第8章

管理Hadoop

8

本章内容

- 生产系统的配置
- 维护HDFS文件系统
- 建立作业调度器

遵照第2章中的安装说明可以很快建立一个运行的Hadoop集群。这种配置较为简单，但是它不能胜任生产集群中持久而繁重的工作。有许多不同的配置参数可用于生产集群的调优，8.1节会介绍这些参数。

此外，任何系统都会随时间发生变化，Hadoop集群也一样。而你（或某些管理员）将不得不了解如何维护它，使之保持良好的运行状态。HDFS文件系统尤其如此。从8.2节到8.5节，我们将介绍各种标准的文件系统维护任务，如健康检查、权限设置、配额管理以及已删除文件（回收站）的恢复。从8.6节到8.10节，我们将讨论更复杂也是更为少见的管理任务，且更专注于HDFS。其中包括添加/删除节点（容量）和NameNode的故障恢复。本章最后一节将讨论调度器的建立，用以管理多个运行中的作业。

8.1 为实际应用设置特定参数值

Hadoop有各种各样的参数。它们的默认值通常仅针对于单机模式，而且致力于变得让用户使用更为方便。设置默认值时要让它们能够适用于更多的系统，且不会导致任何错误。然而，大多时候它们和生产集群对优化的需求相去甚远。表8-1显示了配置生产集群所需修改的系统属性。

表8-1 可用于生产集群调优的Hadoop属性

属性	描述	推荐值
dfs.name.dir	在NameNode的本地文件系统中，用于存储HDFS元数据的目录	/home/hadoop/dfs/name
dfs.data.dir	在DataNode的本地文件系统中，用于存储HDFS文件块的目录	/home/hadoop/dfs/data
mapred.system.dir	在HDFS中用于存储共享的MapReduce系统文件的目录	/hadoop/mapred/system

（续）

属　性	描　述	推 荐 值
mapred.local.dir	在TaskNode的本地文件系统中，用于存储临时数据的目录	
mapred.tasktracker.{map|reduce}.tasks.maximum	在一个TaskTracker上可同时运行的map和reduce任务的最大值	
hadoop.tmp.dir	Hadoop临时目录	/home/hadoop/tmp
dfs.datanode.du.reserved	DataNode应具备的最小空闲空间	1073741824
mapred.child.java.opts	分配给每个子任务的堆栈大小	-Xmx512m
mapred.reduce.tasks	一个作业的reduce任务个数	

dfs.name.dir和dfs.data.dir默认值指向/tmp目录，这个目录在几乎所有的Unix系统中仅用来存储临时的数据。在生产集群中，这些属性必须做修改。[①] 此外，这些属性可以被设为一个以逗号分隔的目录列表。对于dfs.name.dir，多个目录可以更好地实现备份。如果DataNode有多个磁盘驱动器，应该在每一个磁盘上建立一个数据目录，并全部列在dfs.data.dir中。DataNode将通过并行访问来提高I/O性能。[②]你还应该把来自多个磁盘的目录列表指向mapred.local.dir，以加快临时数据的处理。

hadoop.tmp.dir为Hadoop的临时目录，其默认配置依赖于用户名。因为提交作业的用户名会和启动Hadoop节点的用户名可能并不一致，你应该避免任何Hadoop属性与用户名相关。应将其设置为/home/hadoop/tmp这样的目录，以独立于任何用户名。使用hadoop.tmp.dir默认值的另一个问题是它指向/tmp目录。虽然这个位置适用于临时存储数据，但是Linux的大部分默认配置给/tmp的配额对Hadoop而言太小了。如果不去增加/tmp的配额，最好让hadoop.tmp.dir指向一个已知有很多空间的目录。

默认配置下，HDFS不需要在DataNode上预留可用空间。但在实践中，大多数系统在可用空间太低时稳定性就会出问题。你应通过设置dfs.datanode.du.reserved在DataNode中预留1 GB的可用空间。当空闲的空间低于预留量时，DataNode会停止接受数据块的写入。

可以为每个TaskTracker配置最多被允许运行的map和Reduce任务数。Hadoop的默认值为4个任务（2个map任务和2个reduce任务）。合适的数量取决于许多因素，虽然大多数情况下每个核仅会调用1到2个任务。你可以设置一个四核机器的map和reduce任务数最多为6个（每种各3个），因为TaskTracker和DataNode已经分别有一个任务，总计为8个。同样，你可以设置双四核机器的map和reduce任务最多为14，不过这是基于大多数MapReduce作业有很重的I/O负载。如果你希望负载为CPU密集的，就应该减少允许的最大任务数。

---

① /tmp的使用印证了默认值是为操作简单而设的。每个Unix系统都有/tmp目录，这样就不会遇到“目录找不到”的错误了。

② 在Hadoop论坛上曾讨论过是否应将DataNode上的多个硬盘驱动器配置为RAID或JBOD。Hadoop并不需要RAID的数据冗余，因为HDFS已经在计算机之间复制了数据。此外，雅虎已表示它们能够使用JBOD获得明显的性能改进。因为，即使是相同型号的硬盘，它们的速度也有很大差异。RAID配置会使I/O性能受限于最慢的驱动器。另外，让每个驱动器的功能独立，将允许其以最高速度运行，使系统的整体吞吐量更高。

