Native Libraries Guide

Table of contents

1 Overview

This guide describes the native hadoop library and includes a small discussion about native shared libraries.

Note: Depending on your environment, the term "native libraries" *could* refer to all *.so's you need to compile; and, the term "native compression" *could* refer to all *.so's you need to compile that are specifically related to compression. Currently, however, this document only addresses the native hadoop library (*libhadoop.so*).

2 Native Hadoop Library

Hadoop has native implementations of certain components for performance reasons and for non-availability of Java implementations. These components are available in a single, dynamically-linked native library called the native hadoop library. On the *nix platforms the library is named *libhadoop.so*.

2.1 Usage

It is fairly easy to use the native hadoop library:

1. Review the components.
2. Review the supported platforms.
3. Either download a hadoop release, which will include a pre-built version of the native hadoop library, or build your own version of the native hadoop library. Whether you download or build, the name for the library is the same: *libhadoop.so*
4. Install the compression codec development packages (>**zlib-1.2**, >**gzip-1.2**):
   - If you download the library, install one or more development packages - whichever compression codecs you want to use with your deployment.
   - If you build the library, it is **mandatory** to install both development packages.
5. Check the runtime log files.

2.2 Components

The native hadoop library includes two components, the zlib and gzip compression codecs:

- zlib
- gzip

The native hadoop library is imperative for gzip to work.

2.3 Supported Platforms

The native hadoop library is supported on *nix platforms only. The library does not to work with Cygwin or the Mac OS X platform.

The native hadoop library is mainly used on the GNU/Linus platform and has been tested on these distributions:

- RHEL4/Fedora
- Ubuntu
- Gentoo

On all the above distributions a 32/64 bit native hadoop library will work with a respective 32/64 bit jvm.

2.4 Download

The pre-built 32-bit i386-Linux native hadoop library is available as part of the hadoop distribution and is located in the `lib/native` directory. You can download the hadoop distribution from Hadoop Common Releases.

Be sure to install the zlib and/or gzip development packages - whichever compression codecs you want to use with your deployment.

2.5 Build

The native hadoop library is written in ANSI C and is built using the GNU autotools-chain (autoconf, autoheader, automake, autoscan, libtool). This means it should be straight-forward to build the library on any platform with a standards-compliant C compiler and the GNU autotools-chain (see the supported platforms).

The packages you need to install on the target platform are:

- C compiler (e.g. GNU C Compiler)
- GNU Autools Chain: autoconf, automake, libtool
- zlib-development package (stable version >= 1.2.0)

Once you installed the prerequisite packages use the standard hadoop `build.xml` file and pass along the `compile.native` flag (set to `true`) to build the native hadoop library:

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

You should see the newly-built library in:

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

where <`platform`> is a combination of the system-properties: `${os.name}-${os.arch}-${sun.arch.data.model}` (for example, Linux-i386-32).

Please note the following:

- It is **mandatory** to install both the zlib and gzip development packages on the target platform in order to build the native hadoop library; however, for deployment it is sufficient to install just one package if you wish to use only one codec.

- It is necessary to have the correct 32/64 libraries for zlib, depending on the 32/64 bit jvm for the target platform, in order to build and deploy the native hadoop library.

2.6 Runtime

The `bin/hadoop` script ensures that the native hadoop library is on the library path via the system property:
-Djava.library.path=<path>

During runtime, check the hadoop log files for your MapReduce tasks.

- If everything is all right, then:
  ```
  DEBUG util.NativeCodeLoader - Trying to load the custom-
  built native-hadoop library...
  INFO util.NativeCodeLoader - Loaded the native-hadoop
  library
  ```
- If something goes wrong, then:
  ```
  INFO util.NativeCodeLoader - Unable to load native-hadoop
  library for your platform... using builtin-java classes
  where applicable
  ```

3 Native Shared Libraries

You can load **any** native shared library using DistributedCache for *distributing* and *symlinking* the library files.

This example shows you how to distribute a shared library, `mylib.so`, and load it from a MapReduce task.

1. First copy the library to the HDFS:
   ```
   bin/hadoop fs -copyFromLocal mylib.so.1 /libraries/
   mylib.so.1
   ```
2. The job launching program should contain the following:
   ```
   DistributedCache.createSymlink(conf);
   DistributedCache.addCacheFile("hdfs://host:port/libraries/
   mylib.so.1#mylib.so", conf);
   ```
3. The MapReduce task can contain:
   ```
   System.loadLibrary("mylib.so");
   ```

Note: If you downloaded or built the native hadoop library, you don't need to use DistibutedCache to make the library available to your MapReduce tasks.