Creating 64-bit Hadoop native libraries

Compiling libhadoop.so.1.0.0 for 64-bit systems

This is the fix to the warning that looks lie:

```
...
OpenJDK 64-Bit Server VM warning: You have loaded library
/home/benji/opt/hadoop-2.7.2/lib/native/libhadoop.so.1.0.0
which might have disabled stack guard. The VM will try to fix the stack guard now.
It's highly recommended that you fix the library with 'execstack -c <libfile>', or link
it with '-z noexecstack'.
14/06/17 12:49:47 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
your platform... using builtin-java  classes where applicable
...
```

You get this warning constantly when you are running Hadoop on a 64-bit machine. The reason you see the warning is the native Hadoop library

```
$HADOOP_HOME/lib/native/libhadoop.so.1.0.0
```

was compiled 32-bit.

Anyway, it's just a warning and won't impact Hadoop's functionalities - but it does impact performance.

To eliminate this warning, we download the source code of Hadoop, recompile libhadoop.so.1.0.0 on 64 bit system, then replace the 32 bit one. This document shows you how.

1. Install necessary tools: maven, libssl-dev and cmake, etc.

If you are using Ubuntu, use apt-get:

apt-get install maven build-essential zlib1g-dev cmake pkg-config libssl-dev

If you are using RHEL (and hence also CentOS), use yum:

yum -y install lzo-devel zlib-devel gcc autoconf automake libtool openssl-devel

If you are using SUSE:

Download maven from http://maven.apache.org/download.cgi.

The following instructions are for maven-3.3.x. (Last checked, the latest version was 3.3.9)

1. Extract the distribution archive, i.e. apache-maven-3.3.x-bin.tar.gz to the directory you wish to install Maven. These instructions assume you chose /usr/local/apache-maven. The subdirectory apache-maven-3.3.x will be created from the archive.
2. In a command terminal, add the M2_HOME environment variable, e.g. export M2_HOME=/usr/local/apache-maven/apache-maven-3.3.x.
3. Add the M2 environment variable, e.g. export M2=$M2_HOME/bin.
4. Add M2 environment variable to your path, e.g. export PATH=$M2:$PATH.
5. Run mvn --version to verify that it is correctly installed.

Then use **YAST > Software Management** to install **lzo-devel, zlib-devel, gcc, autoconf, automake, libtool, openssl-devel**.

2. Install locally the latest version of protobuf and insert it in the PATH

If you are using Ubuntu or SUSE, you'll need to build `protoc` - the executable for protobufs. Download the latest version of protobuf from

https://developers.google.com/protocol-buffers/docs/downloads

Create a directory for building and unpack the protobuf tar.gz file there. For example, if we are downloading protobuf 3.0.0 (released July, 2016):

```
>> mkdir /tmp/protobuf
>> cd /tmp/protobuf
>> wget http:// protobuf.googlecode.com/files/protobuf-3.0.0.tar.gz
>> tar -xvzf ./protobuf-3.0.0.tar.gz
>> cd protobuf-3.0.0
```

The following is the shortest approach to installation. (If you want more information, read the README.txt files in `protobuf-3.0.0` and `protobuf-3.0.0/java`.)
Enter:

```
>> ./configure
 -- this creates a makefile and tells it to put the 'protoc' executable in /user/local/bin --
>> make
  -- runs the makefile - please wait, this takes a minute or two --
>> sudo make install
  -- ok, you are done building protobuf C++, now on to Java --
>> cd java
>> mvn install
  -- use maven for the java install.  This takes a few seconds, especially the tests --
>> mvn package
  -- wait, ... more tests --
>> sudo ldconfig
  -- helps the system find your new library by caching its location --
>> cd /tmp
  -- you're done: the libraries and executables are in /usr/lib64 and usr/local/bin so you
can delete the temporary build directory.
>> rm -rf protobuf
```

3. Download, compile the Hadoop sources.

This example uses hadoop-2.7.2

Create a temporary build location and put the hadoop source there.

```
>> mkdir /tmp/hadoop-build
>> cd /tmp/hadoop-build
>> wget /
http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.7.2/hadoop-2.7.2-src.tar.gz
>> tar -xvzf ./hadoop-2.7.2-src.tar.gz
>> cd hadoop-2.7.2-src
```

Compile Hadoop

```
>> export Platform=x64
>> cd /tmp/hadoop-build/hadoop-2.7.2-src
>> mvn clean install -DskipTests
>> cd hadoop-mapreduce-project
>> mvn package -Pdist,native -DskipTests=true -Dtar
>> cd /tmp/hadoop-build/hadoop-2.7.2-src
>> mvn package -Pdist,native -DskipTests=true -Dtar
```

This should build the **Hadoop** distribution and you should find it in

hadoop-dist/target/hadoop-2.7.2.tar.gz

Note: If you are running multi-node, you should now copy hadoop-2.7.2.tar.gz to other nodes, so you don't have to build it on each one of them. Then follow remaining steps. Remember to also install the appropriate version of Oracle JDK on each node.

4. Unpack our distribution and move new library into place

The previous steps created a new tar.gz file with the compiled 64-bit libraries - hadoop-dist/target/hadoop-2.7.2.tar.gz

If you have already installed a 32-bit Hadoop

You need only replace the native libs in your current installation - they are in $HADOOP_HOME/lib/native - with the new native libs.

Then, if applicable, remove from $HADOOP_HOME/etc/hadoop-env.sh:

-- export HADOOP_COMMON_LIB_NATIVE_DIR="~/hadoop/lib/"

-- export HADOOP_OPTS="$HADOOP_OPTS -Djava.library.path=~/hadoop/lib/"

If this is a new installation of Hadoop

Here, we assume $HADOOP_HOME is set to `/usr/local/hadoop.`

Enter:

```
>> sudo tar -xvf hadoop-dist/target/hadoop-2.7.2.tar.gz -C /usr/local/
 -- extract your newly built hadoop to the /usr/local directory --
>> sudo ln -s /usr/local/hadoop-2.7.2 /usr/local/hadoop
 -- create a softlink you can swap out versions easily without changing $HADOOP_HOME  --
>> sudo chown -R hduser:hadoop /usr/local/hadoop-2.7.2
 -- shown:  owner is a hadoop user hduser and group hadoop.  If you used the quick setup for a single-cluster, use your user name and group (for instance, benji:users).  --

You can now delete Hadoop sources.
```