。哪里体现保护和控制？

这样做的目的并不提供绝对的访问控制,其目的是为了传达给用户和开发。一个可以访问私有继承的函数在 `libc`.

hadoop 入门-第一章 General: 第七节 Hadoop 文件系统 API : 概述

这是 hadoop 文件系统 APIs 规范。这个文件系统的模型内容是一组路径，可以是路径，符号链接（Linux 中概念），或则文件

目前在这方面令人惊讶的一点现有技术。Unix 文件系统作为树形节点有多个规范。但是没有公共定义，“Unix 文件系统作为数据存储访问的概念模型”。

这个说明尝试这样做。为了定义 hadoop 文件系统模型和 APIs，一致多个文件系统可以继承 APIs，提供一致的数据模型的应用。它不试图规范文件系统的并发行为，除了期望 Hadoop 客户端展示 hdfs 通用行为。

1. [Introduction](#)
2. [Notation](#)
3. [Model](#)
4. [FileSystem class](#)
5. [FSDataInputStream class](#)
6. [Testing with the Filesystem specification](#)
7. [Extending the specification and its tests](#)

hadoop 入门-第二章 common: 第一节 hadoop 本地库 指南

问题导读

1.hadoop 本地库支持哪些平台?

2.本地库是否有 32, 64 之分?

3.hadoop 通过什么工具来检测本地库是否加载正确?

4.如何加载本地库? 包含哪些步骤?

5.本地库在什么情况下不需要使用 **DistibutedCache?**



概述

这个指南描述了 **hadoop** 本地库，包括关于共享本地库的小讨论。

注意：

取决于你的环境，这个词 “native libraries”涉及所有的*.so’，你需要编译；这个词 “native compression”涉及所有的*.so’ 的你需要编译指定与压缩相关的。当前，尽管如此，这个文档仅限于 **hadoop** 本地库 (libhadoop.so).文档为 libhdfs 库 (libhdfs.so) 点这 [here](#).

hadoop 本地库

处于性能考虑和非可用性 **Java** 实现，**Hadoop** 的具有某些部件的本地实现，这些组件单独是可用的。动态链接库调用本地原生 **Hadoop** 库。在 *nix (如：Linux, unix) 平台上本地库名字为 libhadoop.so.

用法

相当容易使用 **hadoop** 本地库：

- 1.查看组件
- 2.查看支持平台
- 3.下载一个构建好的 **hadoop** 发布版本或则自己构建本地库。无论是自己下载还是自己构建（编译），本地的名字是相同的： libhadoop.so
- 4.安装压缩编解码开发包 (> > **zlib-1.2**, **gzip-1.2**) :
如果你下载本地库，安装一个或则多个开发包，无论压缩编解码你想用于部署。
如果你想编译本地库，它是强制性安装了开发包。
- 5.检查运行时日志文件

组件

本地库包含各种组件：

- Compression Codecs (bzip2, lz4, snappy, zlib)
- Native IO utilities for HDFS Short-Circuit Local Reads and Centralized Cache Management in HDFS
- CRC32 校验实现

支持平台

hadoop 本地库仅支持在*nix 平台上，不支持 Cygwin 和 Mac OS X 平台。

hadoop 本地库主要用在 GNU/Linux 平台，已测试

- RHEL4/Fedora
- Ubuntu
- Gentoo

上面 hadoop 32/64 位本地库与相应的 32/64 位 jvm 工作

下载

hadoop 预构建 32-位 i386-Linux 本地库作为 hadoop 分布式一部分是有效的，位于 `e lib/native` 目录。你可以下载从 hadoop Common Releases.

一定要安装 `zlib` 和/或 `gzip` 开发包

构建

hadoop 本地库是用 ANSI C 编写，并使用 GNU 自动工具链 (`autoconf`, `autoheader`, `automake`, `autoscan`, `libtool`) 构建。这意味着他可以直接使用工具链 (`autoconf`, `autoheader`, `automake`, `autoscan`, `libtool`) 在任何平台上构建（查看支持平台）

在平台上包需要安装：

- C compiler (e.g. GNU C Compiler)
- GNU Autotools Chain: `autoconf`, `automake`, `libtool`
- `zlib-development` package (stable version $\geq 1.2.0$)
- `openssl-development` package(e.g. `libssl-dev`)

一旦你安装必备包使用标准的 Hadoop 的 pom.xml 文件，通过本地库标识构建 hadoop 本地库

[Bash shell] 纯文本查看 复制代码

?

```
1 $ mvn package -Pdist,native -DskipTests -Dtar
```

你可以查看新构建的库在

[Bash shell] 纯文本查看 复制代码

?

```
1 $ hadoop-dist/target/hadoop-2.7.1/lib/native
```

请注意以下几点：

- 1.它强制安装 `zlib` 和 `gzip` 开发包在目标平台，构建 hadoop 本地库。尽管如此，如果你想安装部署使用一个 `codec` 包，也是足够的。
- 2.为了构建和部署 hadoop 本地库，使用正确的 `zlib 32/64 位` 库是需要的，在目标平台上依赖 `32/64 位 jvm`。

运行时

`bin/hadoop` 脚本确保 hadoop 本地库通过系统属性 `-Djava.library.path=<path>` 在库路径。

在运行时，检查 hadoop MapReduce 任务日志文件

- 1.如果所有的事情准备好，然后调试 `util.NativeCodeLoader`，尝试加载自定义构建本地库。。。 `INFO util.NativeCodeLoader - Loaded the native-hadoop library`
- 2.如果产生错误，然后：`INFO util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable`

检查

`NativeLibraryChecker` 是一个工具用来检测本地库是否加载正确，你可以启动 `nativelibrarychecker` 如下：

[Bash shell] 纯文本查看 复制代码

?

```
1$ hadoop checknative -a
214/12/06 01:30:45 WARN bzip2.Bzip2Factory: Failed to load/initialize
3native-bzip2 library system-native, will use pure-Java version
414/12/06 01:30:45 INFO zlib.ZlibFactory: Successfully loaded & initialized
5native-zlib library
6Native library checking:
7hadoop: true /home/ozawa/hadoop/lib/native/libhadoop.so.1.0.0
8zlib:     true /lib/x86_64-linux-gnu/libz.so.1
```

```
9snappy: true /usr/lib/libsnappy.so.1
  lz4:      true revision:99
  bzip2:    false
```

本地共享库

你可以加载任何本地共享库使用 DistributedCache，分发和符号链接库文件

这个例子展示了如何分发共享本地库 mylib.so,和从 mapreduce 任务加载的。

1.首先复制库到 hdfs:bin/hadoop fs -copyFromLocal mylib.so.1 /libraries/mylib.so.1

2.job 启动程序应该包含下面:

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 DistributedCache.createSymlink(conf);
1 DistributedCache.addCacheFile("https://host:port/libraries/mylib.so.
2 1#mylib.so", conf);
```

3.MapReduce 任务应该包含:

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 System.loadLibrary("mylib.so");
```

注意:

如果你下载或则构建了 hadoop 本地库，不需要使用 DistibutedCache 使库提供给 mapreduce 任务版本: 2.7.1

hadoop 入门-第二章 common: 第二节 hadoop 代理用户 - 超级用户代理其它用户

问题导读

1.你认为什么情况下使用代理用户?

2.代理用户与普通用户的区别是什么?

3.如何配置超级用户代理其它用户?



说明

这个文档描述怎么使用超级用户提交 job 和代理其它用户访问 hdfs

用例

下一节中描述的代码示例适用于以下用例。

超级用户'super' 代理用户 joe 提交 job 和访问 hdfs,超级用户有 kerberos 认证, 用户 joe 没有。任务需要运行用户 joe 和任何文件访问 NameNode 需要用户 joe。它需要用户 joe 连接 namenode 或则 job tracker 使用 super 的 kerberos 凭证的连接认证, 换句话说, 超级用户模拟【冒充】用户 joe

代码示例

在这个例子中超级的凭证用于登录和创建一个代理用户 ugi 对象为 joe。这个操作实现了代理用户 ugi 对象的 doAs 方法

[Bash shell] 纯文本查看 复制代码

[?](#)

```
01...
02 //Create ugi for joe. The login user is 'super'.
03 UserGroupInformation ugi =
04     UserGroupInformation.createProxyUser("joe",
05UserGroupInformation.getLoginUser());
06 ugi.doAs(new PrivilegedExceptionAction<Void>() {
07     public Void run() throws Exception {
08         //Submit a job
09         JobClient jc = new JobClient(conf);
10         jc.submitJob(conf);
11         //OR access hdfs
12         FileSystem fs = FileSystem.get(conf);
13         fs.mkdir(someFilePath);
```

```
14      }  
      }
```

配置

你可以配置代理用户使用属性 `hadoop.proxyuser.$superuser.hosts` 或则使用 `hadoop.proxyuser.$superuser.groups` 和 `hadoop.proxyuser.$superuser.users`.通过指定如下 `core-site.xml`, 超级用户 `super` 冒充用户属于 `group1` 和 `group2`, 仅能连接 `host1` 和 `host2`

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 <property>  
2     <name>hadoop.proxyuser.super.hosts</name>  
3     <value>host1,host2</value>  
4 </property>  
5 <property>  
6     <name>hadoop.proxyuser.super.groups</name>  
7     <value>group1,group2</value>  
8 </property>
```

如果没有配置这些, 冒充不被允许, 连接失败。更宽松的不严格的安全性, 通配符的值*允许代理任何用户, 任何 `host`。比如通过以下 `core-site.xml`, 用户 `oozie` 访问任何 `host`, 可以代理任何用户所属的任何组

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 <property>  
2     <name>hadoop.proxyuser.oozie.hosts</name>  
3     <value>*</value>  
4 </property>  
5 <property>  
6     <name>hadoop.proxyuser.oozie.groups</name>  
7     <value>*</value>  
8 </property>
```

`hadoop.proxyuser.$superuser.hosts` 访问 `ip` 地址列表,CIDR 格式和/或主机名 IP 地址范围。比如以下, 用户 `super` 访问主机 `ip` 地址范围 `10.222.0.0-15` 到 `10.113.221.221`, 可以代理 `user1` 和 `user2`.

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 <property>  
2     <name>hadoop.proxyuser.super.hosts</name>
```

```
3     <value>10.222.0.0/16, 10.113.221.221</value>
4 </property>
5 <property>
6     <name>hadoop.proxyuser.super.users</name>
7     <value>user1, user2</value>
8 </property>
```

注意事项

如果集群运行在安全模式下，超级用户必须有 `kerberos` 凭据才能代理另一个用户。它不能使用委托 `tokens` 功能，如果

超级用户添加自己的委托 `token` 到代理用户 `ugi` 这是错误的，它允许代理用户连接到具有超级用户的特权服务。尽管如

此，如果超级用户确实想委托 `token` 给 `joe` 用户，它必须首先代理 `joe`，得到 `joe` 的委托 `token`，同样的方法如上面代码示

例，添加它到 `joe` 的 `ugi`，以这样的方式委托 `token` 属于 `joe`。

hadoop 入门-第二章 common: 第三节机架智能感知

问题导读

- 1.Hadoop 组件机架感知的作用是什么？
- 2.拓扑信息以什么格式保存，各自的含义是什么？
- 3.hadoop master 保存 slave 集群机架 id 由哪两种方式，该如何配置？
- 4.如何指定 Java 类实现拓扑映射？



Hadoop 组件机架感知。HDFS 放置为了容错，使用机架只能感知将副本放置不同机架。在网络故障或则集群内分区，可以提供有效的数据。hadoop master 保存 slave 集群机架 id，通过调用外部脚本或则 Java class 指定的配置文件。无论是使用 Java class，还是使用外部脚本，输出必须遵循 Java **org.apache.hadoop.net.DNSToSwitchMapping** 接口。这个接口期望一对一的保存，拓扑信息以 ``myrack/myhost`` 格式，``` 是拓扑分隔符，``myrack`` 是机架标识，``myhost`` 是客户端。每个机架假设单个 / 24 子网，一个可以使用的格式 / 192.168.100.0 / 192.168.100.5 作为一个的机架主机拓扑映射。

使用拓扑映射 Java 类，类的名字由配置文件参数 **topology.node.switch.mapping.impl** 指定。举例：`NetworkTopology.java`，包括 hadoop 分布式，可以通过管理员定制。使用 Java class 代替外部脚本有一个性能优势，当一个新的 slave 节点注册时，hadoop 不需 fork 外部进程

如果实现外部脚本，它由配置文件参数 **topology.script.file.name** 指定。不想 Java class，外部拓扑脚本不包括 hadoop 分布式，它是由管理员提供的。

当 forking 多个拓扑脚本，hadoop 将发送多个 ip 地址到 ARGV.IP 地址的数量发送到拓扑脚本由

net.topology.script.number.args 控制，默认是 100.如果

net.topology.script.number.args 改为 1，拓扑脚本会 forked 每个人 ip 提交给 DataNodes 和/或 NodeManagers.

如果 **topology.script.file.name** 或则 **topology.node.switch.mapping.impl** 没有设置，机架 id ``default-rack`` 返回任何通过的 ip 地址。尽管这种行为是可以的，它可能导致 HDFS block replication 问题，离架写一个备份作为默认行为。不能这样做，当仅有一个名为 ``default-rack`` 机架，

一个额外的配置，设置 **mapreduce.jobtracker.taskcache.levels**，它决定了 MapReduce 将使用的 caches 的级别（网络拓扑）。举例，如果默认值为 2，将初始两个级别缓存。一个为 hosts（`host -> task mapping`）和另外一个为 racks（`rack -> task mapping`）。给我们一对一映射 ``myrack/myhost``。

python 例子

[Bash shell] 纯文本查看 复制代码

?

```

#!/usr/bin/python
# this script makes assumptions about the physical environment.
# 1) each rack is its own layer 3 network with a /24 subnet, which
01 # could be typical where each rack has its own
02 # switch with uplinks to a central core router.
03 #
04 #
05 #           +-----+
06 #           |core router|
07 #           +-----+
08 #           /               \
09 #   +-----+           +-----+
10 #   |rack switch|       |rack switch|
11 #   +-----+           +-----+
12 #   | data node |       | data node |
13 #   +-----+           +-----+
14 #   | data node |       | data node |
15 #   +-----+           +-----+
16 #
17 # 2) topology script gets list of IP's as input, calculates network address, and p
18
19 import netaddr
20 import sys
21 sys.argv.pop(0)
22 discard name of topology script from argv list as we just want IP addresses
23
24 netmask = '255.255.255.0'
25 to what's being used in your environment. The example uses a /24
26
27 for ip in sys.argv:
28 over list of datanode IP's
29 address = '{0}/{1}'.format(ip, netmask)
30 like 'ip/netmask' to make netaddr work
31 try:
32     network_address = netaddr.IPNetwork(address).network # calculate and
33     print "{0}/".format(network_address)
34 except:
35     print "/rack-unknown"
36 value if unable to calculate network address

```

bash 例子

[Bash shell] 纯文本查看 复制代码

?

