

DistCp

目录

1 概述.....	2
2 使用方法.....	2
2.1 基本使用方法.....	2
2.2 选项.....	3
3 附录.....	5
3.1 Map数目.....	5
3.2 不同HDFS版本间的拷贝.....	6
3.3 Map/Reduce和副效应.....	6

1. 概述

DistCp（分布式拷贝）是用于大规模集群内部和集群之间拷贝的工具。它使用Map/Reduce实现文件分发，错误处理和恢复，以及报告生成。它把文件和目录的列表作为map任务的输入，每个任务会完成源列表中部分文件的拷贝。由于使用了Map/Reduce方法，这个工具在语义和执行上都会有特殊的地方。这篇文档会为常用DistCp操作提供指南并阐述它的工作模型。

2. 使用方法

2.1. 基本使用方法

DistCp最常用在集群之间的拷贝：

```
bash$ hadoop distcp hdfs://nn1:8020/foo/bar \
                    hdfs://nn2:8020/bar/foo
```

这条命令会把nn1集群的/foo/bar目录下的所有文件或目录名展开并存储到一个临时文件中，这些文件内容的拷贝工作被分配给多个map任务，然后每个TaskTracker分别执行从nn1到nn2的拷贝操作。注意DistCp使用绝对路径进行操作。

命令行中可以指定多个源目录：

```
bash$ hadoop distcp hdfs://nn1:8020/foo/a \
                    hdfs://nn1:8020/foo/b \
                    hdfs://nn2:8020/bar/foo
```

或者使用-f选项，从文件里获得多个源：

```
bash$ hadoop distcp -f hdfs://nn1:8020/srclist \
                    hdfs://nn2:8020/bar/foo
```

其中srclist 的内容是

```
hdfs://nn1:8020/foo/a
hdfs://nn1:8020/foo/b
```

当从多个源拷贝时，如果两个源冲突，DistCp会停止拷贝并提示出错信息，如果在目标位置发生冲突，会根据[选项设置](#)解决。默认情况会跳过已经存在的目标文件（比如不用源文件做替换操作）。每次操作结束时 都会报告跳过的文件数目，但是如果某些

拷贝操作失败了，但在之后的尝试成功了，那么报告的信息可能不够精确（请参考[附录](#)）。

每个TaskTracker必须都能够与源端和目的端文件系统进行访问和交互。对于HDFS来说，源和目的端要运行相同版本的协议或者使用向下兼容的协议。（请参考[不同版本间的拷贝](#)）。

拷贝完成后，建议生成源端和目的端文件的列表，并交叉检查，来确认拷贝真正成功。因为DistCp使用Map/Reduce和文件系统API进行操作，所以这三者或它们之间有任何问题都会影响拷贝操作。一些Distcp命令的成功执行可以通过再次执行带-update参数的该命令来完成，但用户在如此操作之前应该对该命令的语法很熟悉。

值得注意的是，当另一个客户端同时在向源文件写入时，拷贝很有可能会失败。尝试覆盖HDFS上正在被写入的文件的操作也会失败。如果一个源文件在拷贝之前被移动或删除了，拷贝失败同时输出异常 FileNotFoundException。

2.2. 选项

2.2.1. 选项索引

标识	描述	备注
-p[rbugp]	Preserve r: replication number b: block size u: user g: group p: permission	修改次数不会被保留。并且当指定 -update 时，更新的状态不会被同步，除非文件大小不同（比如文件被重新创建）。
-i	忽略失败	就像在 附录 中提到的，这个选项会比默认情况提供关于拷贝的更精确的统计，同时它还将保留失败拷贝操作的日志，这些日志信息可以用于调试。最后，如果一个map失败了，但并没完成所有分块任务的尝试，这不会导致整个作业的失败。
-log <logdir>	记录日志到 <logdir>	DistCp为每个文件的每次尝试拷贝操作都记录日志，并把日志作为map的输出。如果一个map失败了，当重新执行时这个日志不

