Hadoop本地库

目录

1. 目的

鉴于性能问题以及某些Java类库的缺失，对于某些组件，Hadoop提供了自己的本地实现。 这些组件保存在Hadoop的一个独立的动态链接的库里。这个库在*nix平台上叫libhadoop.so. 本文主要介绍本地库的使用方法以及如何构建本地库。

2. 组件

Hadoop现在已经有以下 compression codecs本地组件:

- zlib
- gzip
- lzo

在以上组件中，lzo和gzip压缩编解码器必须使用hadoop本地库才能运行。

3. 使用方法

hadoop本地库的用法很简单:

- 看一下 支持的平台.
- 下载 预构建的32位i386架构的Linux本地hadoop库（可以在hadoop发行版的lib/native目录下找到）或者自己 构建 这些库。
- 确保你的平台已经安装了zlib-1.2以上版本或者lzo2.0以上版本的软件包或者两者均已安装（根据你的需要）。

bin/hadoop 脚本通过系统属性 -Djava.library.path=<path>来确认hadoop本地库是否包含在库路径里。

检查hadoop日志文件可以查看hadoop库是否正常，正常情况下会看到:

```
DEBUG util.NativeCodeLoader - Trying to load the custom-built native-hadoop library...
INFO util.NativeCodeLoader - Loaded the native-hadoop library
```

如果出错，会看到:

```
INFO util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

4. 支持的平台

Hadoop本地库只支持*nix平台，已经广泛使用在GNU/Linux平台上，但是不支持 Cygwin 和 Mac OS X。

已经测试过的GNU/Linux发行版本：

- RHEL4/Fedora
- Ubuntu
- Gentoo

在上述平台上，32/64位Hadoop本地库分别能和32/64位的jvm一起正常运行。

5. 构建Hadoop本地库

Hadoop本地库使用 ANSI C 编写，使用GNU autotools工具链 (autoconf, autoheader, automake, autoscan, libtool)构建。也就是说构建hadoop库的平台需要有标准C的编译器和GNU autotools工具链。请参看 支持的平台。

你的目标平台上可能会需要的软件包：

- C 编译器 (e.g. GNU C Compiler)
- GNU Autools 工具链: autoconf, automake, libtool
- zlib开发包 (stable version >= 1.2.0)
- lzo开发包 (stable version >= 2.0)

如果已经满足了上述先决条件，可以使用build.xml 文件，并把其中的 compile.native置为 true，这样就可以生成hadoop本地库：

```
$ ant -Dcompile.native=true <target>
```

因为不是所有用户都需要Hadoop本地库，所以默认情况下hadoop不生成该库。

你可以在下面的路径查看新生成的hadoop本地库：

```
$ build/native/<platform>/lib
```

其中<platform>是下列系统属性的组合 ${os.name}-${os.arch}-${sun.arch.data.model}；例如 Linux-i386-32。

5.1. 注意

- 在生成hadoop本地库的目标平台上必须 安装了zlib和lzo开发包；但是如果你只希望使用其中一个的话，在部署时，安装其中任何一个都是足够的。
- 在目标平台上生成以及部署hadoop本地库时，都需要根据32/64位jvm选取对应的32/64位zlib/lzo软件包。

6. 使用DistributedCache 加载本地库

用户可以通过 DistributedCache 加载本地共享库，并分发和建立库文件的符号链接。

这个例子描述了如何分发库文件并在从map/reduce任务中装载库文件。

1. 首先拷贝库文件到HDFS。
   ```
   bin/hadoop fs -copyFromLocal mylib.so.1 /libraries/mylib.so.1
   ```
2. 启动作业时包含以下代码:
   ```
   DistributedCache.createSymlink(conf);
   DistributedCache.addCacheFile("hdfs://host:port/libraries/mylib.so.1#mylib.so",
   conf);
   ```
3. map/reduce任务中包含以下代码:
   ```
   System.loadLibrary("mylib.so");
   ```