```
#!/bin/bash
# Here's a bash example to show just how simple these scripts can be
# Assuming we have flat network with everything on a single switch, we
# can fake a rack topology.
01# This could occur in a lab environment where we have limited nodes, like
02# 2-8 physical machines on a unmanaged switch.
03# This may also apply to multiple virtual machines running on the same
04# physical hardware.
05# The number of machines isn't important, but that we are trying to fake
06# a network topology when there isn't one.
07#
08#          +-----+          +-----+
09#          |jobtracker|          |datanode|
10#          +-----+          +-----+
11#
12#          \                      /
13# +-----+  +-----+  +-----+
14# |datanode|--| switch |--|datanode|
15# +-----+  +-----+  +-----+
16#          /                      \
17#          +-----+          +-----+
18#          |datanode|          |namenode|
19#          +-----+          +-----+
20# With this network topology, we are treating each host as a rack. This
21# is being done by taking the last octet
22# in the datanode's IP and prepending it with the word '/rack-'. The
23# advantage for doing this is so HDFS
24# can create its 'off-rack' block copy.
25# 1) 'echo $@' will echo all ARGV values to xargs.
26# 2) 'xargs' will enforce that we print a single argv value per line
27# 3) 'awk' will split fields on dots and append the last field to the
28# string '/rack-'. If awk
# fails to split on four dots, it will still print '/rack-' last
# field value

echo $@ | xargs -n 1 | awk -F '.' '{print "/rack-"$NF}'
```

注释:

fork 叉子\分岔\岔口\复刻，西方人吃饭用的东西，经常用作刀和叉。

计算机程序设计中的分叉函数。返回值：若成功调用一次则返回两个值，子进程返回 0，父进程返回子进程标记；否则，

出错返回-1。

fork 函数将运行着的程序分成 2 个（几乎）完全一样的进程，每个进程都启动一个从代码的同一位置开始执行的线程。

这两个进程中的线程继续执行，就像是两个用户同时启动了该应用程序的两个副本。

argv

argv 接收从命令行传来的参数,在程序里可以通过 **argv** 来使用。

ARGC 和 **ARGV** 中的 **ARG** 指的是"参数"（外语：**ARG**uments, **arg**ument counter 和 **arg**ument vector ） [1]

至少有两个参数至主函数：**ARGC** 和 **ARGV**;

首先是一个计算提供的参数到程序，

第二个是对字符串数组的指针。

hadoop 入门-第二章 common: 第四节安全模式说明

问题导读

1.什么是 Kerberos Principal?

2.Hadoop 守护进程是否使用不同的用户?

3.各个守护进程 keytab 文件各有什么特点? 区别在什么地方?



说明

本文档介绍了如何配置验证 Hadoop 的安全模式。默认 hadoop 运行在非安全模式下，没有实际的身份验证要求。使用 hadoop 服务，每一个用户和服务都需要 Kerberos 身份验证。Hadoop 的安全功能，包括身份验证，服务级授权，Web 控制台认证和数据保密。

身份验证

(<http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/SecureMode.html#Authentication>)

服务级授权 (<http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/ServiceLevelAuth.html>)

Web 控制台认证

(<http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/HttpAuthentication.html>)

数据保密

(http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/SecureMode.html#Data_confidentiality)

身份认证

终端用户帐户

当打开服务级别授权，终端用户使用 `hadoop` 安全模式需要 Kerberos 身份验证。最简单的方式认证是使用 `kinit` Kerberos 命令

【注释：`kinit` 命令：

用途

获得或更新 Kerberos 票据授权票据 (ticket-granting ticket)。

描述

`kinit` 命令获得或更新 Kerberos 票据授权票据。如果不在命令行上指定一个票据标志，则使用由在 Kerberos 配置文件 (`kdc.conf`) 中的 `[kdcdefault]` 和 `[realms]` 指定的密钥分发中心 (KDC) 选项。

】

Hadoop 守护进程用户帐户

确保 HDFS 和 YARN 守护进程运行不同的 Unix 用户，例如 `hdfs` 和 `yarn`。同样确保 MapReduce JobHistory server 运行不同的用户比如 `mapred`。

建议让它们共享 Unix 组，比如 `hadoop`。详细查看 “Mapping from user to group”

User:Group	Daemons
<code>hdfs:hadoop</code>	NameNode, Secondary NameNode, JournalNode, DataNode
<code>yarn:hadoop</code>	ResourceManager, NodeManager
<code>mapred:hadoop</code>	MapReduce JobHistory Server

Kerberos Principal hadoop 守护进程和用户

运行 `hadoop` 守护进程服务在安全模式下，Kerberos principals 是需要的。每个服务读取身份认证信息保存在 `keytab` 文件【有相应的权限】HTTP web-consoles 服务于来自不同的 RPC 的 `sprincipal`【原文：HTTP web-consoles should be served by principal different from RPC's one.】

下面部分给出了 Hadoop 服务凭证的例子。

注释：

【Kerberos 协议术语解释【此部分是为帮助理解额外附加内容】

Principal: 在 Kerberos 中，Principal 是参加认证的基本实体。一般来说有两种，一种用来表示 Kerberos 数据库中的用户，另一种用来代表某一特定主机，也就是说 Principal 是用来表示客户端和服务端身份的实体，Principal 的格式采用 ASN.1 标准，即 Abstract Syntax Notation One，来准确定义)，Principal 是由三个部分组成：名字(name)，实例(instance)，REALM (域)。比如一个标准的 Kerberos 的用户是：name/instance@REALM。

Name: 第一部分。在代表客户方的情况，它是一个用户名；在代表主机的情况，它是写成 host。

Instance: 第二部分。对 name 的进一步描述，例如 name 所在的主机名或 name 的类型等，可省略。它与第一部分之间用 '/' 分隔，但是作为主机的描述时写成 host/Instance。

Realm: 第三部分。是 Kerberos 在管理上的划分，在 KDC 中所负责的一个域数据库称作为 Realm。这个数据库中存放

有该网络范围内的所有 **Principal** 和它们的密钥，数据库的内容被 **Kerberos** 的认证服务器 **AS** 和票据授权服务器 **TGS** 所使用。**Realm** 通常是永远是大写的字符,并且在大多数 **Kerberos** 系统的配置中，一般 **Realm** 和该网络环境的 **DNS** 域是一致的。与第二部分之间用 '@' 分隔，缺省为本地的 **Realm**。】

HDFS

NameNode keytab 文件，在 NameNode 主机，应该是下面样子

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
$ klist -e -k -t /etc/security/keytab/nn.service.keytab
Keytab name: FILE:/etc/security/keytab/nn.service.keytab
KVNO Timestamp Principal
14 07/18/11 21:08:09 nn/full.qualified.domain.name@REALM.TLD (AES-256 CTS
2mode with 96-bit SHA-1 HMAC)
34 07/18/11 21:08:09 nn/full.qualified.domain.name@REALM.TLD (AES-128 CTS
4mode with 96-bit SHA-1 HMAC)
54 07/18/11 21:08:09 nn/full.qualified.domain.name@REALM.TLD (ArcFour with
6HMAC/md5)
74 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-256
8CTS mode with 96-bit SHA-1 HMAC)
94 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-128
CTS mode with 96-bit SHA-1 HMAC)
4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (ArcFour
with HMAC/md5)
```

Secondary NameNode keytab 文件，应该是下面样子

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
$ klist -e -k -t /etc/security/keytab/sn.service.keytab
Keytab name: FILE:/etc/security/keytab/sn.service.keytab
1KVNO Timestamp Principal
24 07/18/11 21:08:09 sn/full.qualified.domain.name@REALM.TLD (AES-256 CTS
3mode with 96-bit SHA-1 HMAC)
44 07/18/11 21:08:09 sn/full.qualified.domain.name@REALM.TLD (AES-128 CTS
5mode with 96-bit SHA-1 HMAC)
64 07/18/11 21:08:09 sn/full.qualified.domain.name@REALM.TLD (ArcFour with
7HMAC/md5)
84 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-256
9CTS mode with 96-bit SHA-1 HMAC)
4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-128
CTS mode with 96-bit SHA-1 HMAC)
```

```
4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (ArcFour
with HMAC/md5)
```

DataNode keytab 文件，应该是下面样子

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
$ klist -e -k -t /etc/security/keytab/dn.service.keytab
Keytab name: FILE:/etc/security/keytab/dn.service.keytab
KVNO Timestamp Principal
14 07/18/11 21:08:09 dn/full.qualified.domain.name@REALM.TLD (AES-256 CTS
2mode with 96-bit SHA-1 HMAC)
34 07/18/11 21:08:09 dn/full.qualified.domain.name@REALM.TLD (AES-128 CTS
4mode with 96-bit SHA-1 HMAC)
54 07/18/11 21:08:09 dn/full.qualified.domain.name@REALM.TLD (ArcFour with
6HMAC/md5)
74 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-256
8CTS mode with 96-bit SHA-1 HMAC)
94 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-128
CTS mode with 96-bit SHA-1 HMAC)
4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (ArcFour
with HMAC/md5)
```

YARN

ResourceManager keytab 文件，在 ResourceManager 客户端，应该是下面样子

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
$ klist -e -k -t /etc/security/keytab/rm.service.keytab
Keytab name: FILE:/etc/security/keytab/rm.service.keytab
1 KVNO Timestamp Principal
2 4 07/18/11 21:08:09 rm/full.qualified.domain.name@REALM.TLD (AES-256 CTS
3 mode with 96-bit SHA-1 HMAC)
4 4 07/18/11 21:08:09 rm/full.qualified.domain.name@REALM.TLD (AES-128 CTS
5 mode with 96-bit SHA-1 HMAC)
6 4 07/18/11 21:08:09 rm/full.qualified.domain.name@REALM.TLD (ArcFour with
7 HMAC/md5)
8 4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-256
9 CTS mode with 96-bit SHA-1 HMAC)
4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-128
```

CTS mode with 96-bit SHA-1 HMAC)

4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (ArcFour with HMAC/md5)

NodeManager keytab 文件, 在每个客户端, 应该是下面样子

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
$ klist -e -k -t /etc/security/keytab/nm.service.keytab
```

```
Keytab name: FILE:/etc/security/keytab/nm.service.keytab
```

```
KVNO Timestamp Principal
```

```
14 07/18/11 21:08:09 nm/full.qualified.domain.name@REALM.TLD (AES-256 CTS 2mode with 96-bit SHA-1 HMAC)
```

```
34 07/18/11 21:08:09 nm/full.qualified.domain.name@REALM.TLD (AES-128 CTS 4mode with 96-bit SHA-1 HMAC)
```

```
54 07/18/11 21:08:09 nm/full.qualified.domain.name@REALM.TLD (ArcFour with 6HMAC/md5)
```

```
74 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-256 8CTS mode with 96-bit SHA-1 HMAC)
```

```
94 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-128 CTS mode with 96-bit SHA-1 HMAC)
```

```
4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (ArcFour with HMAC/md5)
```

MapReduce JobHistory Server

MapReduce JobHistory Server keytab 文件, 应该是下面样子

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
$ klist -e -k -t /etc/security/keytab/jhs.service.keytab
```

```
1 Keytab name: FILE:/etc/security/keytab/jhs.service.keytab
```

```
2 KVNO Timestamp Principal
```

```
3 4 07/18/11 21:08:09 jhs/full.qualified.domain.name@REALM.TLD (AES-256 CTS 3 mode with 96-bit SHA-1 HMAC)
```

```
4 4 07/18/11 21:08:09 jhs/full.qualified.domain.name@REALM.TLD (AES-128 CTS 5 mode with 96-bit SHA-1 HMAC)
```

```
6 4 07/18/11 21:08:09 jhs/full.qualified.domain.name@REALM.TLD (ArcFour 7 with HMAC/md5)
```

```
8 4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-256 9 CTS mode with 96-bit SHA-1 HMAC)
```