在考虑允许的任务数时，还应考虑分配给每个任务的堆栈大小。Hadoop默认给每个任务200 MB是远远不够的。很多配置将它提升为512 MB，有些甚至在1 GB。这并非一个终极属性。每个作业给一个任务所请求的堆栈空间都可以或多或少。你需要确保在机器上有足够的可用内存适应这些配置参数。请记住DataNode和TaskTracker每个都已经占用了1 GB的RAM。

虽然你可以在每个单独的MapReduce作业中设置reduce任务的数量，最行之有效的办法还是让默认值能够适应大部分时间的工作。Hadoop默认一个作业一个reduce任务，在大多数情况下当然不够理想。通常建议把默认值设为集群中运行reduce的TaskTracker的最大值的0.95倍或1.75倍。这意味着在一个作业中reduce任务的数量应等于0.95或1.75乘以工作节点数，再乘以mapred.tasktracker.reduce.tasks.maximum。因子为0.95时会让所有的reduce任务立即装载，并当map任务结束时复制它们的输出结果。当因子为1.75时，一些reduce任务会被立即装载，而其他一些则会等待。更快的节点会较早地完成第一轮reduce任务，并开始第二轮。最慢的节点在第二轮不需要处理任何reduce任务。这可以带来更好的负载平衡。

8.2 系统体检

Hadoop提供的文件系统检查工具叫做fsck。如参数为文件路径时，它会递归地检查该路径下所有文件的健康状态。如果参数为/，它就会检查整个文件系统。如下为一个输出的示例：

```
bin/hadoop fsck /
Status: HEALTHY
 Total size:    143106109768 B
 Total dirs:    9726
 Total files:    41532
 Total blocks (validated):      42419 (avg. block size 3373632 B)
 Minimally replicated blocks:   42419 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
 Average block replication:     3.0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          8
 Number of racks:               1
```

大多数信息是不言而喻的。默认情况下，fsck会忽略正在被客户端写入而打开的文件。如果你想看到这些文件的列表，可以在fsck使用-openforwrite参数。

在fsck检查文件系统时，它每发现一个健康的文件就会打印出一个圆点（未在上面的输出中显示）。遇到不健康的文件时，它会打出相应信息，包括过度复制的块、复制不足的块、未复制的块、损坏的块和失踪的副本。因为HDFS是自我修复的，所以过度复制的块、复制不足的块、未复制的块都不足为虑。但是，损坏的块和失踪的副本意味着数据已永久丢失。默认情况下，fsck对损坏的文件什么也不做，但你可以运行fsck的-delete选项将其删除。但更好的方式是用-move选项运行fsck，它会把已损坏的文件移动到/lost+found目录中备用。

你还可以在fsck中加上-files、-blocks、-locations和-racks选项，以打印出更多的信息。每个后续的选项需要前面选项的存在。-blocks需要-files；-locations既要有-files，也要有-blocks，以此类推。-files选项让fsck每检查一个文件便打印一行信息，包含文件路径、文件字节数与块个数，以及文件的状态。-blocks选项进一步让fsck为文件中的每个块打印出一行信息。该行包括块名、长度及其副本数。-locations选项会让每一行包含块的副本位置。-racks选项则会在位置信息中增加机架名。例如，一个小到仅有一个数据块的文件会给出如下的报告

```
bin/hadoop fsck /user/hadoop/test -files -blocks -locations -racks
/user/hadoop/test/part-00000 35792 bytes, 1 block(s):  OK
0. blk_-4630072455652803568_97605 len=35792 repl=3
➥ [/default-rack/10.130.164.71:50010, /default-rack/10.130.164.177:50010,
➥ /default-rack/10.130.164.186:50010]

Status: HEALTHY
 Total size:    35792 B
 Total dirs:    0
 Total files:   1
 Total blocks (validated):      1 (avg. block size 35792 B)
 Minimally replicated blocks:   1 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       0 (0.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
 Average block replication:     3.0
 Corrupt blocks:                0
 Missing replicas:              0 (0.0 %)
 Number of data-nodes:          8
 Number of racks:               1
```

虽然fsck会给出HDFS中每个文件的报告，但获知DataNode的情况却需要用dfsadmin命令。你可以使用dfsadmin命令中的-report选项：

```
bin/hadoop dfsadmin -report
Total raw bytes: 535472824320 (498.7 GB)
Remaining raw bytes: 33927731366 (31.6 GB)
Used raw bytes: 379948188541 (353.85 GB)
% used: 70.96%

Total effective bytes: 0 (0 KB)
Effective replication multiplier: Infinity
-------------------------------------------------
Datanodes available: 8

Name: 123.45.67.89:50010
State          : In Service
Total raw bytes: 76669841408 (71.4 GB)
Remaining raw bytes: 2184594843(2.03 GB)
Used raw bytes: 56598956650 (52.71 GB)
% used: 73.82%
Last contact: Sun Jun 21 16:13:32 PDT 2009
Name: 123.45.67.90:50010
State          : In Service
```

```
Total raw bytes: 76669841408 (71.4 GB)
Remaining raw bytes: 6356175381(5.92 GB)
Used raw bytes: 54220537856 (50.5 GB)
% used: 70.72%
Last contact: Sun Jun 21 16:13:33 PDT 2009

Name: 123.45.67.91:50010
State          : In Service
Total raw bytes: 76669841408 (71.4 GB)
Remaining raw bytes: 6106387206(5.69 GB)
Used raw bytes: 52412190091 (48.81 GB)
% used: 68.36%
Last contact: Sun Jun 21 16:13:33 PDT 2009

...
```