		会被保留。
<code>-m <num_maps></code>	同时拷贝的最大数目	指定了拷贝数据时map的数目。请注意并不是map数越多吞吐量越大。
<code>-overwrite</code>	覆盖目标	如果一个map失败并且没有使用 <code>-i</code> 选项，不仅仅那些拷贝失败的文件，这个分块任务中的所有文件都会被重新拷贝。就像 下面 提到的，它会改变生成目标路径的语义，所以 用户要小心使用这个选项。
<code>-update</code>	如果源和目标的大小不一样则进行覆盖	像之前提到的，这不是"同步"操作。执行覆盖的唯一标准是源文件和目标文件大小是否相同；如果不同，则源文件替换目标文件。像 下面 提到的，它也改变生成目标路径的语义， 用户使用要小心。
<code>-f <urilist_uri></code>	使用<urilist_uri> 作为源文件列表	这等价于把所有文件名列在命令行中。 <code>urilist_uri</code> 列表应该是完整合法的URI。

2.2.2. 更新和覆盖

这里给出一些 `-update`和 `-overwrite`的例子。考虑一个从 `/foo/a` 和 `/foo/b` 到 `/bar/foo`的拷贝，源路径包括：

```
hdfs://nn1:8020/foo/a
hdfs://nn1:8020/foo/a/aa
hdfs://nn1:8020/foo/a/ab
hdfs://nn1:8020/foo/b
hdfs://nn1:8020/foo/b/ba
hdfs://nn1:8020/foo/b/ab
```

如果没设置`-update`或 `-overwrite`选项， 那么两个源都会映射到目标端的 `/bar/foo/ab`。如果设置了这两个选项，每个源目录的内容都会和目标目录的 内容做比较。DistCp碰到这类冲突的情况会终止操作并退出。

默认情况下，/bar/foo/a 和 /bar/foo/b 目录都会被创建，所以并不会冲突。

现在考虑一个使用-update合法的操作：

```
distcp -update hdfs://nn1:8020/foo/a \
               hdfs://nn1:8020/foo/b \
               hdfs://nn2:8020/bar
```

其中源路径/大小：

```
hdfs://nn1:8020/foo/a
hdfs://nn1:8020/foo/a/aa 32
hdfs://nn1:8020/foo/a/ab 32
hdfs://nn1:8020/foo/b
hdfs://nn1:8020/foo/b/ba 64
hdfs://nn1:8020/foo/b/bb 32
```

和目的路径/大小：

```
hdfs://nn2:8020/bar
hdfs://nn2:8020/bar/aa 32
hdfs://nn2:8020/bar/ba 32
hdfs://nn2:8020/bar/bb 64
```

会产生：

```
hdfs://nn2:8020/bar
hdfs://nn2:8020/bar/aa 32
hdfs://nn2:8020/bar/ab 32
hdfs://nn2:8020/bar/ba 64
hdfs://nn2:8020/bar/bb 32
```

只有nn2的aa文件没有被覆盖。如果指定了 -overwrite选项，所有文件都会被覆盖。

3. 附录

3.1. Map数目

DistCp会尝试着均分需要拷贝的内容，这样每个map拷贝差不多相等大小的内容。但因为文件是最小的拷贝粒度，所以配置增加同时拷贝（如map）的数目不一定会增加实

际同时拷贝的数目以及总吞吐量。

如果没使用-m选项，DistCp会尝试在调度工作时指定map的数目 为 $\min(\text{total_bytes} / \text{bytes.per.map}, 20 * \text{num_task_trackers})$ ，其中bytes.per.map默认是256MB。

建议对于长时间运行或定期运行的作业，根据源和目标集群大小、拷贝数量大小以及带宽调整map的数目。

3.2. 不同HDFS版本间的拷贝

对于不同Hadoop版本间的拷贝，用户应该使用HftpFileSystem。这是一个只读文件系统，所以DistCp必须运行在目标端集群上（更确切的说是在能够写入目标集群的TaskTracker上）。源的格式是 `hftp://<dfs.http.address>/<path>`（默认情况dfs.http.address是 `<namenode>:50070`）。

3.3. Map/Reduce和副效应

像前面提到的，map拷贝输入文件失败时，会带来一些副效应。

- 除非使用了-i，任务产生的日志会被新的尝试替换掉。
- 除非使用了-overwrite，文件被之前的map成功拷贝后当又一次执行拷贝时会被标记为 "被忽略"。
- 如果map失败了mapred.map.max.attempts次，剩下的map任务会被终止（除非使用了-i）。
- 如果mapred.speculative.execution被设置为 final和true，则拷贝的结果是未定义的。