```
4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (AES-128
```

CTS mode with 96-bit SHA-1 HMAC)

4 07/18/11 21:08:09 host/full.qualified.domain.name@REALM.TLD (ArcFour with HMAC/md5)

从 Kerberos principal 映射用户帐户

hadoop 映射 Kerberos principal 到操作系统用户帐户，使用由 `hadoop.security.auth_to_local` 指定的规则。它是同样的工作方式在 Kerberos 配置文件

(`krb5.conf` http://web.mit.edu/Kerberos/krb5-latest/doc/admin/conf_files/krb5_conf.html) 作为 `auth_to_local`。额外，`hadoop auth_to_local` 映射支持 `/L` 标记,小写字母返回名称。

默认,如果领域匹配到 `default_realm`，它选择 `principal` 名称第一个组件（通常定义在`/etc/krb5.conf`）。例如 `host/full.qualified.domain.name@REALM.TLD` 映射 `host` 为默认规则。

定制规则使用 `hadoop kerbname` 命令来测试。这个命令允许指定一个 `principal` 和使用 Hadoop 的当前 `auth_to_local` 规则集。输出什么身份验证，`hadoop` 将使用它的用法。

从用户到组的映射

尽管 `hdfs` 文件是有用户和组的，但是 `hadoop` 自己没有定义组，用户到组的映射是由操作系统或 LDAP 来做。改变映射的方式由 `hadoop.security.group.mapping` 的值提供映射名称。查看 [Permissions Guide for details](#)。实际上你需要管理 SSO 环境使用 Kerberos 和 LDAP 的 Hadoop 的安全模式。

代理用户

一些产品比如 Apache Oozie，它代表终端用户访问 `hadoop` 访问需要能够代理终端用户。详细查看 [HDFS Permissions Guide](#) (<http://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-hdfs/HdfsPermissionsGuide.html>)

datanode 安全

因为 `datanode` 传输数据协议不需要使用 `hadoop RPC` 框架，`DataNode` 必须使用特权端口验证自己的身份，端口由 `dfs.datanode.address` 和 `dfs.datanode.http.address` 指定。验证基于这样的假设，攻击者将无法获得根权限。当你使用 `root` 用户执行 `hdfs datanode` 命令，服务器进程首先绑定特权端口，然后放弃特权，由 `HADOOP_SECURE_DN_USER` 指定的用户帐户来运行。启动进程使用 `jsvc` 安装到 `JSVC_HOME`。你必须指定 `HADOOP_SECURE_DN_USER` 和 `JSVC_HOME` 作为环境变量（在 `hadoop-env.sh` 文件中）。

2.6.0 版，SASL 可以用来验证数据传输协议。这样的配置，不在需要安全集群以 `root` 用户使用 `jsvc` 和绑定特权端口启动 `datanode`。为了启用数据传输协议 SASL，需要设置 `hdfs-site.xml` 文件中的 `dfs.data.transfer.protection`，对于

dfs.datanode.address 设置一个非特权端口，为 **HTTPS_ONLY** 设置 **dfs.http.policy** 和确定 **HADOOP_SECURE_DN_USER** 环境变量没有定义。需要注意的是，如果 **dfs.datanode.address** 设置特权端口，使用 SASL 数据传输协议是不可能的。这是必需的向后兼容的原因。

为了迁移集群，使用 **root** 身份认证开始使用 SASL 来代替。首先确保是版本 2.6.0 或则后面版本，部署集群节点跟外部应用程序一样需要连接集群。仅 2.6.0 版本或则 2.6.0 以后版本 HDFS 客户端可以连接 **DataNode** 使用数据传输协议 SASL，因此至关重要是在迁移前有一个正确的版本。

2.6.0 版本或则以后版本已经部署，更新任何外部应用程序启用 SASL 认证配置。如果一个 HDFS 客户端启用了 SASL，然后它即可以使用 **root** 身份认证，也可以使用 SASL 身份认证连接 **DataNode**。所有客户端的配置改变保障随后的 **DataNode** 配置改变不影响应用程序。最后每一个独立的 **DataNode** 可以被迁移，修改它的配置和重启。可以有一个混合的，有的 **DataNodes** 运行使用 **root** 身份验证，有的是用 SASL 身份验证，在迁移期间暂时这是被允许的。因为 HDFS 客户端启用 SASL 可以连接二者。

数据保密性

RPC 数据加密

hadoop 的服务和客户端之间传送的数据。设置 **hadoop.rpc.protection** 为"privacy"在 **core-site.xml** 文件中激活数据加密

Block 数据传输的数据加密

DataNode 数据传输协议加密激活，需要设置 **dfs.encrypt.data.transfer** 为 true.可选，你需要设置 **dfs.encrypt.data.transfer.algorithm** 为 "3des" 或则 "rc4"，选择指定的算法。如果没有指定，系统默认使用配置 JCE，通常是 3DES。

设置 **dfs.encrypt.data.transfer.cipher.suites** 为 AES/CTR/NoPadding 激活 AES encryption. 默认，如果没有指定，不使用 AES。什么时候使用 AES，由 **dfs.encrypt.data.transfer.algorithm** 指定，它仍然在初始密钥交换时使用。AES 密钥位长度可以被配置，设置 **dfs.encrypt.data.transfer.cipher.key.bitlength** 为 128, 192 或则 256，默认 128.AES 提供了最大的加密强度和最佳的性能。于此同时，3DES 和 RC4 被经常用于 hadoop 集群。

HTTP 数据加密

用于 hadoop 集群。

在 Web-console 和 clients 之间传输数据由 SSL 保护。

配置

HDFS 和本地文件系统二者的权限

下面表格列出了 HDFS 不同的路径和本地文件系统（所有节点）和推荐权限

Filesystem	Path	User:Group	Permissions
local	dfs.namenode.name.dir	hdfs:hadoop	drwx-----
local	dfs.datanode.data.dir	hdfs:hadoop	drwx-----
local	\$HADOOP_LOG_DIR	hdfs:hadoop	drwxrwxr-x
local	\$YARN_LOG_DIR	yarn:hadoop	drwxrwxr-x
local	yarn.nodemanager.local-dirs	yarn:hadoop	drwxr-xr-x
local	yarn.nodemanager.log-dirs	yarn:hadoop	drwxr-xr-x
local	container-executor	root:hadoop	--Sr-s--*
local	conf/container-executor.cfg	root:hadoop	r-----*
hdfs	/	hdfs:hadoop	drwxr-xr-x
hdfs	/tmp	hdfs:hadoop	drwxrwxrwt
hdfs	/user	hdfs:hadoop	drwxr-xr-x
hdfs	yarn.nodemanager.remote-app-log-dir	yarn:hadoop	drwxrwxrwt
hdfs	mapreduce.jobhistory.intermediate-done-dir	mapred:hadoop	drwxrwxrwt
hdfs	mapreduce.jobhistory.done-dir	mapred:hadoop	drwxr-x---

通用配置

为了开启 hadoop rpc 身份验证，设置 **hadoop.security.authentication** 属性值为 "kerberos"。设置下面列出的安全相关设置。下面属性在集群所有节点 core-site.xml

参数	值	说明
hadoop.security.authentication	<i>kerberos</i>	简单：没有身份验证。（默认）：启用 Kerberos 认证协议。
hadoop.security.authorization	<i>true</i>	启用 RPC service-level

		authorization. (http://hadoop.apache.org/docs/r2.7.1/hadoop-pro)
hadoop.rpc.protection	authentication	authentication : 仅身份验证 (默认) integrity : 完整性检查除了身份验证 privacy : 数据加密除了完整性
hadoop.security.auth_to_local	RULE:exp/RULE:exp2 ..DEFAULT	这个字符串值包含新的一行字符 查看 Kerberos documentation(http://web.mit.edu/Kerberos/krb5-1)
hadoop.proxyuser.superuser.hosts		逗号分隔 host, host 超级用户允许代理(这里理解感觉不是太准确, 贴 superuser access are allowd to impersonation) *表示通配符
hadoop.proxyuser.superuser.groups		逗号分隔组, 组的用户由超级用户代理 (这里理解感觉不是太准确, 贴 users impersonated by superuser belongs.) *表示通配符

NameNode

参数	值	说明
dfs.block.access.token.enable	true	为了安全操作, 启用 HDFS block 访问令牌
dfs.https.enable	true	这个值被弃用. 使用 dfs.http.policy
dfs.http.policy	HTTP_ONLY or HTTPS_ONLY or HTTP_AND_HTTPS	HTTPS_ONLY 关闭 http access. 这个选项优先于弃用配置 dfs.https.enable 和 hadoop.ssl.enabled。如果使用数 据传输协议 SASL, 代替 DataNode root 和使用特权端口, 这个属性必 须设置 HTTPS_ONLY 保障 HTTP servers 身份认证。(查看 dfs.data.transfer.protection.)
dfs.namenode.https-address	nn_host_fqdn:50470	
dfs.https.port	50470	
dfs.namenode.keytab.file	/etc/security/keytab/ nn.service.keytab	NameNode Kerberos keytab 文件 .
dfs.namenode.kerberos.principal	nn/_HOST@REALM.TLD	NameNode Kerberos principal 名字.
dfs.namenode.kerberos.internal.spnego.principal	HTTP/_HOST@REALM.TLD	NameNode HTTP Kerberos principal 名字 .

Secondary NameNode

参数	值	说明
dfs.namenode.secondary.http-address	<i>c_nn_host_fqdn:50090</i>	
dfs.namenode.secondary.https-port	<i>50470</i>	
dfs.secondary.namenode.keytab.file	<i>/etc/security/keytab/ sn.service.keytab</i>	Secondary NameNode Kerberos keytab 文件 .
dfs.secondary.namenode.kerberos.principal	<i>sn/_HOST@REALM.TLD</i>	Secondary NameNode Kerberos principal 名字.
dfs.secondary.namenode.kerberos.internal.spnego.principal	<i>HTTP/_HOST@REALM.TLD</i>	Secondary NameNode HTTP Kerberos principal 名字 .

DataNode

参数	值	说明
dfs.datanode.data.dir.perm	<i>700</i>	
dfs.datanode.address	<i>0.0.0.0:1004</i>	DataNode 安全必须使用特权端口, 为了确保 server 安全启动。这意味着 server 必须通过 jsvc 启动。另外, 如果使用数据传输协议 SASL 身份验证, 必须设置一个非特权端口 (查看 <i>dfs.data.transfer.protection</i> .)
dfs.datanode.http.address	<i>0.0.0.0:1006</i>	DataNode 安全必须使用特权端口, 确保 server 安全启动。意味着 server 必须通过 jsvc 启动
dfs.datanode.https.address	<i>0.0.0.0:50470</i>	
dfs.datanode.keytab.file	<i>/etc/security/keytab /dn.service.keytab</i>	DataNode Kerberos keytab 文件
dfs.datanode.kerberos.principal	<i>dn/_HOST@REALM.TLD</i>	DataNode Kerberos principal 名称
dfs.encrypt.data.transfer	<i>false</i>	当使用数据加密时, 设置为 true
dfs.encrypt.data.transfer.algorithm		当使用数据加密来控制加密算法时, 可选设置为 3des 或则 rc4
dfs.encrypt.data.transfer.cipher.suites		当使用数据加密, 可选设置 AES/CTR/NoPadding, 激活 AES 数据加密
dfs.encrypt.data.transfer.cipher.key.bitlength		当使用 AES 数据加密时, 设置 128, 192 或则 256 to 控制 key 位长度
dfs.data.transfer.protection		<i>authentication</i> : 仅仅身份验证 <i>integrity</i> : 除了身份验证, 还有完整性验证

		<p><i>privacy</i> : 除了完整性验证, 还有数据加密</p> <p>此属性未指定默认值。设置这个属性启用 SASL 数据传输协议身份验证。如果这个启用了, 然后 <code>dfs.datanode.address</code> 必须使用非特权端口, <code>dfs.http.policy</code> 必须设置为 <code>HTTPS_ONLY</code> 和 <code>HADOOP_SECURE_DN_USER</code> 环境变量必须未指定, 当启动 <code>DataNode</code> 进程</p>
--	--	--

WebHDFS

参数	值	说明
<code>dfs.web.authentication.kerberos.principal</code>	<code>http/_HOST@REALM.TLD</code>	WebHDFS Kerberos principal 名称.
<code>dfs.web.authentication.kerberos.keytab</code>	<code>/etc/security/keytab/http.service.keytab</code>	WebHDFS Kerberos keytab 文件 .

ResourceManager

参数	值	说明
<code>yarn.resourcemanager.keytab</code>	<code>/etc/security/keytab/rm.service.keytab</code>	ResourceManager Kerberos keytab 文件.
<code>yarn.resourcemanager.principal</code>	<code>rm/_HOST@REALM.TLD</code>	ResourceManager Kerberos principal 名称

NodeManager

参数	值	说明
<code>yarn.nodemanager.keytab</code>	<code>/etc/security/keytab/nm.service.keytab</code>	NodeManager Kerberos keytab 文件
<code>yarn.nodemanager.principal</code>	<code>nm/_HOST@REALM.TLD</code>	NodeManager Kerberos principal 名称
<code>yarn.nodemanager.container-executor.class</code>	<code>org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor</code>	使用 <code>LinuxContainerExecutor</code> .
<code>yarn.nodemanager.linux-container-executor.group</code>	<code>hadoop</code>	NodeManager 的 Unix 组
<code>yarn.nodemanager.linux-container-executor.path</code>	<code>/path/to/bin/container-executor</code>	容器执行器可执行程序的路径。

配置 WebAppProxy

WebAppProxy 提供了一个由应用程序和最终用户输出代理端口, 在 web 应用程序之间。如果安全启用, 他将警告用户, 在访问可能非安全的 web 应用程序。身份验证和授权使用代理处理就像任何其他特权的 web 应用程序。

参数	值	说明
yarn.web-proxy.address	WebAppProxy host:port 代理	AM web apps 代理 说明 <i>host:port</i> 如果跟 <code>yarn.resourcemanager.webapp.address</code> 一样或则它没有定义 <code>ResourceManager</code> 将运行代理,否则一个独立的代理服务器将需要启动。
yarn.web-proxy.keytab	<code>/etc/security/keytab/web-app.service.keytab</code>	WebAppProxy Kerberos keytab 文件.
yarn.web-proxy.principal	wap/ <u>_HOST@REALM.TLD</u>	WebAppProxy Kerberos principal 名称 .

LinuxContainerExecutor

Yarn 框架使用 ContainerExecutor。Yarn 框架定义了容器的启动和控制，一下是 hadoop yarn 可用的：

ContainerExecutor	描述
DefaultContainerExecutor	默认执行器，Yarn 用来管理容器的执行。容器的进程 NodeManager 与 Linux 用户相同
LinuxContainerExecutor	仅支持 GNU/Linux，这个 executor 运行容器即可以作为 YARN 用户提交应用程序（当完整安全性启用）也可以作为一个专用用户（默认 nobody），当完整的安全性未启用。当完整安全性启用，这个 executor 需要创建所有用户，在集群节点容器启动时。它使用一个 setuid 执行文件，包括在 Hadoop 分布中。NodeManager 使用该可执行程序来启动和杀死容器。setuid 执行切换到用户提交和启动应用程序或杀死容器。为了最大限度的安全,这个 executor 设置严格的权限,和本地文件和目录的用户/组所有者,比如共享的 objects, jars, 中间文件, 日志文件等. 需要需要说明的是, 因为这, 除了应用程序所有者和 NodeManager, 没有其它用户可以访问任何的本地文件/目录包括这些本地化的分布式缓存的部分。

要构建 LinuxContainerExecutor 可执行文件运行：

[Bash shell] 纯文本查看 复制代码

?

```
1 $ mvn package -Dcontainer-executor.conf.dir=/etc/hadoop/
```

路径 `-Dcontainer-executor.conf.dir` 应该是集群节点上的路径，它的 `setuid` 可执行文件的配置文件被定位，可执行文件应安装在 `$HADOOP_YARN_HOME/bin`。

可执行文件必须有特定的权限：`6050` 或则 `--Sr-s---` 允许用户拥有 `root` 权限(超级用户)和组拥有专用组（比如 `hadoop`），`NodeManager` Unix 用户是它的组成员和不是普通的应用程序用户。

`LinuxTaskController` 需要包括路径和由 `yarn.nodemanager.local-dirs` 和 `yarn.nodemanager.log-dirs` 指定的目录，如上表所述对目录设置的权限，设置为 `755`。

- `conf/container-executor.cfg`

可执行文件需要配置文件称之为 `container-executor.cfg`，现在在配置文件目录传递给上述提到的 `mvn target`。
 配置文件必须由拥有的用户运行 `NodeManager`（上述例子的 `yarn` 用户），任何人拥有的组，和应该有权限 `0400` 或则 `r-----`。

可执行文件需要下面配置项，现在在 `conf/container-executor.cfg` 文件中。该配置项被提及为简单的键=值对，每行一个：

参数	值	说明
<code>yarn.nodemanager.linux-container-executor.group</code>	<code>hadoop</code>	NodeManager Unix 组. <code>container-executor</code> 的二进制所属组应该属于这个组，应该是跟 <code>NodeManager</code> 配置的值相同. 这个配置需要验证访问 <code>container-executor</code> 二进制的安全.
<code>banned.users</code>	<code>hdfs, yarn, mapred, bin</code>	禁止用户
<code>allowed.system.users</code>	<code>foo, bar</code>	允许系统用户
<code>min.user.id</code>	<code>1000</code>	防止其他超级用户。

下面是本地系统文件权限，需要相关的 `LinuxContainerExecutor` 不同的路径

文件系统	路径	用户:组	权限
local	<code>container-executor</code>	<code>root:hadoop</code>	<code>--Sr-s--*</code>
local	<code>conf/container-executor.cfg</code>	<code>root:hadoop</code>	<code>r-----*</code>
local	<code>yarn.nodemanager.local-dirs</code>	<code>yarn:hadoop</code>	<code>drwxr-xr-x</code>
local	<code>yarn.nodemanager.log-dirs</code>	<code>yarn:hadoop</code>	<code>drwxr-xr-x</code>

MapReduce JobHistory Server

参数	值	说明
<code>mapreduce.jobhistory.address</code>	<code>MapReduce JobHistory Server</code> <i>host:port</i>	默认端口是 10020.
<code>mapreduce.jobhistory.keytab</code>	<code>/etc/security/keytab/jhs.service.keytab</code>	MapReduce JobHistory Server Kerberos keytab 文件
<code>mapreduce.jobhistory.principal</code>	<code>jhs/_HOST@REALM.TLD</code>	MapReduce JobHistory Server. Kerberos principal 名称

hadoop 入门-第二章 common: 第五节服务级别授权指南

问题导读

- 1.hadoop 默认是否启用服务级别授权?
- 2.访问控制列表该如何配置?
- 3.访问控制列表的格式是什么?
- 4.如何刷新服务级别授权配置? 是否需要重启?



目的

这个文档描述了怎么配置和管理 hadoop 服务级别授权

准备

确保 hadoop 安装, 配置和设置正确。更多信息查看:

- 新手首次安装

hadoop2.7【单节点】单机、伪分布、分布式安装指导

<http://www.aboutyun.com/thread-12798-1-1.html>

- 大的集群安装

hadoop (2.x) 以 hadoop2.2 为例完全分布式最新高可靠安装文档

<http://www.aboutyun.com/thread-7684-1-1.html>

概述

服务级授权是初始化授权机制, 确保客户端连接指定的 hadoop 服务, 预配置, 权限和授权访问给定的服务。例子, 一个 mapreduce 集群可以使用这种机制, 允许配置用户/组列表提交 jobs。这个\$HADOOP_CONF_DIR/hadoop-policy.xml

配置文件用于定义访问控制 **hadoop** 服务变量列表。

服务级别授权比以前执行许多，其它访问控制检查，比如文件权限检查，**job** 队列访问控制等。

配置

这一部分描述了如何通过配置文件 `$HADOOP_CONF_DIR/hadoop-policy.xml` 配置服务级别授权

启用服务级别授权

默认，服务级别授权 **hadoop** 是禁用的.在`$HADOOP_CONF_DIR/core-site.xml` 中设置配置属性

`hadoop.security.authorization` 为 `true`,启用服务级别授权

hadoop 服务和配置属性

这部分列出了不同的 **hadoop** 服务和它们的 **knobs** 配置

属性	服务
<code>security.client.protocol.acl</code>	客户端协议 ACL 用于用户代码通过 <code>DistributedFileSystem</code> .
<code>security.client.datanode.protocol.acl</code>	<code>ClientDatanodeProtocol</code> ACL ， <code>datanode</code> 客户端协议用于 <code>block</code> 恢复.
<code>security.datanode.protocol.acl</code>	<code>DatanodeProtocol</code> ACL, 用于 <code>datanodes</code> 与 <code>namenode</code> 通信.
<code>security.inter.datanode.protocol.acl</code>	<code>InterDatanodeProtocol</code> ACL, <code>datanode</code> 之间协议, 用于更新时间戳
<code>security.namenode.protocol.acl</code>	<code>NamenodeProtocol</code> ACL ， 这个协议用于 <code>secondary namenode</code> 与 <code>namenode</code> 通信
<code>security.job.client.protocol.acl</code>	<code>JobSubmissionProtocol</code> ACL, 用于作业提交, <code>job</code> 客户端与 <code>resourcemanager</code> 通信 , 查询 <code>job</code> 状态等.
<code>security.job.task.protocol.acl</code>	<code>TaskUmbilicalProtocol</code> ACL ， 用于 <code>map</code> 任何与 <code>reduce</code> 任务与父 <code>nodemanager</code> 通信
<code>security.refresh.policy.protocol.acl</code>	<code>RefreshAuthorizationPolicyProtocol</code> ACL, 用于 <code>dfsadmin</code> 和 <code>rmadmin</code> 命令 刷新安全策略
<code>security.ha.service.protocol.acl</code>	<code>HAService</code> ACL 用于 <code>HAAdmin</code> 管理 <code>namenode</code> 的 <code>active</code> 和 <code>stand-by</code> 状态

访问控制列表

`$HADOOP_CONF_DIR/hadoop-policy.xml` 定义了每个 hadoop 服务访问控制列表。每个访问控制列表有一个简单的格式：用户和组的列表是逗号分隔的名称列表。这两者是由一个空格隔开的。

例子：

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 user1,user2 group1, group2.
```