若要看NameNode的当前活动状态，可以在dfsadmin中使用-metasave选项：

```
bin/hadoop dfsadmin -metasave filename
```

这会将一部分NameNode的元数据保存到日志目录下filename文件中。在这些元数据中，你会发现待复制的块、正在复制的块以及待删除块的列表。每个块也会有一个复制列表，指向它被复制到的DataNode。最后，这个文件还会给出每个DataNode的统计摘要。

8.3 权限设置

HDFS采用类似于POSIX语义的基本文件权限管理系统。每个文件有9种权限设置：与每个文件关联的所有者组和其他用户分别有读(r)、写(w)和执行(x)权限。不是所有的权限设置都有意义。在HDFS中我们不能执行文件，所以不设置文件的x权限。

目录的权限设置也严格遵循POSIX语义。权限r允许列出目录。权限w允许创建或删除文件或目录。权限x允许访问子目录。

当前版本的HDFS没有太多安全上的考虑。使用HDFS的权限管理系统只是为了防止在共享Hadoop集群的受信任用户之间发生数据的意外误用和覆盖。HDFS不会对用户进行身份验证，并相信用户就是主机操作系统所声称的身份。Hadoop的用户名就是登录名，等同于whoami所示的用户。组列表与bash -c groups所列出的一致。启动name node的用户名是一个例外。该用户名有一个特别的Hadoop用户名superuser。这个超级用户可以执行任何文件操作，不受权限设置的约束。此外，管理员可以在一个超级用户组中通过配置参数dfs.permissions.supergroup来指定成员。所有超级用户组的成员也都是超级用户。

更改权限设置和所有权可以使用bin/hadoop fs -chmod、-chown以及-chgrp。使用方式与Unix中对应的命令类似。

8.4 配额管理

默认情况下，HDFS 不设任何配额来限制一个目录中可以放多少内容。你可以在指定目录上启用和指定所谓的名字配额，限制放在该目录下的文件名和目录名的个数。其主要用途是防止用户生成过多的小文件，令NameNode负担过重。以下命令用于设置和清除名字配额：

```
bin/hadoop dfsadmin -setQuota <N> directory [...directory]
bin/hadoop dfsadmin -clrQuota directory [...directory]
```

HDFS从0.19版开始，还支持对每个目录做空间配额。它有助于管理一个用户或应用程序可占用的存储量。

```
bin/hadoop dfsadmin -setSpaceQuota <N> directory [...directory]
bin/hadoop dfsadmin -clrSpaceQuota directory [...directory]
```

在SetSpaceQuota命令中，代表每个目录配额的参数以字节为单位。这个参数还可以用后缀来表示单位。例如，20 g将代表20 GB，而5 t则代表5 TB。所有的副本都计入配额。

若要获取目录的配额，以及它使用的名字个数和字节数，可使用带有-q选项的HDFS shell命令count。

```
bin/hadoop fs -count -q directory [...directory]
```

8.5　启用回收站

除了文件权限之外，还有一个保护机制可以防止在HDFS上意外删除文件，这就是回收站。默认情况下该功能被禁用。当它启用后，用于删除文件的命令行程序不会立即删除文件。相反，它们暂时把文件移动到用户工作目录下的.Trash/文件夹下。在用户设置的时间延迟到来之前，它都不会被永久删除。只要文件仍在.Trash/文件夹下，你就可以通过将它移回到原来位置的方法来恢复它。

若要启用回收站功能并设置清空回收站的时间延迟，可以通过设置core-site.xml的fs.trash.interval属性（以分钟为单位）。例如，如果你希望用户有24个小时（1440分钟）的时间来还原已删除的文件，应该在core-site.xml中设置

```
<property>
  <name>fs.trash.interval</name>
  <value>1440</value>
</property>
```

如将该值设置为0，则将禁用回收站功能。

8.6　删减 DataNode

有时你会从HDFS集群中删除DataNode。例如离线升级或者维护一台机器。从Hadoop中删除节点是非常简单的。虽然不推荐这样做，但你的确可以杀死节点或将之从集群中断开。HDFS的设计非常有弹性。让一两个DataNode离线不会影响操作的正常运行。NameNode会检测到节点的死亡，并开始复制那些低于约定副本数的数据块。为了让操作更为顺畅和安全，特别当删减大批DataNode时，你应该使用Hadoop的退役（decommissioning）功能。该功能确保所有块在剩余的活动节点上仍达到所需的副本数。要使用此功能，你必须在NameNode的本地文件系统上生成一个排除文件（最初是空的），并让参数dfs.hosts.exclude在NameNode的启动过程中指向该文件。当你想删减DataNode时，把它们列在排除文件中，每行列一个节点。还必须用完整的主机名、IP或IP:port的格式来指定节点。执行

```
bin/hadoop dfsadmin -refreshNodes
```

来强制Name Node重新读取排除文件，并开始退役过程。当此过程结束后，NameNode的日志文件中会出现像“Decommission complete for node 172.16.1.55:50010”这样的消息，此时你就可以将节点从集群中移出。

如果你在启动HDFS时没有让dfs.hosts.exclude指向排除文件，退役DataNode的正确方法是：关闭NameNode。设置dfs.hosts.exclude指向一个空的排除文件。重新启动NameNode。在NameNode成功重启后，就可以按照上面的步骤操作。请注意如果你在重启NameNode之前在排除文件中列出了需要删减的DataNode，那么NameNode就会混淆，而在日志中引发“ProcessReport from unregistered node: node055:50010”这样的消息。NameNode会认为它接触到的是系统之外的DataNode，而不是即将去除的节点。