在该行的开始处添加一个空白，如果只提供组列表，等效于一个逗号分隔用户列表后跟一个空格或则什么也没有暗示仅仅一组给定的用户。

给定值*暗示所有用户允许访问服务。如果访问控制列表没有定义，那么使用 **security.service.authorization.default.acl** 的值。

如果 **security.service.authorization.default.acl** 没有定义，使用*。

Blocked 访问控制列表在某些情况下，它需要指定一个服务的 blocked 访问控制列表。这个指定的用户和组的列表，没有授权访问服务。blocked 访问控制列表格式与上面访问控制列表格式相同。blocked 访问控制列表由

`$HADOOP_CONF_DIR/hadoop-policy.xml` 指定。这个属性名字起源于后缀 `".blocked"`。

例子：blocked 的属性名访问控制列表 `security.client.protocol.acl>>` 将会是 `<<<security.client.protocol.acl.blocked` 对于一个服务，可以指定访问控制列表和 blocked 的控制列表。如果一个用户在访问控制列表，不在 blocked 访问控制列表，此用户被授权，可以访问服务。

如果 blocked 访问控制列表没有定义一个服务，使用 `security.service.authorization.default.acl.blocked` 的值，如果 `security.service.authorization.default.acl.blocked` 没有定义，使用空 blocked 访问控制列表

刷新服务级别授权配置

NameNode 和 ResourceManager 服务级别授权配置可以在没有重启 hadoop master 守护进程的情况下改变。集群管理员可以改变 master 节点上的 `$HADOOP_CONF_DIR/hadoop-policy.xml` 文件和使 NameNode 和 ResourceManager 重新加载各自的配置通过 `-refreshServiceAcl` 切换到各自 `dfsadmin` 和 `rmadmin` 命令。刷新 NameNode 服务级别授权配置

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 $ bin/hdfs dfsadmin -refreshServiceAcl
```

刷新 ResourceManager 服务级别授权配置

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 $ bin/yarn rmadmin -refreshServiceAcl
```

当然，可以使用 `$HADOOP_CONF_DIR/hadoop-policy.xml` 中 **security.refresh.policy.protocol.acl** 属性限制有能力刷新服务级别授权配置的某些用户/组。

使用 ip 地址列表, host 名称,和 ip 地址范围访问控制服务, 访问服务控制基于客户端 ip 地址访问服务。可以限制对服务的访问从一组机器通过指定一个 ip 地址列表, host 名称 和 ip 范围. 每一个服务的属性名称是来自相应的 ACL 属性名称。如果 acl 属性名是 **security.client.protocol.acl**, host 列表属性名将是 **security.client.protocol.hosts** .如果 hosts 列表没有定义一个服务, 则使用 **security.service.authorization.default.hosts** 的值。如果 **security.service.authorization.default.hosts** 的值没有定义, 使用*。

它可以指定一个 hosts 的 blocked 列表。仅这些机器在 hosts 列表, 但不在 blocked hosts 列表将会授权访问服务。这个属性名的后缀源于 “.blocked”。

例子: 这个属性名 blocked hosts 列表 `security.client.protocol.hosts>>` 将会是 `<<security.client.protocol.hosts.blocked`。如果 blocked hosts 列表没有定义一个服务, 则使用 `security.service.authorization.default.hosts.blocked` 的值, 如果 `security.service.authorization.default.hosts.blocked` 没有定义, 则应用空 blocked hosts 列表。

例子

只允许用户 `alice, bob` 和在 `mapreduce` 组的用户提交 `jobs` 到 `mapreduce` 集群:

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 <property>
2     <name>security.job.client.protocol.acl</name>
3     <value>alice,bob mapreduce</value>
4 </property>
```

只允许 `DataNodes` 运行作为属于 `datanodes` 组与 `NameNode` 通信

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 <property>
2     <name>security.datanode.protocol.acl</name>
3     <value>datanodes</value>
4 </property>
```

允许任何用户作为 `DFSClient` 与 `HDFS` 集群通信

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 <property>
2     <name>security.client.protocol.acl</name>
3     <value>*</value>
4 </property>
```

hadoop 入门-第二章 common: 第六节 Hadoop HTTP web-consoles 认证机制

问题导读

1.如何配置 **Hadoop HTTP web-consoles** 所需要的用户身份验证?

2.哪个配置文件可以配置 **Hadoop HTTP** 认证?

3.**hadoop.http.authentication.type** 的默认值是什么?

4.你认为 **Hadoop HTTP web** 认证的价值是什么?



注释:

Hadoop HTTP web-consoles 翻译为 hadoop HTTP 控制台, 这里不太好理解, 这里个人理解为我们的普通 web 网页。为

什么要对它认证那，应该是不随便让其它人访问，以免遭到攻击，破坏等原因。

说明

这个文档描述了如何配置 Hadoop HTTP web-consoles 所需要的用户身份验证

默认 Hadoop HTTP web-consoles (JobTracker, NameNode, TaskTrackers 和 DataNodes)允许访问没有任何的验证。

类似的 Hadoop RPC,Hadoop HTTP web-consoles 可以被配置需要 Kerberos 身份认证使用 HTTP SPNEGO 协议（支持浏览器比如 Firefox 和【IE】 Internet Explorer）

额外的，Hadoop HTTP web-consoles 支持等效的 hadoop 的伪/简单 身份认证.如果选项被禁用，用户必须指定它们的

用户名在浏览器中交互使用 user.name 查询字符串参数。举例：<http://localhost:50030/jobtracker.jsp?user.name=babu>。

如果 HTTP web-consoles 定制身份验证是必需的，实现一个插件来支持所述备用认证机制是有可能的。（参考 Hadoop

的 Hadoop 的身份验证的详细信息，编写验证处理程序）

下面描述了如何配置 hadoop HTTP web-consoles 需要的用户认证

配置

下面属性应该在集群所有节点 `core-site.xml` 文件中。**hadoop.http.filter.initializers**: 添加到这个属性的

`org.apache.hadoop.security.AuthenticationFilterInitializer` 初始化类

hadoop.http.authentication.type: 定义用户 HTTP web-consoles. 的认证。支持值是: `simple | kerberos |`

`#AUTHENTICATION_HANDLER_CLASSNAME#`. 默认值是 `simple`

hadoop.http.authentication.token.validity: 指示认证 token 多长时间是有效的, 在更新之前。默认值是 `36000`

hadoop.http.authentication.signature.secret.file: 签名密钥文件用于签名密钥 tokens.

同样密钥应该用于集群所有节点, `JobTracker`, `NameNode`, `DataNode` and `TastTracker`. 默认值是 `s`

`$user.home/hadoop-http-auth-signature-secret`. 重要: 文件是只读的对于运行守护进程的 Linux 用户。

hadoop.http.authentication.cookie.domain: domain 使用 HTTP cookie 存储认证 token。为了认证通过集群所有节

点正确的工作, domain 必须正确设置。没有默认值, http 没有 domain 仅与 hostname 发布的 http cookie 工作。

重要:

当使用 ip 地址, 浏览器忽略 cookie 域名设置。这样的设置集群所有节点正确工作必须配置产生带有 `hostname.domain`

的 urls 在它上面

hadoop.http.authentication.simple.anonymous.allowed: 意思是如果使用 'simple' 身份验证时, 允许匿名请求.

默认值是 *true*

hadoop.http.authentication.kerberos.principal:意思是当使用 'kerberos'认证时, Kerberos principal 用于 HTTP

endpoint。

The principal short 名称必须是 *HTTP* 对 Kerberos HTTP SPNEGO 规范。默认值是

HTTP/_HOST@\$LOCALHOST,_HOST 如果当前是-, 替换为绑定 HTTP server 的地址

hadoop.http.authentication.kerberos.keytab:带有 Kerberos principal 认证的本地 keytab 文件用于 HTTP

endpoint。默认值是 ***\$user.home/hadoop.keytab.i***

版本【2.7.1】

hadoop 入门-第二章 common: 第七节 Hadoop Key 管理服务器(KMS) - 文档集

问题导读

1.Hadoop KMS 是什么?

2.如何启动/停止 KMS?

3.KMS 的安全如何配置?



Hadoop KMS 是一个基于 Hadoop 的 KeyProvider API 的用密码写的 key 管理 server。Client 是一个 KeyProvider 的实现，使用 KMS HTTP REST API 与 KMS 交互。

KMS 和它的客户端内置安全和它们支持 HTTP SPNEGO Kerberos 身份验证和 HTTPS 安全转换。

KMS 是一个 Java Web 应用程序，运行在与 Hadoop 发行版绑定在一起的预先配置好的 Tomcat 服务器上。

KMS 客户端配置

KMS 客户端 *KeyProvider* 使用 kms scheme,嵌入的 URL 必须是 KMS 的网址 (URL)。举例：`http://localhost:16000/kms` 运行 KMS，KeyProvider URI 是 `kms://http@localhost:16000/kms`。并且 `https://localhost:16000/kms` 运行 KMS，KeyProvider URI 是 `kms://https@localhost:16000/kms`。

KMS

KMS 配置

配置 KMS 支持 KeyProvider 属性在 `etc/hadoop/kms-site.xml` 配置文件

[Bash shell] 纯文本查看 复制代码

?

```
1<property>
```

```
2     <name>hadoop.kms.key.provider.uri</name>
```

```
3     <value>jceks://file@/${user.home}/kms.keystore</value>
```

```
4</property>
5
6<property>
7   <name>hadoop.security.keystore.java-keystore-provider.password-file</name>
8   <value>kms.keystore.password</value>
9</property>
```

这个密码文件通过 `classpath` 查找 `hadoop` 的配置文件

注意：需要重启 KMS 生效配置文件。

KMS 缓存

KMS caches keys 是为了避免在短时间内过多的点击

Cache 默认是启用的。（禁用设置 `hadoop.kms.cache.enable boolean` 属性为 `false`）cache 只用下面三个方法：

`getCurrentKey()` 和 `getKeyVersion()` 和 `getMetadata()`。

`getCurrentKey()` 方法,cached 保留最多 30000 毫秒不管 key 正被访问的数量(避免过期 key 被认为是当前的)

`getKeyVersion()` 方法,cached 默认闲置时 600000 毫秒（10 分钟）超时。超时配置通过下面配置文件

`etc/hadoop/kms-site.xml` 的属性配置

[Bash shell] 纯文本查看 复制代码

2

```
01<property>
02   <name>hadoop.kms.cache.enable</name>
03   <value>>true</value>
04 </property>
05
06 <property>
07   <name>hadoop.kms.cache.timeout.ms</name>
08   <value>600000</value>
09 </property>
10
11 <property>
12   <name>hadoop.kms.current.key.cache.timeout.ms</name>
13   <value>30000</value>
14 </property>
```

KMS 聚集 Audit 日志

Audit 日志汇总 为 API 访问 `GET_KEY_VERSION`, `GET_CURRENT_KEY`, `DECRYPT_EEK`, `GENERATE_EEK` 的操作

Entries 由 combined key (用户,key,操作)进行了分组， combined key 为一个可配置的聚合间隔，访问数次之后由 user

指定 end-point, 用户给予的 key flushed 到 audit 日志.

聚合间隔由下面属性配置

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 <property>
2     <name>hadoop.kms.aggregation.delay.ms</name>
3     <value>10000</value>
4 </property>
```

启动/停止 KMS

启动/停止 KMS 使用 KMS 的 bin/kms.sh 脚本.举例:

[Bash shell] [纯文本查看](#) [复制代码](#)

?

```
1 hadoop-2.7.1 $ sbin/kms.sh start
```

注意:不需任何参数的情况下, 调用这个脚本就可以列出所有可能的参数(开始、停止、运行等)。kms.sh 脚本是一个包装的 Tomcat's catalina.sh 脚本, 设置环境变量和 Java 系统属性运行 KMS

嵌入 Tomcat 配置

配置嵌入 Tomcat ,去 share/hadoop/kms/tomcat/conf.KMS 预配置 HTTP 和 Admin 端口在 Tomcat 的 server.xml 16000 和 16001.

Tomcat 日志也是预先配置到 hadoop 的 logs/目录

下面环境变量(设置在 KMS 的 etc/hadoop/kms-env.sh 脚本)可以用来改变这些值

- KMS_HTTP_PORT
- KMS_ADMIN_PORT
- KMS_MAX_THREADS
- KMS_LOGNOTE: 你需要重启 KMS, 生效配置改变.

加载本地库

下面环境变量(可以设置在 KMS 的 etc/hadoop/kms-env.sh 脚本)用于指定任何所需的本地库的位置。因为 Tomact 本地 Apache Portable Runtime (APR) 库:

[Bash shell] 纯文本查看 复制代码

[?](#)

```
1 JAVA_LIBRARY_PATH
```

KMS 的安全配置

启用 Kerberos HTTP SPNEGO 认证

配置 Kerberos `etc/krb5.conf` 你的 KDC server 信息的文件为 KMS 创建 `principal` 和 `keytab` ,它必须是一个 HTTP 服务 `principal`.

配置 KMS `etc/hadoop/kms-site.xml` 用正确的安全值, 举例

[Bash shell] 纯文本查看 复制代码

[?](#)

```
01 <property>
02     <name>hadoop.kms.authentication.type</name>
03     <value>kerberos</value>
04 </property>
05
06 <property>
07     <name>hadoop.kms.authentication.kerberos.keytab</name>
08     <value>${user.home}/kms.keytab</value>
09 </property>
10
11 <property>
12     <name>hadoop.kms.authentication.kerberos.principal</name>
13     <value>HTTP/localhost</value>
14 </property>
15
16 <property>
17     <name>hadoop.kms.authentication.kerberos.name.rules</name>
18     <value>DEFAULT</value>
19 </property>
```

注意: 你需要重启 KMS 生效配置

KMS 代理用户配置

每一个代理用户必须配置在 `etc/hadoop/kms-site.xml` 使用下面属性

[Bash shell] 纯文本查看 复制代码

2

```
01 <property>
02     <name>hadoop.kms.proxyuser.#USER#.users</name>
03     <value>*</value>
04 </property>
05
06 <property>
07     <name>hadoop.kms.proxyuser.#USER#.groups</name>
08     <value>*</value>
09 </property>
10
11 <property>
12     <name>hadoop.kms.proxyuser.#USER#.hosts</name>
13     <value>*</value>
14 </property>
```

*#USER#*是代理用户配置的用户名 *users* 属性的意思是 *users* 是模拟的。

groups 属性的意思是用户代理必须属于这个组

hosts 属性的意思是从 *host* 的代理用户可以做模拟请求

如果 *users*, *groups* 或则 *hosts* 有一个 ***,它的意思是代理用户对于 *users*, *groups* or *hosts* 没有限制

KMS 通过 HTTPS (SSL)

配置 KMS 工作通过 HTTPS, 下面两个属性必须设置在 `etc/hadoop/kms_env.sh` 脚本 (展示默认值)

- `KMS_SSL_KEYSTORE_FILE=$HOME/.keystore`
- `KMS_SSL_KEYSTORE_PASS=password`

在 KMS 目录, 与所提供的 `ssl-server.xml` 文件替换 `server.xml` 文件。

你需要创建 SSL 认证为 KMS, 作为 `kms` Unix 用户, 使用 `Javakeytool` 命令创建 SSL 证书

[Bash shell] 纯文本查看 复制代码

2

```
1 $ keytool -genkey -alias tomcat -keyalg RSA
```

在交互的提示中, 你将会被问一系列问题。它将会创建 `keystore` 文件, 被命名为 `.keystore` 和位于 `kms` 用户的 `home` 目录。

你输入的密码-"`keystore password`", 必须匹配在配置目录的 `kms-env.sh` 脚本的 `KMS_SSL_KEYSTORE_PASS` 环境变量的值

"What is your first and last name?" (即"CN") 的答案必须是运行 KMS 的机器的 `hostname`。

注意:

你必须重启 KMS 生效改变配置

KMS 访问控制

KMS ACL 配置定义在 KMS *etc/hadoop/kms-acls.xml* 配置文件。当它改变时，这个文件是 hot-reloaded。KMS 支持细粒度访问以及 KMS 操作黑名单通过 ACL 配置属性设置。

用户访问 KMS 首先检查列入访问控制列表的请求的操作，然后检查授权之前是否排除在黑名单。

[XML] 纯文本查看 复制代码

2

```
001<configuration>
002   <property>
003       <name>hadoop.kms.acl.CREATE</name>
004       <value>*</value>
005       <description>
006           ACL for create-key operations.
007           If the user is not in the GET ACL, the key material
008is not returned
009           as part of the response.
010       </description>
011   </property>
012
013   <property>
014       <name>hadoop.kms.blacklist.CREATE</name>
015       <value>hdfs,foo</value>
016       <description>
017           Blacklist for create-key operations.
018           If the user is in the Blacklist, the key material
019is not returned
020           as part of the response.
021       </description>
022   </property>
023
024   <property>
025       <name>hadoop.kms.acl.DELETE</name>
026       <value>*</value>
027       <description>
028           ACL for delete-key operations.
029       </description>
030   </property>
031
032   <property>
033       <name>hadoop.kms.blacklist.DELETE</name>
```

```
034     <value>hdfs, foo</value>
035     <description>
036         Blacklist for delete-key operations.
037     </description>
038 </property>
039
040 <property>
041     <name>hadoop.kms.acl.ROLLOVER</name>
042     <value>*</value>
043     <description>
044         ACL for rollover-key operations.
045         If the user is not in the GET ACL, the key material
046is not returned
047         as part of the response.
048     </description>
049 </property>
050
051 <property>
052     <name>hadoop.kms.blacklist.ROLLOVER</name>
053     <value>hdfs, foo</value>
054     <description>
055         Blacklist for rollover-key operations.
056     </description>
057 </property>
058
059 <property>
060     <name>hadoop.kms.acl.GET</name>
061     <value>*</value>
062     <description>
063         ACL for get-key-version and get-current-key
064operations.
065     </description>
066 </property>
067
068 <property>
069     <name>hadoop.kms.blacklist.GET</name>
070     <value>hdfs, foo</value>
071     <description>
072         ACL for get-key-version and get-current-key
073operations.
074     </description>
075 </property>
```

```
076
077     <property>
078         <name>hadoop.kms.acl.GET_KEYS</name>
079         <value>*</value>
080         <description>
081             ACL for get-keys operation.
082         </description>
083     </property>
084
085     <property>
086         <name>hadoop.kms.blacklist.GET_KEYS</name>
087         <value>hdfs,foo</value>
088         <description>
089             Blacklist for get-keys operation.
090         </description>
091     </property>
092
093     <property>
094         <name>hadoop.kms.acl.GET_METADATA</name>
095         <value>*</value>
096         <description>
097             ACL for get-key-metadata and get-keys-metadata
098operations.
099         </description>
100     </property>
101
102     <property>
103         <name>hadoop.kms.blacklist.GET_METADATA</name>
104         <value>hdfs,foo</value>
105         <description>
106             Blacklist for get-key-metadata and
107get-keys-metadata operations.
108         </description>
109     </property>
110
111     <property>
112         <name>hadoop.kms.acl.SET_KEY_MATERIAL</name>
113         <value>*</value>
114         <description>
115             Complimentary ACL for CREATE and ROLLOVER
116operation to allow the client
117             to provide the key material when creating or
```

```
118rolling a key.
119     </description>
120 </property>
121
122 <property>
123     <name>hadoop.kms.blacklist.SET_KEY_MATERIAL</name>
124     <value>hdfs,foo</value>
125     <description>
126         Complimentary Blacklist for CREATE and
127ROLLOVER operation to allow the client
128         to provide the key material when creating or
129rolling a key.
130     </description>
131 </property>
132
133 <property>
134     <name>hadoop.kms.acl.GENERATE_EEK</name>
135     <value>*</value>
136     <description>
137         ACL for generateEncryptedKey
138         CryptoExtension operations
139     </description>
140 </property>
141
142 <property>
143     <name>hadoop.kms.blacklist.GENERATE_EEK</name>
144     <value>hdfs,foo</value>
145     <description>
146         Blacklist for generateEncryptedKey
147         CryptoExtension operations
148     </description>
149 </property>
150
151 <property>
152     <name>hadoop.kms.acl.DECRYPT_EEK</name>
153     <value>*</value>
154     <description>
155         ACL for decrypt EncryptedKey
156         CryptoExtension operations
157     </description>
</property>
```

```
<property>
  <name>hadoop.kms.blacklist.DECRYPT_EEK</name>
  <value>hdfs,foo</value>
  <description>
    Blacklist for decrypt EncryptedKey
    CryptoExtension operations
  </description>
</property>
</configuration>
```

Key 访问控制

KMS 支持访问控制为非可读操作在 key 级别。所有的 key 访问操作被分类为：

- MANAGEMENT - createKey, deleteKey, rolloverNewVersion (创建 key, 删除 key, 翻新版本)
- GENERATE_EEK - generateEncryptedKey, warmUpEncryptedKeys (生成 key, 加密 key)
- DECRYPT_EEK - decryptEncryptedKey(解密加密 key)
- READ - getKeyVersion, getKeyVersions, getMetadata, getKeysMetadata, getCurrentKey(获取 key 版本, 获取 key 版本【多个】, 得到元数据, 到元数据【复数】, 得到当前 key)
- ALL - 上面所有

这些可以定义在 KMS etc/hadoop/kms-acls.xml 如下

所有的 keys, key 访问没有明确的配置, 它可能配置默认 key 访问控制为操作类型的一个子集。

他也可能配置一个“白名单”key ACL 为操作类型的一个子集。白名单 key ACL 是一个白名单额外的明确的或则默认每个 key ACL。如果没有 per-key ACL 明确设置, 一个用户将会授权访问, 如果他们出现在 per-key ACL 或则白名单 key ACL。

如果没有 ACL 配置为一个指定 key 和没有默认 ACL 配置和没有 root key ACL 为请求操作, 然后访问将会被拒绝。

注意: 默认和白名单 key ACL 不支持所有的操作限定符(The default and whitelist key ACL does not support ALL operation qualifier.)

[XML] 纯文本查看 复制代码

2

```
01 <property>
02   <name>key.acl.testKey1.MANAGEMENT</name>
03   <value>*</value>
04   <description>
05     ACL for create-key, deleteKey and rolloverNewVersion
06operations.
07   </description>
08 </property>
09
10 <property>
11   <name>key.acl.testKey2.GENERATE_EEK</name>
```

```
12     <value>*</value>
13     <description>
14         ACL for generateEncryptedKey operations.
15     </description>
16 </property>
17
18 <property>
19     <name>key.acl.testKey3.DECRYPT_EEK</name>
20     <value>admink3</value>
21     <description>
22         ACL for decryptEncryptedKey operations.
23     </description>
24 </property>
25
26 <property>
27     <name>key.acl.testKey4.READ</name>
28     <value>*</value>
29     <description>
30         ACL for getKeyVersion, getKeyVersions, getMetadata,
31getKeysMetadata,
32         getCurrentKey operations
33     </description>
34 </property>
35
36 <property>
37     <name>key.acl.testKey5.ALL</name>
38     <value>*</value>
39     <description>
40         ACL for ALL operations.
41     </description>
42 </property>
43
44 <property>
45     <name>whitelist.key.acl.MANAGEMENT</name>
46     <value>admin1</value>
47     <description>
48         whitelist ACL for MANAGEMENT operations for all keys.
49     </description>
50 </property>
51
52 <!--
53 'testKey3' key ACL is defined. Since a 'whitelist'
```

```
54 key is also defined for DECRYPT_EEK, in addition to
55 admink3, adminl can also perform DECRYPT_EEK operations
56 on 'testKey3'
57-->
58 <property>
59     <name>whitelist.key.acl.DECRYPT_EEK</name>
60     <value>adminl</value>
61     <description>
62         whitelist ACL for DECRYPT_EEK operations for all keys.
63     </description>
64 </property>
65
66 <property>
67     <name>default.key.acl.MANAGEMENT</name>
68     <value>user1,user2</value>
69     <description>
70         default ACL for MANAGEMENT operations for all keys that are
71not
72         explicitly defined.
73     </description>
74 </property>
75
76 <property>
77     <name>default.key.acl.GENERATE_EEK</name>
78     <value>user1,user2</value>
79     <description>
80         default ACL for GENERATE_EEK operations for all keys that
81are not
82         explicitly defined.
83     </description>
84 </property>
85
86 <property>
87     <name>default.key.acl.DECRYPT_EEK</name>
88     <value>user1,user2</value>
89     <description>
90         default ACL for DECRYPT_EEK operations for all keys that are
91not
92         explicitly defined.
93     </description>
94 </property>
95
```

```
96 <property>
97   <name>default.key.acl.READ</name>
98   <value>user1,user2</value>
    <description>
      default ACL for READ operations for all keys that are not
      explicitly defined.
    </description>
</property>
```

KMS Delegation Token 配置

KMS delegation token 密钥管理器可以用下面属性配置:

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
01<property>
02   <name>hadoop.kms.authentication.delegation-token.update-interval.sec</name>
03   <value>86400</value>
04   <description>
05     How often the master key is rotated, in seconds. Default value 1 day.
06   </description>
07 </property>
08
09 <property>
10   <name>hadoop.kms.authentication.delegation-token.max-lifetime.sec</name>
11   <value>604800</value>
12   <description>
13     Maximum lifetime of a delagation token, in seconds. Default value 7 days.
14   </description>
15 </property>
16
17 <property>
18   <name>hadoop.kms.authentication.delegation-token.renew-interval.sec</name>
19   <value>86400</value>
20   <description>
21     Renewal interval of a delagation token, in seconds. Default value 1 day.
22   </description>
23 </property>
24
```

```
25 <property>
26     <name>hadoop.kms.authentication.delegation-token.removal-scan-interval.sec</name>
27     <value>3600</value>
28     <description>
29         Scan interval to remove expired delegation tokens.
30     </description>
31 </property>
```

使用多个 KMS 实例背后的负载均衡器或则 VIP

使用多个 KMS 实例背后的负载均衡器或则 VIP 为了可扩展性和 HA 目的

当使用使用多个 KMS 实例背后的负载均衡器或则 VIP，一个用户的多个请求被不同的 KMS 实例处理。KMS 实例背后的负载均衡器或则 VIP 必须指定配置工作属性作为单个逻辑服务。

HTTP Kerberos Principals 配置

当 KMS 实例背后的负载均衡器或则 VIP，客户端将会使用 VIP 的 hostname 。为了 Kerberos SPNEGO 身份认证,URL 的 hostname 用于构建 Kerberos 服务的服务器名称， HTTP/#HOSTNAME#。这意味着所有的 KMS 实例必须有一个 Kerberos 服务名称与负载均衡器或则 VIP hostname

为了能否访问指定的 KMS 实例，KMS 实例必须有 Keberos 服务名与它自己的 hostname。这是所需的监控和管理的目的。

Kerberos 服务 principal 身份认证（对于负载均衡器/VIP hostname 和对于实际的 KMS 实例 hostname）必须在配置身份验证 keytab 文件。principal 名字在配置中指定必须是 '*'。举例：

[Bash shell] 纯文本查看 复制代码

?