如果退役的机器在后来的某个时刻还会重新加入集群，你应该在外部文件中删除它们，并重新执行bin/hadoop dfsadmin -refreshNodes来更新NameNode。当机器已准备好重新加入集群时，你可以按照下一节中介绍的步骤来添加它们。

8.7 增加 DataNode

除了让离线维护的机器重新上线，你可能还会在Hadoop集群中增加DataNode，以便有更多的作业来处理更多的数据。采用与集群中所有DataNode都一样的方式在新节点上安装Hadoop并设置配置文件。手动启动DataNode的守护进程（bin/hadoop datanode）。它会自动联系NameNode并加入集群。你还应把新节点添加到主服务器的conf/slaves文件中。脚本命令会识别到这个新节点。

当你添加一个新的DataNode时，它最初会是空的，然而早先的DataNode已经存了一些内容。这时，文件系统被认为是不平衡的。新的文件将有可能进入新节点，但其副本块仍会进入先前的节点。我们应该主动地启动HDFS平衡器来获得最优性能。平衡器的运行脚本为：

```
bin/start-balancer.sh
```

该脚本将在后台运行，直到集群达到平衡为止。管理员还可以提前终止它，即运行：

```
bin/stop-balancer.sh
```

当所有DataNode的利用率处于平均利用率加减一个阈值的范围内时，集群就被认为是平衡的。当启动一个平衡器脚本时，可以指定一个不同的阈值。默认情况下的阈值为10%，当启动平衡器脚本时，也可以指定一个与此不同的阈值。例如，要设置阈值为5%以便让集群达到更优的均匀分布，需这样启动平衡器

```
bin/start-balancer.sh -threshold 5
```

因为均衡操作会占用网络资源，我们建议在晚上或者周末做，此时集群可能不会太忙。或者，你可以设置dfs.balance.bandwidthPerSec参数，以限制用于做均衡的带宽。

8.8 管理 NameNode 和 SNN

NameNode是HDFS体系结构中最为重要的组件之一。它保存文件系统的元数据，并在RAM中缓存集群的块映射来获得不错的性能。除非集群非常小，不然，你应当为NameNode单独

指定一个节点，不在上面运行任何DataNode、JobTracker或TaskTracker服务。NameNode节点应该是集群中最强大的机器。给它尽可能多的RAM。虽然DataNode使用JBOD磁盘设备会获得更高的性能，但你绝对应该把RAID分给NameNode用以提高可靠性，以防单个驱动器出现故障。

减轻NameNode负担的一种方法是增加数据块的大小，以降低文件系统中元数据的数量。将块大小增加一倍，元数据几乎减少一半。不过对于那些不是很大的文件，它们的访问并行度也因此降低了。理想的块大小会由具体场景来决定。在配置参数dfs.block.size中可以设置块的大小。例如，若要让块的大小提高一倍，从默认值64 MB增加到128 MB，可以将dfs.block.size设置为134217728。

默认情况下，NameNode和SNN（Secondary NameNode）[①] 运行在同一台计算机上。对于中等规模的集群（10个或者更多节点），应该为SNN分配专用的机器，规格应相当于NameNode。但是，在讨论如何安装一个独立的服务器作为SNN之前，我会先解释SNN会做什么，以及不会做什么，继而讨论NameNode的一些内在机制。

由于取了一个不妥的名字，SNN有时会被混淆为NameNode的失效备份。它绝对不是。SNN仅仅是定期清理NameNode上文件系统的状态信息，使之紧凑，进而帮助 NameNode变得更有效率。NameNode使用FsImage和EditLog这两个文件来管理文件系统的状态信息。文件FsImage是文件系统在一些检查点上的快照，而EditLog记录在此检查点之后文件系统的每个增量修改（delta）。通过这两个文件可以完全确定文件系统的当前状态。当初始化NameNode时，它将合并这两个文件来创建新的快照。在NameNode初始化结束时，FsImage将包含新的快照，而EditLog将为空。之后，任何导致HDFS状态改变的操作都被追加到EditLog，而FsImage将保持不变。NameNode关闭并重新启动时，将再次进行合并，并产生新的快照。请注意这两个文件仅为在NameNode不运行时（有计划的关闭或系统故障）文件系统保留的状态信息。NameNode将文件系统的状态信息常驻在内存中，以快速地响应对文件系统的相关查询。

在繁忙的集群中，EditLog文件会变得非常大，并且NameNode在下次重新启动时将花很长的时间才能把EditLog合并到FsImage中。而且，集群繁忙时，NameNode两次重启之间的时间也会很长，你可能需要更频繁地做快照以便于存档。此时SNN便有了用武之地。它合并FsImage和EditLog到一个新的快照中，并让NameNode专注于活动的事务。因此，把SNN视为一个检查点服务器更为合适。合并FsImage和EditLog是很耗内存的，需要SNN与正常NameNode操作相当的内存容量。最好把SNN放在一个与主NameNode同等级别的独立的服务器上。

若要配置HDFS让其使用单独的服务器作为SNN，首先需在conf/masters上列出该服务器的主机名或IP地址。不过这个文件名同样令人困惑。Hadoop上的主节点（NameNode和JobTracker）为你在其上运行bin/start-dfs.sh和bin/start-mapred.sh的机器。在conf/masters中所列出的却是SNN，与主节点没有什么关系。

你还需在SNN上修改conf/hdfs-site.xml文件，让属性dfs.http.address指向NameNode主机

---

① 在本书撰写时，Secondary NameNode被提出将在Hadoop版本0.21中被弃用，该版本将在本书印刷时被发行。Secondary NameNode将会被一个更鲁棒的热备设计方案所取代。你应该检查自己所使用的Hadoop版本的在线文档，来确定它是否仍在使用Secondary NameNode。这个修改的专用补丁详见https://issues.apache.org/jira/browse/HADOOP-4539。

地址的端口50070，如

```
<property>
    <name>dfs.http.address</name>
    <value>namenode.hadoop-host.com:50070</value>
</property>
```

你必须设置此属性，因为SNN从NameNode中获取FsImage和EditLog，是通过向下面的网址发送HTTP的Get请求得到的：

- *FsImage*——http://*namenode.hadoop-host.com*:50070/getimage?getimage=1
- *EditLog*——http://*namenode.hadoop-host.com*:50070/getimage?getedit=1

SNN也把合并的元数据通过相同的地址和端口更新到NameNode上。

8.9 恢复失效的 NameNode

故障总会发生，所以Hadoop被设计得非常有弹性。不过NameNode仍然是薄弱环节。如果NameNode当机，HDFS就会失效。一种常见的设计是通过重用SNN来建立后备的NameNode服务器[①]。毕竟，SNN具有与NameNode相似的硬件规格，而且在Hadoop上安装的目录配置应该也是一样的。如果我们做一些额外的工作，维护SNN为NameNode的功能镜像，我们可以在NameNode出现故障的情况下快速启动这个备份机。启动备份节点作为新的NameNode需要人工的干预和时间，但至少我们不会丢失任何数据。

NameNode在dfs.name.dir目录下保存了所有的文件系统元数据，包括FsImage和EditLog文件。请注意SNN服务器根本不会使用该目录。它把系统的元数据下载到fs.checkpoint.dir目录下，并在那里进一步合并FsImage和EditLog。因为SNN上的dfs.name.dir目录未被使用，我们可以通过网络文件系统（NFS）把它暴露给NameNode。我们让NameNode每次写入本地元数据目录时，也把数据写入这个挂载目录。HDFS支持在多个目录中写入元数据。你需要用一个逗号分隔的列表来设定NameNode上的dfs.name.dir，就像下面这样。

```
<property>
    <name>dfs.name.dir</name>
    <value>/home/hadoop/dfs/name,/mnt/hadoop-backup</value>
    <final>true</final>
</property>
```

这里假定NameNode和SNN上的本地dfs.name.dir目录都位于/home/hadoop/dfs/name，而且SNN上的目录被挂载到NameNode上的/mnt/hadoop-backup。当HDFS在dfs.name.dir中看到以逗号分隔的列表时，它就会将其元数据写入到列表中的每个目录。

采用这种安装方式，当NameNode当机时，NameNode和备份节点（SNN）上的本地dfs.name.dir目录将拥有相同的内容。但是，要让备份节点作为NameNode，你必须把它的IP地址改为原始NameNode的IP地址。（遗憾的是，仅更改主机名是不够的，因为DataNode缓存了DNS entry。）你还得通过执行bin/start-dfs.sh让备份节点运行起来，让它成为NameNode。

① 不过这种常见的设计也促成了将SNN 视为备份节点的误解。你可以在NameNode和SNN之外的机器上设置备份节点，但该机器应该在绝大部分时间处于空闲状态。

若要更安全，这个新的NameNode也需要在启动之前安装备份节点。否则如果新的NameNode再失效会给你带来麻烦。如果你没有现成的节点作备份机，你至少应创建一个挂载NFS的目录。这样文件系统的状态信息可以放在多个位置。

因为HDFS会将元数据写入到dfs.name.dir所列出的所有目录中，如果NameNode中有多个硬盘，你可以将元数据的副本保存在属于不同硬盘的目录中。这样如果一个驱动器出现故障，就可以在没有坏盘的节点上重启NameNode。这比切换到备份节点更容易，因为切换会涉及IP地址改变及安装新的备份节点等。

回顾一下，SNN在目录fs.checkpoint.dir中创建文件系统元数据的快照。因为它只会周期性做检查点（默认每小时一次），元数据对于故障恢复而言不够新。但是，定期将这个目录存档到远程存储器上仍然是一个好主意。在发生灾难的情况下，从陈旧的数据中恢复起码比没有数据可用要强。NameNode及其备份节点同时失效的情况是存在的（比如一次电涌影响到了两台计算机）。另一种不幸的情况是文件系统的元数据早已损坏（比如因为人为错误或软件错误），而导致所有副本都毫无用处。从检查点的映像中恢复的详细解释可见http://issues.apache.org/jira/browse/HADOOP-2585。

本书撰写时HDFS的备份和恢复机制仍在积极地完善。你可以检查HDFS的在线文档来了解最新的消息。也有一些特殊的Linux软件，如DRBD[①]，可用在Hadoop集群上来实现很高的可用性。你可以在http://www.cloudera.com/blog/2009/07/22/hadoop-ha-configuration/看到一个示例。