```
1<property>
2    <name>hadoop.kms.authentication.kerberos.principal</name>
3    <value>*</value>
4</property>
```

注意：

如果使用 HTTPS, SSL 证书用于 KMS 实例必须配置支持多个 hostnames（查看 Java 7 以上 SAN 扩展支持具体如何做）

HTTP 认证签名

KMS 使用 Hadoop 认证 HTTP 认证。一旦客户端认证成功，Hadoop 认证发布一个签名 HTTP Cookie。 HTTP Cookie 已过期，过期之后，它将触发一个新的认证序列。这样做是为了避免在客户机的每个 HTTP 请求触发的认证。

一个 KMS 实例必须验证 HTTP Cookie 签名由其它 KMS 实例签署的。为了这个所有的 KMS 实例必须共享签署的 secret。

这个 secret 共享可以被用于 zookeeper 服务，zookeeper 服务这些配置在 KMS 以下的在 kms-site.xml 的属性

[XML] 纯文本查看 复制代码

2

```
01<property>
02    <name>hadoop.kms.authentication.signer.secret.provider</name>
03    <value>zookeeper</value>
04    <description>
05        Indicates how the secret to sign the authentication cookies will be
06        stored. Options are 'random' (default), 'string' and 'zookeeper'.
07        If using a setup with multiple KMS instances, 'zookeeper' should be us
08    </description>
09 </property>
10 <property>
11    <name>hadoop.kms.authentication.signer.secret.provider.zookeeper.path</name>
12    <value>/hadoop-kms/hadoop-auth-signature-secret</value>
13    <description>
14        The Zookeeper ZNode path where the KMS instances will store and retrie
15        the secret from.
16    </description>
17 </property>
18 <property>
19    <name>hadoop.kms.authentication.signer.secret.provider.zookeeper.connection
20    <value>#HOSTNAME#:#PORT#,...</value>
21    <description>
22        The Zookeeper connection string, a list of hostnames and port comma
23        separated.
24    </description>
25 </property>
26 <property>
27    <name>hadoop.kms.authentication.signer.secret.provider.zookeeper.auth.type
28    <value>kerberos</value>
29    <description>
30        The Zookeeper authentication type, 'none' or 'sas1' (Kerberos).
31    </description>
32 </property>
33 <property>
34    <name>hadoop.kms.authentication.signer.secret.provider.zookeeper.kerberos.l
35    <value>/etc/hadoop/conf/kms.keytab</value>
36    <description>
37        The absolute path for the Kerberos keytab with the credentials to
38        connect to Zookeeper.
```

```
39     </description>
40 </property>
41 <property>
42     <name>hadoop.kms.authentication.signer.secret.provider.zookeeper.kerberos.
43     <value>kms/#HOSTNAME#</value>
44     <description>
45         The Kerberos service principal used to connect to Zookeeper.
46     </description>
47 </property>
```

Delegation Tokens

TBD

KMS HTTP REST API

创建 Key

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
01 POST http://HOST:PORT/kms/v1/keys
02 Content-Type: application/json
03
04 {
05     "name"           : "<key-name>",
06     "cipher"         : "<cipher>",
07     "length"         : <length>,           //int
08     "material"       : "<material>",       //base64
09     "description"   : "<description>"
10 }
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1201 CREATED
2LOCATION: http://HOST:PORT/kms/v1/key/<key-name>
```

```
3Content-Type: application/json
4
5{
6  "name"          : "versionName",
7  "material"      : "<material>", //base64, not present without
8GET ACL
  }
```

Rollover Key

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

```
2
1 POST http://HOST:PORT/kms/v1/key/<key-name>
2 Content-Type: application/json
3
4 {
5   "material"      : "<material>",
6 }
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

```
2
1 200 OK
2 Content-Type: application/json
3
4 {
5   "name"          : "versionName",
6   "material"      : "<material>", //base64, not present without
7GET ACL
  }
```

删除 Key

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 DELETE http://HOST:PORT/kms/v1/key/<key-name>
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1          200 OK
```

获取 Key 元数据

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 GET http://HOST:PORT/kms/v1/key/<key-name>/_metadata
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
01 200 OK
02 Content-Type: application/json
03
04 {
05     "name"           : "<key-name>",
06     "cipher"        : "<cipher>",
07     "length"        : <length>,                //int
08     "description"   : "<description>",
09     "created"       : <millis-epoch>,         //long
10     "versions"      : <versions>              //int
11 }
```

获取当前 key

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 GET http://HOST:PORT/kms/v1/key/<key-name>/_currentversion
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 200 OK
2 Content-Type: application/json
3
4 {
5     "name"           : "versionName",
6     "material"       : "<material>", //base64
7 }
```

生成加密密钥的当前 **key** 版本

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 GET
1 http://HOST:PORT/kms/v1/key/<key-name>/_eek?eek_op=generate&num_keys=<number-of-key>
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
01200 OK
02Content-Type: application/json
03[
04  {
05      "versionName"           : "encryptionVersionName",
06      "iv"                     :
07"<iv>", //base64
08      "encryptedKeyVersion" : {
09          "versionName"       : "EEK",
10          "material"          :
11"<material>", //base64
12      }
13  },
14  {
```

```
15     "versionName"           : "encryptionVersionName",
16     "iv"                    :
17     "<iv>",                  //base64
18     "encryptedKeyVersion" : {
19         "versionName"       : "EEK",
20         "material"          :
21     "<material>",          //base64
22     },
23     ...
24 ]
```

解密加密 key

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
POST
1 http://HOST:PORT/kms/v1/keyversion/<version-name>/_eek?ee_op=decrypt
2 Content-Type: application/json
3
4 {
5     "name"           : "<key-name>",
6     "iv"             : "<iv>",           //base64
7     "material"       : "<material>",    //base64
8 }
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 200 OK
2 Content-Type: application/json
3
4 {
5     "name"           : "EK",
6     "material"       : "<material>",    //base64
7 }
```

获取 Key 版本

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 GET http://HOST:PORT/kms/v1/keyversion/<version-name>
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 200 OK
2 Content-Type: application/json
3
4 {
5     "name"           : "versionName",
6     "material"       : "<material>", //base64
7 }
```

获取 Key Versions

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 GET http://HOST:PORT/kms/v1/key/<key-name>/_versions
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
01 200 OK
02 Content-Type: application/json
03
04 [
05     {
06         "name"           : "versionName",
```

```
07     "material"      : "<material>",      //base64
08   },
09   {
10     "name"           : "versionName",
11     "material"      : "<material>",      //base64
12   },
13   ...
14 ]
```

获取 **Key** 名字

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 GET http://HOST:PORT/kms/v1/keys/names
```

响应

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 200 OK
2 Content-Type: application/json
3
4 [
5   "<key-name>",
6   "<key-name>",
7   ...
8 ]
```

获取 **Keys** 元数据

请求

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 GET
1 http://HOST:PORT/kms/v1/keys/metadata?key=<key-name>&key=<key-name>,...
```

响应

[Bash shell] 纯文本查看 复制代码

?