8.10　感知网络布局和机架的设计

随着Hadoop集群变大，节点将分布在多个机架中，而集群的网络拓扑结构会开始影响可靠性和性能。你也许会希望集群能够在整个机架的故障中得以幸免。如上一节所述，你应在一个NameNode机架之外放置一个备份服务器。这样任何机架的失效都不会导致文件系统中元数据的所有副本被销毁。

当超过一个机架时，块副本和任务的位置就变得更加复杂。块的副本应放在彼此独立的机架上，以减少潜在的数据丢失。标准的副本数为3，在写入一个块时的默认放置策略是：如果执行写操作的客户端是Hadoop集群的一部分，第一个副本应放在客户端所在的DataNode中。否则随机放置在集群中。第二个副本随机放置在与第一个副本不同的机架中。第三个副本放在与第二个副本相同机架上的不同节点上。如果副本数大于3，后续的副本在节点上随机放置。撰写本文时，这种块放置策略被并入NameNode中。计划在版本0.21中将实现可插拔的策略。[②]

除了块的放置，任务的放置也是机架感知的。任务被放置的节点通常拥有该任务所处理块的副本。当没有这种节点可用来承担这项新任务时，任务就随机分配给机架上的一个节点，只要该机架的某个节点上可以获得这个副本。就是说，当数据局部性无法在节点级别达成，Hadoop就尝试在机架级别实现。如果还不成功，任务就随机分配给剩余节点中的一个。

---

① 详见http://www.drbd.org。

② 关于这个改变的描述见http://issues.apache.org/jira/browse/HDFS-385。

在这一点上，你可能想知道Hadoop是如何知道节点所处机架的。这需要你来告诉它。它假定你的Hadoop集群为层次式网络拓扑，结构类似于图8-1。每个节点都有类似于文件路径的机架名。例如，图8-1中的节点H1、H2及H3全都有机架名/D1/R1。图8-1给出了一个例子，如果有多个数据中心（D1和D2），每个有多个机架（R1至R4）。在大多数情况下你需要对多个机架做统一的管理。你的机架名将采用平面名称空间，如/R1和/R2。

为了帮助 Hadoop知道每个节点的位置，你必须提供一个可执行脚本，能够把IP地址映射到机架名。这个网络拓扑的脚本必须驻留在主节点上，它的位置在core-site.xml的topology.script.file.name属性中指定。Hadoop调用脚本时会使用一组IP地址作为彼此独立的参数。该脚本会以相同的顺序（通过STDOUT）打印出每个IP地址所对应的机架名，中间以空格分隔。属性topology.script.number.args控制在任一时刻Hadoop所请求的IP地址的最大数目。简便的办法是将这个值设为1，以简化你的脚本。下面是一个网络拓扑脚本的示例。

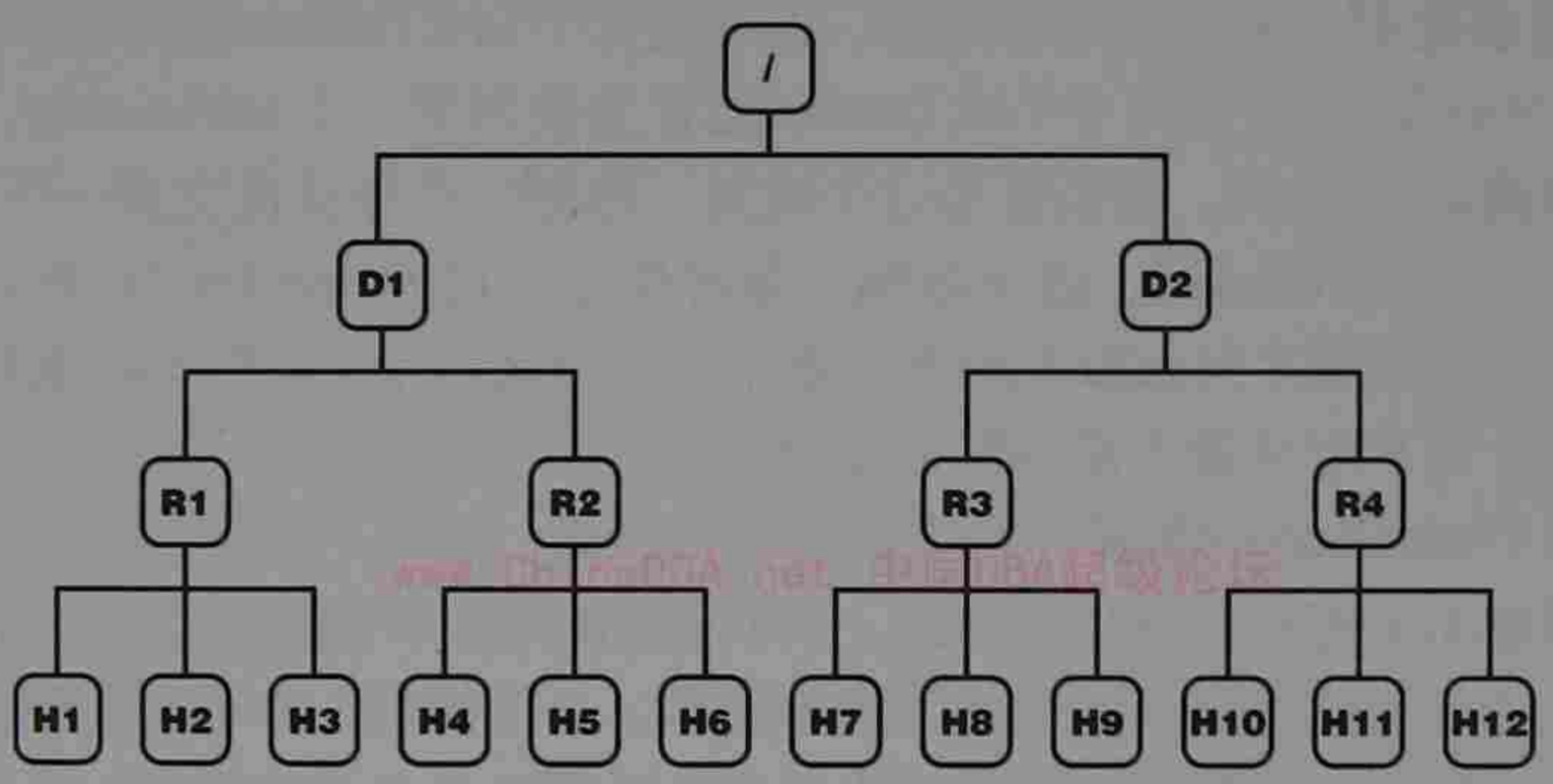

图8-1 一个采用层次化网络拓扑的集群。该集群跨越多个数据中心（D1和D2）。每个数据中心拥有多个机架(R)，每个机架有多台计算机

```
#!/bin/bash

ipaddr=$1
segments='echo $ipaddr | cut --delimiter=. --fields=4'
if [ "$segments" -lt 128 ]; then
   echo /rack-1
else
   echo /rack-2
fi
```

这个bash脚本取得IPv4地址，并提取4个八位字节（假定为十进制点符号）中的最后一个。如果最后一个八位字节少于128，节点被认为在机架1中，否则在机架2中。在较复杂的集群拓扑结构中使用查找表会更合适。另外，如果没有网络拓扑的脚本，Hadoop将假定采用平坦拓扑，所有节点都被分配到/default-rack中。

8.11 多用户作业的调度

当Hadoop集群上有越来越多来自于多个用户的作业时，你需要采取一些控制措施来避免冲突。Hadoop默认采用FIFO调度器，只要有一个作业被送到Hadoop上执行，JobTracker就会根据作

业处理的需求为它分配尽可能多的TaskTracker。当事务不繁忙而你有大量闲置的处理能力时，这种调度工作得很好。但是，一些大的Hadoop作业会很容易长时间占用集群，而让较小的作业等待。如果在Hadoop集群中为较小的作业建立一个类似快速收银通道的东西，不是很棒吗？

8.11.1　多个 JobTracker

回到Hadoop版本0.19以前的时代，你不得不在物理上设置多个MapReduce集群，把基本的CPU资源在作业间分配。但是为了保持高效合理的存储利用率，这仍然是一个单一的HDFS集群。就是说，你的Hadoop集群有Z个从节点。你必须让一个NameNode把所有Z个节点作为DataNode管理起来。所有Z个节点也都将为TaskTracker。到目前为止，所有这些TaskTracker将指向相同或唯一的JobTracker。

多集群安装的一个诀窍是用多个JobTracker，每个JobTracker（互斥地）控制一个TaskTracker子集。例如，若要创建两个MapReduce集群，必须让X个TaskTracker指向一个JobTracker（利用mapred.job.tracker属性），让Y个TaskTracker配置为使用第二个JobTracker。两个MapReduce集群之间的从节点满足X+Y=Z。若要使用这个设置，你把一些作业提交到一个JobTracker，而把其他的作业分给另一个JobTracker。这就限制了每种作业可用的TaskTracker数目。作业的类型并不一定会左右作业分配到哪个MapReduce池。更常见的是将每个MapReduce池划给一个用户组来使用。这将强制一个用户组只能占用有限的资源。

物理上设置多个MapReduce集群这种方式有很多缺点。它并不方便，因为需要人们记住使用的是哪个池。数据也不太可能总在任务本地。（有时所有的副本可能都处于池外的DataNode上。）而且，这种设置无法灵活地适应不断变化的资源需求。可喜的是，从0.19版开始，Hadoop在调度器上采用了插件式的架构，并提供了两个新的调度器以解决作业冲突问题。一个是Facebook开发的公平调度器（Fair Scheduler）；另一个是雅虎开发的容量调度器（Capacity Scheduler）。

8.11.2　公平调度器

公平调度器引入了池的概念。作业都标记为属于特定的池，并且每个池被配置为必须拥有一定数量的map slot和reduce slot。当task slot被释放时，公平调度器会首先满足这些最低限度的保障。在保障被满足后，slot再在作业之间“公平共享”，使得每个作业获取大致相同的计算资源。你也可以设置作业的优先级，让更高优先级的作业获得更多资源（一些作业实际上比其他作业更有特权）。

公平调度器可以从Hadoop安装目录contrib/fairscheduler下的hadoop-*-fairscheduler.jar文件中获得。安装时，可将此jar文件直接移动到Hadoop的lib/目录下。或者，可以修改脚本conf/hadoop-env.sh 中的HADOOP_CLASSPATH来包含此jar文件。