```
01 200 OK
02 Content-Type: application/json
03
04 [
05   {
06     "name"           : "<key-name>",
07     "cipher"         : "<cipher>",
08     "length"         : <length>,                //int
09     "description"    : "<description>",
10     "created"        : <millis-epoc>,          //long
11     "versions"       : <versions>              //int
12   },
13   {
14     "name"           : "<key-name>",
15     "cipher"         : "<cipher>",
16     "length"         : <length>,                //int
17     "description"    : "<description>",
18     "created"        : <millis-epoc>,          //long
19     "versions"       : <versions>              //int
20   },
21   ...
22 ]
```

hadoop 入门:第三章 HDFS 文档概述

随着版本的改变, hadoop 官网的网址可能会发生变化,但是无论是哪个版本【2.x】,文档的作用几乎不会改变。本文是

接着 hadoop 官网帮助文档的第三章，这里根据官网的顺序，依次介绍各个章节【文档】，帮助大家认识每个文档的作用，

后期如用得着可以快速定位。

1.HDFS 的用户指南

文档简介：

使用 HDFS 既可以作为 Hadoop 集群的一部分，也可以单独作为通用的分布式文件系统。HDFS 设计在多个环境中工作，

HDFS 知识帮助改进配置和诊断指定的集群。

网址：<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html>

HDFS Users Guide

- HDFS Users Guide
 - Purpose
 - Overview
 - Prerequisites
 - Web Interface
 - Shell Commands
 - DFSAdmin Command
 - Secondary NameNode
 - Checkpoint Node
 - Backup Node
 - Import Checkpoint
 - Balancer
 - Rack Awareness
 - Safemode
 - fsck
 - fetchdt
 - Recovery Mode
 - Upgrade and Rollback
 - DataNode Hot Swap Drive
 - File Permissions and Security
 - Scalability
 - Related Documentation

推荐: HDFS 用户手册

<http://www.aboutyun.com/forum.php?mod=viewthread&tid=17304>

2.HDFS 命令指南

文档简介:

文档中介绍了 HDFS 所有的命令及如何使用

HDFS Commands Guide

- Overview
- User Commands
 - classpath
 - dfs
 - fetchdt
 - fsck
 - getconf
 - groups
 - lsSnapshottableDir
 - jmxget
 - oev
 - oiv
 - oiv_legacy
 - snapshotDiff
 - version
- Administration Commands
 - balancer
 - cacheadmin
 - crypto
 - datanode
 - dfsadmin
 - haadmin
 - journalnode
 - mover
 - namenode
 - nfs3
 - portmap
 - secondarynamenode
 - storagepolicies
 - zkfc
- Debug Commands
 - verify
 - recoverLease

网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html>

3.HDFS 高可用使用 QJM 【 Quorum Journal Manager】

文档简介:

本指南提供了 HDFS 高可用性 (HA) 功能, 以及如何配置和管理 HA HDFS 集群, 使用 QJM【 Quorum Journal Manager】

功能的概述。

本文假设读者对通用组件和节点类型在 HDFS 集群一个大致的了解。

HDFS High Availability Using the Quorum Journal Manager

- HDFS High Availability Using the Quorum Journal Manager
 - Purpose
 - Note: Using the Quorum Journal Manager or Conventional Shared Storage
 - Background
 - Architecture
 - Hardware resources
 - Deployment
 - Configuration overview
 - Configuration details
 - Deployment details
 - Administrative commands
 - Automatic Failover
 - Introduction
 - Components
 - Deploying ZooKeeper
 - Before you begin
 - Configuring automatic failover
 - Initializing HA state in ZooKeeper
 - Starting the cluster with start-dfs.sh
 - Starting the cluster manually
 - Securing access to ZooKeeper
 - Verifying automatic failover
 - Automatic Failover FAQ
 - HDFS Upgrade/Finalization/Rollback with HA Enabled

网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html>

4.HDFS 高可用【NFS】

文档简介:

这个文档提供了一个 HDFS HA 功能概述和如何管理配置一个 HDFS HA 集群，使用 NFS 共享存储的 NameNodes 节点要求。

注意:

使用 QJM 或则常规共享存储

这个向导描述了如何配置和使用 HDFS HA，使用共享 NFS 目录共享共享 edit 日志在 Active 和 Standby NameNodes。

关于如何配置 HDFS HA 使用 Quorum Journal Manager，不使用 NFS，请看

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html>

HDFS High Availability

- HDFS High Availability
 - Purpose
 - Note: Using the Quorum Journal Manager or Conventional Shared Storage
 - Background
 - Architecture
 - Hardware resources
 - Deployment
 - Configuration overview
 - Configuration details
 - Deployment details
 - Administrative commands
 - Automatic Failover
 - Introduction
 - Components
 - Deploying ZooKeeper
 - Before you begin
 - Configuring automatic failover
 - Initializing HA state in ZooKeeper
 - Starting the cluster with start-dfs.sh
 - Starting the cluster manually
 - Securing access to ZooKeeper
 - Verifying automatic failover
 - Automatic Failover FAQ
 - BookKeeper as a Shared storage (EXPERIMENTAL)

网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithNFS.html>

5.HDFS Federation

文档简介:

这个向导提供了 HDFS Federation 功能概述和如何配置和管理 federated 集群

Federation 可以理解为联盟

HDFS Federation

- HDFS Federation
 - Background
 - Multiple Namenodes/Namespace
 - Key Benefits
 - Federation Configuration
 - Configuration:
 - Formatting Namenodes
 - Upgrading from an older release and configuring federation
 - Adding a new Namenode to an existing HDFS cluster
 - Managing the cluster
 - Starting and stopping cluster
 - Balancer
 - Decommissioning
 - Cluster Web Console

网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/Federation.html>

6.视图文件系统指南

文档简介:

ViewFs（视图文件系统）提供一种方式管理多个文件系统命令空间（或则 namespace 卷）。它是非常有用的特别对于多

个 namenode ， 所 以 多 个 namespaces ， 在 HDFS 联 盟

（ <http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/Federation.html> ） . ViewFs 类似于一些

Unix/Linux 操作系统的客户端安装列表。

本指南描述了，在有多个集群的 Hadoop 系统中，每一个集群都可能联合起来形成多个命名空间。也描述了如何在联邦

的 HDFS 中用 ViewFs 为每一个集群提供一个全局的命名空间，以使应用程序可以以类似于联邦之前的方式运行。

相关：

hadoop 视图文件系统指南

<http://www.aboutyun.com/forum.php?mod=viewthread&tid=17303>

7.HDFS 快照（HDFS Snapshots）

文档简介：

HDFS 快照是一个只读的基于时间点文件系统拷贝。快照可以是整个文件系统的也可以是一部分。常用来作为数据备份，

防止用户错误和容灾。

HDFS 实现了：

- Snapshot 创建的时间复杂度为 $O(1)$ ，但是不包括 INode 的寻找时间
- 只有当修改 SnapShot 时，才会有额外的内存占用，内存使用量为 $O(M)$ ，M 为修改的文件或者目录数
- 在 datanode 上面的 blocks 不会复制，做 Snapshot 的文件是纪录了 block 的列表和文件的大小，但是没有数据的复制
- Snapshot 并不会影响 HDFS 的正常操作：修改会按照时间的反序记录，这样可以直接读取到最新的数据。快照数据是当前数据减去修改的部分计算出来的。

中文参考:

HDFS 快照 (HDFS Snapshots)

<http://www.aboutyun.com/thread-17305-1-1.html>

网址:

<http://hadoop.apache.org/docs/r2.../HdfsSnapshots.html>

8.HDFS 架构

文档简介:

Hadoop 分布式文件系统(HDFS)是一个设计运行在通常的硬件机器上的分布式文件系统。它与已存在的分布式文件系统

有许多相似性。但是,与其它系统的不同之处也很重要。HDFS 是一个高容错性系统,被设计成可以运行在廉价硬件上。

HDFS 可提供高吞吐量,适合于那些具有大数据集的应用场合。HDFS 放宽了一些 POSIX 要求,以适应流式存取文件数

据。HDFS 最初是作为 Apache Nutch web 搜索引擎项目的基础构件来开发的。现在 HDFS 是 Apache Hadoop 的核心项

目,项目 URL 为 <http://hadoop.apache.org/>。

中文参考: HDFS 架构(Apache Hadoop 2.1.1-beta)

<http://www.aboutyun.com/forum.php?mod=viewthread&tid=17306>

英文网址: <http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html>

9. 离线 Edits 阅读器指南

文档简介:

离线的 Edits 查看器是一个解析 Editslog 文件的工具。当前程序需要用于不同格式的转换, 包括可读的和比本地二进制

格式更易编辑 XML 文件。

此工具可以解析 Hadoop0.19 和机器之后版本的 edits 格式。这个工具只操作文件, 不需要 Hadoop 集群正在运行。

英文网址:

<http://hadoop.apache.org/docs/r2...dfsEditsViewer.html>

10. 离线 Image 阅读器指南

文档简介:

离线 Image 阅读器是一个转存 `hdfs fsimage` 文件的内容为可阅读的格式, 提供只读的 WebHDFS API, 允许离线分析和检查 Hadoop 集群的 namespace。这个工具能够处理大的 image 文件相对的较快。工具处理 layout 格式包括 hadoop2.4 及以上版本。如果想处理旧 layout 格式, 你可以使用 `hadoop2.3` 或则 `oiv_legacy` 命令离线 Image 阅读器。如果工具不能处理 image 文件, 它将退出。离线 Image 阅读器不需要集群运行, 它完全处于脱机状态。

英文网址: <http://hadoop.apache.org/docs/r2...dfsImageViewer.html>

11.HDFS 权限指南

文档简介:

hadoop 分布式文件系统 (HDFS) 实现了权限模型为文件和目录, 共享大部分 POSIX 模型。每个文件和目录都有拥有者和所属组。文件或则目录对于所属用户有单独的权限, 对于组成员其它用户, 和其它所有用户。对于文件, `r` 权限是读文件, `w` 权限是写或则追加文件。目录, `r` 权限列出目录的内容, `w` 权限可以创建和删除文件或则目录, `x` 权限允许访问子目录。

网址:

[http://hadoop.apache.org/docs/r2 ... rmissionsGuide.html](http://hadoop.apache.org/docs/r2...rmissionsGuide.html)

12.HDFS 配额指南

文档简介:

hadoop 分布式文件系统允许管理员设置已使用的 name quotas,和 Space Quotas 为单独的目录。Name quotas 和 space

quotas 单独操作, 但是 administration 和 implementation 是两种类型的并行的配额

网址: <http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsQuotaAdminGuide.html>

13.hftp 指南

文档简介:

HFTP 是一个 hadoop 文件系统实现, 让你读取数据从远程 hadoop hdfs 集群。读取通过 HTTP, 和数据源是 DataNodes。

HFTP 是一个只读的文件系统, 如果尝试使用写数据或则修改文件系统状态会抛出异常。

网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/Hftp.html>

14.C API libhdfs

文档简介:

libhdfs 是一个 JNI ， 基于 C API 的 hadoop 分布式文件系统（HDFS）。它提供 HDFS APIs 的一个子集操作 HDFS 文件

和系统。libhdfs 是 hadoop 分布式的部分，来自预编译在\$HADOOP_HDFS_HOME/lib/native/libhdfs.so .libhdfs 是与

windows 兼容的，可以在 window 上运行 mvn 编译构建，在 hadoop-hdfs-project/hadoop-hdfs 源码树目录。

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/LibHdfs.html>

15.WebHDFS REST API

文档简介:

HTTP REST API 支持完整的 FileSystem/FileContext HDFS 接口。

英文网址:

[http://hadoop.apache.org/docs/r2 ... p-hdfs/WebHDFS.html](http://hadoop.apache.org/docs/r2...p-hdfs/WebHDFS.html)

16.HttpFS Gateway

文档简介:

Hadoop HDFS 通过 HTTP - 文档集

HttpFS 是 NameNode 的单独服务。HttpFS 是一个 Java web 应用程序和运行使用预配置的 Tomcat，捆绑 HttpFS 二进制分布

HttpFS 有很多功能，比如读写数据，传输数据等。

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-hdfs-httpfs/index.html>

17. Short-Circuit 本地读取

文档简介:

在 HDFS，读取通常通过 DataNode。因此，当客户端请求 DataNode 读取一个文件，DataNode 读取文件从磁盘，发送数据通过 TCP socket 到客户端。所谓的“short-circuit”的读取，绕过 DataNode，允许客户端读取文件目录。很明显，

唯一可能的情况是客户端与数据共定位。Short-circuit 提升了许多应用的读取性能。

英文网址: <http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/ShortCircuitLocalReads.html>

18. HDFS 集中缓存管理

文档简介:

在 HDFS 集中缓存管理是一个显式缓存机制, 允许用户指定要缓存的 HDFS 路径。NameNode 与 DataNodes 通信, 在

磁盘上有请求的 blocks, 指导他们在 off-heap 来缓存 blocks。

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/CentralizedCacheManagement.html>

19. HDFS NFS Gateway

文档简介:

NFS Gateway 支持 NFSv3 和允许安装的 HDFS 作为客户端本地文件系统部分。

NFS gateway 机器需要同样的事情运行一个 HDFS 客户端, 像 Hadoop JAR 文件, HADOOP_CONF 目录。NFS gateway 可

以在相同的客户端与 DataNode, NameNode, 或则任何 HDFS client.

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsNfsGateway.html>

20. HDFS 滚动升级

文档简介:

HDFS 滚动升级允许单独升级 HDFS 守护进程。比如, datanodes 可以独立于 namenode 升级。一个 namenode 可以

独立于其它 namenodes 升级。namenodes 可以独立于 datanods 和 journal nodes 升级

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsRollingUpgrade.html>

21.HDFS 扩展属性

文档简介:

扩展属性 (abbreviated as xattrs) 是文件系统功能, 允许用户应用程序将附加元数据与文件或目录关联起来。不像系统

级别 inode 元数据比如文件权限或则修改时间, 扩展属性系统没有解释, 取而代之的是应用程序用于存储索引节点的附

加信息。例如, 可以使用扩展属性来指定一个纯文本文档的字符编码。

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/ExtendedAttributes.html>

22.HDFS 透明加密

文档简介:

HDFS 实现透明的、端到端的加密。一旦配置, 数据读取和写入指定 HDFS 目录是透明加密, 解密不需要改变用户应用

程序代码。这个加密也是端到端, 这意味着数据仅被客户端加密和解密。HDFS 不会存储或则访问加密数据或则加密数

据加密 keys。这满足了 2 个典型的加密要求: 在其它加密 (关于持久 media 数据, 比如磁盘) 以及传输加密。(等等当

数据在网络上传输)

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/TransparentEncryption.html>

23.HDFS 支持 Multihomed 网络

文档简介:

这个文档是针对集群管理员部署 HDFS 在 Multihomed 网络。类似支持 YARN/MapReduce

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsMultihoming.html>

24.档案存储、SSD 和内存

档案存储是一种解决日益增长的存储容量,高密度节点和低的存储与低的计算能力成为可用,可以被用做冷存储在集群中。

基于策略的数据可以从热存储移到冷存储。添加更多的节点冷存储增加存储,独立于集群的计算。

通过异构存储和归档存储提供的框架概括了 HDFS 的架构包括其他类型的存储介质包括 SSD 和内存。

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/ArchivalStorage.html>

25.Memory 存储支持 HDFS

HDFS 支持写了堆内存的数据节点进行管理。Data Nodes flush 内存数据到磁盘异构。因此移除昂贵的磁盘 IO 和

checksum 计算从 performance-sensitive IO 路径，因此我们称之为懒惰持久写。HDFS 提供最大努力的坚持保证懒持

久写。在复制前的一个节点重新启动时，很少有数据丢失是可能的。数据丢失有可能的在节点重启时，副本持久化磁盘。

应用可以选择使用懒惰持久写，交换一些耐久性保证，有利于减少延迟。

英文网址：

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/MemoryStorage.html>

hadoop 入门:第四章 mapreduce 文档概述

1.MapReduce 教程

文档简介

这个文档描述所有用户认识 hadoop mapreduce 框架和服务

英文网址

[http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.h](http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html)

[tml](#)

2.MapReduce 命令指南

文档作用

所有的 mapreduce 命令通过 bin/mapred 脚本调用。运行 mapred 脚本没有任何参数打印所有命令的描述。

英文网址:

[http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapredCommands.ht](http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapredCommands.html)

[ml](#)

3. 迁移从 Hadoop 1.x 到 Hadoop 2.x

这个文档提供信息为用户迁移 MapReduce 应用程序从 Hadoop 1.x 到 Hadoop 2.x.

英文网址

http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduce_Compatibility_Hadoop1_Hadoop2.html

[bility_Hadoop1_Hadoop2.html](http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduce_Compatibility_Hadoop1_Hadoop2.html)

4.Hadoop: 加密的 Shuffle

加密 Shuffle 能力允许 MapReduce shuffle 加密使用 HTTPS 和带有选项的客户端身份认证 (HTTPS 双向认证或则 HTTPS

客户端认证)

英文网址:

[http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/EncryptedShuffle.ht](http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/EncryptedShuffle.html)

[ml](http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/EncryptedShuffle.html)

5.Hadoop 的: 可插拔 Shuffle 和可插拔排序

可插拔的 Shuffle 和排序能力允许可插拔更换内置的 Shuffle 和排序逻辑备用的实现。

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/PluggableShuffleAndPluggableSort.html>

6.Hadoop 的分布式缓存部署

MapReduce 应用框架初步支持部署新版本的 MapReduce framework 通过分布式缓存。通过设置适当的配置属性，用户可以运行不同版本的 MapReduce 比最初部署到集群。举例，集群管理员可以放置多个版本的 MapReduce 在 HDFS 和配置 `mapred-site.xml` 指定哪个版本的 jobs 使用默认。这个允许管理员执行滚动升级 MapReduce 框架在一定条件下。

<http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/DistributedCacheDeployment.html>

hadoop 入门:第五章 MapReduce REST APIs 文档概述

1.MapReduce 应用程序 Master REST API's.

文档简介

MapReduce 应用程序 Master REST API's 允许用户获取运行的 MapReduce 应用程序 master 上的状态。这相当于运行 MapReduce job。这个信息包括运行 jobs 应用 master 和所有 job 详情比如任务, 计数器, 配置, attempts, 等.这个应用

master 应该通过 proxy 访问。这个代理配置既可以是 resource manager 或则在一个单独的主机上。这个 proxy URL 通

常 `http://<proxy http address:port>/proxy/appid`.

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapredAppMasterRest.html>

2.MapReduce History Server REST API's.

文档简介

history server REST API's 允许用户获取完成的应用的状态

英文网址:

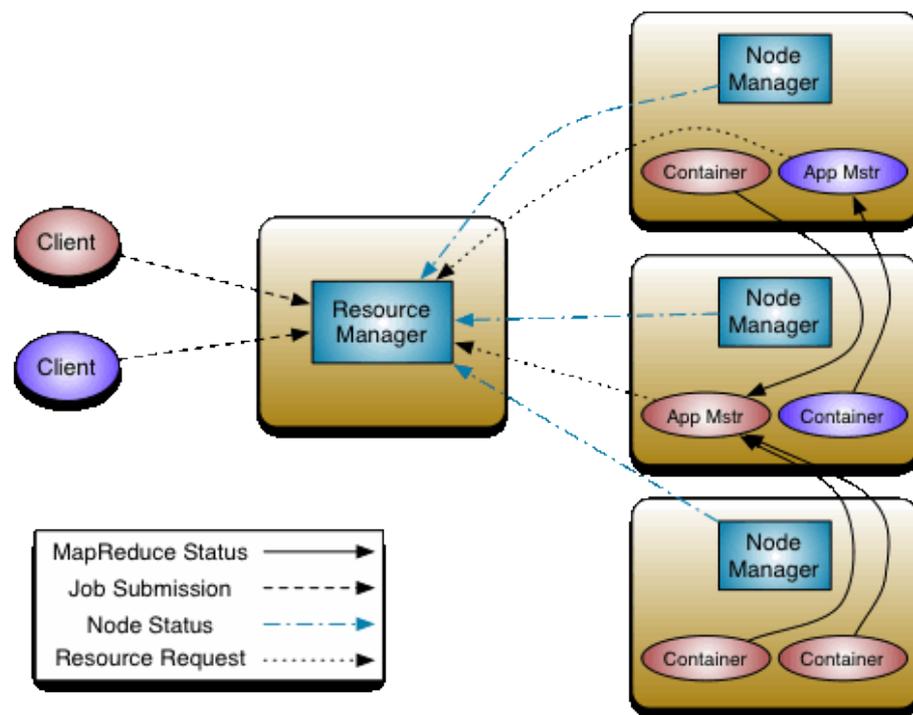
<http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-hs/HistoryServerRest.html>

hadoop 入门:第六章 YARN 文档概述

1.YARN 结构

文档简介:

Yarn 的基本思想是拆分资源管理的功能，作业调度/监控到单独的守护进程



英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html>

2.Hadoop: Capacity 调度

文档简介

本文档介绍了 capacityscheduler, hadoop 可插拔 scheduler 允许多租户安全共享大的集群, 他们的应用在分配的能力约束下及时分配资源。

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html>

3.Hadoop: Fair 调度

文档简介

这个文档描述了 FairScheduler, hadoop 可插拔 scheduler 允许 yarn 应用 fairly 共享资源在大集群。

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/FairScheduler.html>

#####

上面两个文档详细介绍了 hadoop 调度方式

4.ResourceManger 重启

文档简介

ResourceManager 是中央权威，管理资源和调度应用运行 YARN 。因此，潜在单点故障在 YARN 集群。本文给出了概述，重启 ResourceManager，一个功能提高 ResourceManager 保持运行，对于终端用户使 ResourceManager 宕机时间不可见。

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/ResourceManagerRestart.html>

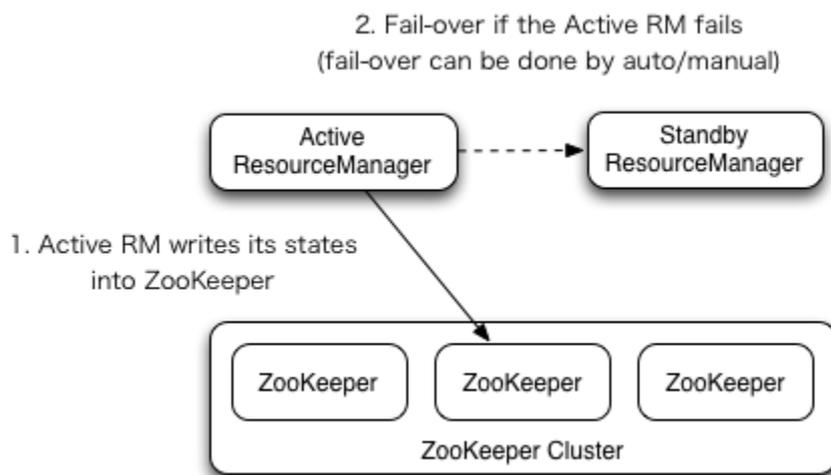
5.ResourceManager 高可用

文档简介

本指南提供了 Yarn 的 ResourceManager 高可用性概述，和详细描述了如何配置和使用这个功能。ResourceManager (RM) 负责跟踪资源集群，和调度应用（等 MapReduce jobs）。Hadoop 2.4 之前，YARN 集群有单点故障。这个高可用功能添

加冗余在一个 Active/Standby ResourceManager 队的形式，移除这个，否则就是单点故障。

结构：



英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/ResourceManagerHA.html>

6.YARN 节点 Labels

文档简介

节点标签是一种具有相似特性的组节点的方式，应用程序可以指定运行的地方。

英文网址：

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/NodeLabel.html>

7.Web 应用代理

文档简介

web 应用代理是 Yarn 一部分。默认它运行作为 Resource Manager(RM)一部分，但是可以被配置为运行单机模式

英文网络:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/WebApplicationProxy.html>

8.YARN Timeline Server

文档简介

存储和检索当前应用和历史信息以通用的方式在 Yarn 处理，通过 Timeline Server

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/TimelineServer.html>

9.Hadoop: Writing YARN 应用

文档简介

本文档描述了一个高层次的方法来实现 Yarn 的新应用。

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/WritingYarnApplications.html>

10.YARN 命令

文档简介

YARN 命令被 bin/yarn 脚本调用。运行 yarn 没有任何脚本，打印所有命令描述

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YarnCommands.html>

11.NodeManager 重启

文档简介

本文给出了 nodemanager 概述(nm)重启, 一个功能, 启用 NodeManager, 重启不会丢失激活的运行在节点的 containers。

在高水平, NM 存储任何需要的状态在本地状态存储, 因为它处理容器管理需求。当 NM 重启, 它首先恢复加载各个子

系统状态和然后让这些子系统执行恢复使用加载状态。

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/NodeManagerRestart.html>

12.Docker Container Executor

文档简介

Docker 包括易于使用的接口与 Linux 容器使用易于构建镜像文件为这些容器。总之，Docker 是一个很轻的虚拟机。

Docker Container Executor (DCE) 允许 YARN NodeManager 启动 Yarn 容器到 Docker 容器。用户可以指定 Docker 镜

像，他们想要他们的 Yarn 容器。这些容器提供定制的软件环境，软件环境中用户代码运行，隔离 NodeManager 软件环

境。这些容器可以包含应用程序所需要的特殊库，和安装在 NodeManager 的软件，有不同版本的 Perl, Python,和甚至

Java 。确实，这些容器可以运行不同的 Linux flavor 比运行在 NodeManager 的，尽管如此，YARN container 必须定

义运行 job 所需要的所有的环境和 libraries ，不会与 NodeManager 共享。

YARN Docker 提供二者一致性（所有 YARN 容器将会有相同的软件环境）和隔离（与物理机安装不会发生冲突）

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/DockerContainerExecutor.html>

13.使用 YARN 的 CGroups

文档简介

CGroups 是一种聚集/划分任务组的机制

CGroups 是一个 Linux 内核的功能，并入内核版本 2.6.24。从 Yarn 的角度，这允许容器限制他们的资源使用。

一个很好的例子就是 CPU 的使用。没有 CGroups，它变的很难限制容器 CPU 的使用。当前，CGroups 仅用于限制 CPU 使用。

英文网址

[http://hadoop.apache.org/docs/r2 ... ManagerCgroups.html](http://hadoop.apache.org/docs/r2... ManagerCgroups.html)

14.Yarn 安全容器

文档简介

一个安全的集群中的 Yarn 容器使用操作系统设备为容器提供执行隔离。安全容器执行在 job 用户凭据下。操作系统强制

访问容器的限制。容器必须运行使用提交应用程序。

安全的容器只有在有安全的 Yarn 的背景下工作。

英文网址

[http://hadoop.apache.org/docs/r2 ... ecureContainer.html](http://hadoop.apache.org/docs/r2... ecureContainer.html)

15.YARN Service 注册

服务注册是一种服务，可以部署在 Hadoop 集群,允许部署的应用程序注册自己的方式与他们通信。客户端应用程序可以

使用绑定信息 连接网络访问端点，他们 REST, IPC, Web UI, Zookeeper quorum+path 或则其它协议

英文网址

<http://hadoop.apache.org/docs/r2...registry/index.html>

hadoop 入门:第七章 YARN REST APIs

REST 推荐

Hadoop web 编程--REST API WebHDFS

<http://www.aboutyun.com/forum.php?mod=viewthread&tid=8823>

1.Hadoop YARN - 介绍 web 服务 REST API's

文档简介

web 服务 REST APIs 是一组 URI 资源，用来访问集群，节点，应用程序和应用程序历史信息。URI 资源被分组为基于

返回信息的类型的 API。一些资源返回集合，一些资源返回单例。

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/WebServicesIntro.html>

2.ResourceManager REST API's.

ResourceManager REST API's 允许用户获取关于集群的信息--集群的状态，集群指标，调度信息，有关集群中节点的信息，有关群集上的应用程序的信息。

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/ResourceManagerRest.html>

3.NodeManager REST API's

NodeManager REST API's 允许用户获取节点状态和关于应用程序信息和运行节点的容器

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/NodeManagerRest.html>

4.YARN Timeline Server

当前应用程序的存储和检索和历史信息以通用的方式在 Yarn 处理通过 Timeline Server。

英文网址

http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/TimelineServer.html#Timeline_Server_REST_API

[I v1](#)

hadoop 入门:第八章 hadoop 兼容文件系统

1.Hadoop-AWS 模块: 整合 AmazonWeb 服务

文档简介

hadoop-aws 模块, 提供支持 AWS 整合。生成的 jar 文件, hadoop-aws.jar 还声明了可传递的 dependency 在外部 artifacts,

这是需要支持的--使下游的应用程序可以很容易地使用这种支持。

网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-aws/tools/hadoop-aws/index.html>

2.Hadoop Azure 支持: Azure Blob 存储

文档简介

hadoop-azure 模块提供支持整合 Azure Blob 存储.构建 jar 文件, 命名 hadoop-azure.jar, 还声明了它所需的额外的 artifacts 的可传递依赖关系, 特别是 Azure 存储 java SDK。

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-azure/index.html>

3.Hadoop 支持 OpenStack: Swift 对象存储

OpenStack 是一个开源的云基础设施, 可以从多个公共 IaaS 提供商访问, 和私有部署。它提供基础设施服务, 如虚拟

机主机 (Nova), 认证 (Keystone) 和存储的二进制对象 (Swift)。

这个模块启用 hadoop 应用程序-包括 MapReduce jobs, 读取和写数据从 OpenStack Swift 对象存储实例

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-openstack/index.html>

hadoop 入门:第九章 hadoop 认证

1.Hadoop 认证, Java HTTP SPNEGO - 例子

文档简介:

SPNEGO :

SPNEGO(SPNEGO: Simple and Protected GSS-API Negotiation)是微软提供的一种使用 GSS-API 认证机制的安全协议,

用于使 Webserver 共享 Windows Credentials, 它扩展了 Kerberos(一种网络认证协议)。

使用浏览器访问 hadoop 认证受保护的 URL

重要:

浏览器必须支持 HTTP Kerberos SPNEGO。比如 Firefox 或则 Internet Explorer.

英文网址:

<http://hadoop.apache.org/docs/r2.7.2/hadoop-auth/Examples.html>

2.Hadoop 认证, Java HTTP SPNEGO - 服务端配置

文档简介:

服务端配置设置

认证过滤器是 hadoop 认证服务端配置。

此筛选器必须配置在所需身份验证请求的所有网络应用程序资源的前面

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-auth/Configuration.html>

3.Hadoop 认证, Java HTTP SPNEGO - 构建

文档简介:

要求:

Java 7+

Maven 3+

Kerberos KDC (for running Kerberos test cases)

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-auth/BuildingIt.html>

hadoop 入门:第十章 hadoop 工具

问题导读

1.hadoop 有哪些工具?

2.hadoop 流的作用是什么？

3.hadoop 集群负载如何模拟？

4.hadoop 数据提取和分析工具是哪个？



1.Hadoop 流

文档简介

Hadoop 的数据流是自带的 Hadoop 发行版的实用程序。这个程序允许你创建和运行可执行的 Map/Reduce jobs 或则脚本，作为 mapper 或则 reducer。举例：

本，作为 mapper 或则 reducer。举例：

[Bash shell] [纯文本查看](#) [复制代码](#)

[?](#)