你需要在hadoop-site.xml中设置几个属性来完全启用与配置公平调度器。首先，将mapred.jobtracker.taskScheduler设置为org.apache.hadoop.mapred.FairScheduler，让Hadoop使用公平调度器，而非默认调度器。然后，对公平调度器的几个属性进行配置。其中最重要的属性为mapred.fairscheduler.allocation.file，它指向用于定义不同池的文件。该文件通常命名为pools.xml，指定了每个池的名称和容量。mapred.fairscheduler.poolnameproperty定义了jobconf属性，调度器会通过它来确定作业使用哪个池。一种实用的配置方式是将它设置

为一个新属性，如pool.name，并将其默认值设为${user.name}。公平调度器自动赋予每个用户其独有的池。默认情况下这个特定的pool.name 将分配每个作业到其宿主的池。你可以在作业的jobconf中修改pool.name的属性来分配这个作业到一个不同的池中[①]。最后，如果mapred.fairscheduler.assignmultiple属性被设置为true，会允许调度器在每个心跳到来时指定map任务和reduce任务，从而提高了构建速度和吞吐量。简而言之，mapred-site.xml中的属性设置如下所示：

```
<property>
  <name>mapred.jobtracker.taskScheduler</name>
  <value>org.apache.hadoop.mapred.FairScheduler</value>
</property>
<property>
  <name>mapred.fairscheduler.allocation.file</name>
  <value>HADOOP_CONF_DIR/pools.xml</value>
</property>
<property>
  <name>mapred.fairscheduler.assignmultiple</name>
  <value>true</value>
</property>
<property>
  <name>mapred.fairscheduler.poolnameproperty</name>
  <value>pool.name</value>
</property>
<property>
  <name>pool.name</name>
  <value>${user.name}</value>
</property>
```

调度器中对池的定义放在配置文件pools.xml中。它给出每个池的名字及其容量限制。约束可以包括map slot或reduce slot的最小数目。还可以包含正在运行作业的最大数目。此外，你可以设置每个用户运行作业的最大数目，并为特定的用户重新定义这个最大值。pools.xml的示例如下所示：

```
<?xml version="1.0"?>
<allocations>
  <pool name="ads">
    <minMaps>2</minMaps>
    <minReduces>2</minReduces>
  </pool>
  <pool name="hive">
    <minMaps>2</minMaps>
    <minReduces>2</minReduces>
    <maxRunningJobs>2</maxRunningJobs>
  </pool>
  <user name="chuck">
    <maxRunningJobs>6</maxRunningJobs>
  </user>
  <userMaxJobsDefault>3</userMaxJobsDefault>
</allocations>
```

① 是的，你可以在另一个用户的池中运行你的作业，但这可不太礼貌。这样做的主要用途是将特殊的作业分配在特定的池中。例如，你可能希望所有cron作业被归入一个单独的池中，而不是让它们在每个用户的池中运行。

这个pools.xml定义了两个特殊的池——“ads”和“hive”。每个都保证有至少两个map slot和两个reduce slot。“hive”池被限定一次最多运行两个作业。若要使用这些池，需设置作业配置中的pool.name属性为“ads”或“hive”。这个pools.xml还限制一个用户最多有3个同时运行的作业，但用户“chuck”最多可以运行6个。

请注意pools.xml文件每15秒被重新读取一次。你可以在运行时修改此文件，并动态地重新分配容量。任何在此文件中未定义的池都没有容量保证，也没有同时运行作业数量的限制。

当你让Hadoop集群运行公平调度器时，有一个Web用户界面可用于管理这个调度器。网页位于http://<jobtracker url>/scheduler。除了让你知道作业被如何调度之外，它还允许更改作业所属的池，以及作业的优先级。图8-2显示了该页面的示例截图。

localhost Job Scheduler Administration

Pools

Pool	Running Jobs	Min Maps	Min Reduces	Running Maps	Running Reduces
ads	0	2	2	0	0
chuck	1	0	0	1	1
hive	0	2	2	0	0
default	0	0	0	0	0

Running Jobs

Submitted	JobID	User	Name	Pool	Priority	Maps			Reduces		
						Finished	Running	Fair Share	Finished	Running	Fair Share
Aug 11, 04:08	job_200908110346_0002	chuck	streamjob6351400141424754280.jar	chuck	NORMAL	3 / 4	1	2.0	0 / 1	1	2.0

Scheduling Mode

The scheduler is currently using **Fair Sharing mode** Switch to FIFO mode

图8-2　监视Hadoop公平调度器的Web用户界面。上面的表显示了所有可用的池和每个池的使用率。下面的表有一个名为Pool的列，可用于监视或更改每个作业的池

容量调度器与公平调度器有相似的目标。但是，容量调度器作用于队列，而不是池。有兴趣的读者可以通过在线文档了解到有关容量调度器的更多内容：http://hadoop.apache.org/common/docs /r0.20.0 /capacity_scheduler.html。

8.12　小结

分布式集群的管理非常复杂，而Hadoop并非异类。在本章，我们介绍了很多常见的管理任务。如果你的设置复杂并且问题繁多，Hadoop的邮件列表[①]就是一个有用的资源。许多经验丰富的Hadoop管理员在其中非常活跃，有可能他们中的一位就曾遇到过你的问题。另外，如果你主要是希望拥有一个基本的Hadoop集群，而免去管理的麻烦，你可以考虑使用Cloudera的发行版[②]，或者采用一个Hadoop云服务，我们会在下一章中加以讨论。

① http://hadoop.apache.org/common/mailing_lists.html。

② http://www.cloudera.com/distribution。