```
1 hadoop jar hadoop-streaming-2.7.2.jar \
```

```
2   -input myInputDirs \
```

```
3   -output myOutputDir \
```

```
4   -mapper /bin/cat \
```

```
5   -reducer /usr/bin/wc
```

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-streaming/HadoopStreaming.html>

2.Hadoop Archives 指南

文档简介

hadoop Archives 是特殊格式的 archives。一个 Hadoop archive 映射一个文件系统目录。hadoop archive 是一个*.har。

一个 hadoop Archives 目录包括元数据（以 `_index` 和 `_masterindex` 形式）和数据（`part-*`）文件。这个 `_index` 文件包

含文件的名字作为归档的一部分和部分文件的位置

英文网址

<http://hadoop.apache.org/docs/r2.7.2/hadoop-archives/HadoopArchives.html>

3. DistCp 指南

文档简介

DistCp Version 2 (分布式 copy) 是一个工具用于大的 集群内和集群间的复制。它用 mapreduce 来影响它的分布，错误处理和恢复，和报告。它扩展了文件列表和目录输入 map 的任务，每个文件分区在资源列表指定。本文档的目的是描

述新 distcp 设计

英文文档

<http://hadoop.apache.org/docs/r2.7.2/hadoop-distcp/DistCp.html>

4. Gridmix

文档简介

GridMix 是 Hadoop 集群的基准。为运行 GridMix，你需要一个 MapReduce job 跟踪描述给定集群的混合 job

英文文档

<http://hadoop.apache.org/docs/r2.7.2/hadoop-gridmix/GridMix.html>

5.Rumen

文档简介

Rumen 是 Apache 的 Hadoop 构建的数据提取和分析工具

英文文档

<http://hadoop.apache.org/docs/r2.7.2/hadoop-rumen/Rumen.html>

6.Yarn 调度负载模拟器（SLS）

文档简介

Yarn 调度器是大家都感兴趣的地方，有不同的实现，例如, Fifo, Capacity 和 Fair schedulers.与此同时，一些优化也作

出改善不同场景和工作负载调度程序的性能。每个调度器算法有其自身的功能集，驱动调度收到许多影响，比如 fairness,

capacity 能力, 资源可用性, 等.这非常重要评估调度算法是好的，在我们部署生产集群之前。一个真正的集群评估始终

是时间和成本消耗，但是比较难以找到一个足够大的集群，因此,模拟器可以预测一个调度器算法对于一些特定的工作负

载将会很有用。

英文文档

<http://hadoop.apache.org/docs/r2.7.2/hadoop-sls/SchedulerLoadSimulator.html>

hadoop 入门:第十一章 hadoop 配置

我们配置集群不同配置文件有不同的配置项，下面是不同的配置文件的配置项

1.core-default.xml

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/core-default.xml>

2.hdfs-default.xml

<http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml>

3.mapred-default.xml

<http://hadoop.apache.org/docs/r2.7.2/hadoop-mapreduce-client/hadoop-mapreduce-client-core/mapred-default.xml>

4.yarn-default.xml

<http://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-common/yarn-default.xml>

相关链接:

[hadoop 入门-第一章 General: 第一节单节点伪分布](#)

[hadoop 入门-第一章 General: 第二节集群配置](#)

[hadoop 入门-第一章 General: 第三节 Hadoop 初级入门之命令指南](#)

[hadoop 入门-第一章 General: 第四节文件系统 shell](#)

[hadoop 入门-第一章 General: 第五节 hadoop 的兼容性说明](#)

[hadoop 入门-第一章 General: 第六节开发人员和用户接口指南: hadoop 接口分类](#)

[hadoop 入门-第一章 General: 第七节 Hadoop 文件系统 API : 概述](#)

[hadoop 入门-第二章 common: 第一节 hadoop 本地库 指南](#)

[hadoop 入门-第二章 common: 第二节 hadoop 代理用户 -超级用户代理其它用户](#)

[hadoop 入门-第二章 common: 第三节机架智能感知](#)

[hadoop 入门-第二章 common: 第四节安全模式说明](#)

[hadoop 入门-第二章 common: 第五节服务级别授权指南](#)

[hadoop 入门-第二章 common: 第六节 Hadoop HTTP web-consoles 认证机制](#)

[hadoop 入门-第二章 common: 第七节 Hadoop Key 管理服务器\(KMS\) - 文档集](#)

[hadoop 入门:第三章 HDFS 文档概述 \(一\)](#)

[hadoop 入门:第三章 HDFS 文档概述 \(二\)](#)

[hadoop 入门:第四章 mapreduce 文档概述](#)

[hadoop 入门:第五章 MapReduce REST APIs 文档概述](#)

[hadoop 入门:第六章 YARN 文档概述](#)

[hadoop 入门:第七章 YARN REST APIs](#)

[hadoop 入门:第八章 hadoop 兼容文件系统](#)

[hadoop 入门:第九章 hadoop 认证](#)

[hadoop 入门:第十章 hadoop 工具](#)

[hadoop 入门:第十一章 hadoop 配置](#)

about 云介绍

为热爱云开发技术人员提供最全面的信息传播和服务平台，为大家提供云技术文档，视频、云技术学习指导，解疑等。

内容包括：**hadoop** 视频，**Nosql**,虚拟化，**OpenStack**，云平台等相关技术。

about 云，本着活到老学到老的精神，为了广大云技术爱好者获取更多知识，在文章开头，都有几个问题，因此 **about**

云亦为学问社区。同样我们准备了每日一读，为了就是每天进步一点点，每天能够学到新的内容。

关注本站:

1.欢迎加入 **about** 云群

大数据群 **432264021**、**425860289**

云技术 **openstack** 群 **322273151**

2.关注腾讯认证空间

[about 云腾讯认证空间](#)

3.惯用手机的用户可关注 **about** 云微信地址:

搜索:

www.aboutyun.com



4.关注微博:

[新浪微博](#)



5.邮件订阅

[邮件订阅](#)

捐助：大数据、云技术视频

[hadoop 生态系统零基础入门及大数据实战【后续不断更新】](#)



为何学习大数据

- 1.工资高
- 2.大数据在中小企业逐渐普及，人才需求旺盛
- 3.两年之内，大数据全面普及，不懂大数据程序员可能会被淘汰或则被迫转型

如何学习大数据

更多内容：

<https://item.taobao.com/item.htm?spm=a1z10.1-c.w4023-4627152319.4.0qf3BM&id=52041335>

[5976](#)

#####

[about 云零基础开发、部署+理论 openstack 入门视频【J版及 K版】](#)

如何学习云技术

入门云技术

对于云技术，[openstack](#) 是最火的，那么该如何入门 [openstack](#)。

我们或多或少应该了解过 [openstack](#)，它有很多的组件。

问题 1:

具体组件是什么，该如何配置，它的作用是什么，我们可能是不太清楚。

问题 2:

我们知道了 [keystone](#) 是什么，[nova](#) 是什么，该如何部署？

问题 3:

我们部署过程中，会遇到各种各样的问题，这些问题该如何解决？

问题 4:

这些问题解决之后，我们终于创建实例成功，可是 ping 不通外网，这又该如何解决？↵

上面的四个问题，相信如果你不花费一周或则两周的时间，根本是解决不了的。并且这是理想的情况下，如果非理想的情况，我们可能会花费一个月甚至更长时间。如果你亲自学习部署过 [openstack](#)，相信会有自己的体验。↵

特别是创建实例，不能 ping 通外网，这个根本没有错误，如果不懂网络，无从下手。↵

about 云 [openstack juno](#) 解决了上面问题，如果完全按照视频，认识了解 [openstack](#)，部署成功，并且创建实例能 ping 通外网，这完全没有问题的。↵

about 云其它文档

链接: <http://pan.baidu.com/s/1gd00aEv> 密码: dr4h