Use NativeBLAS for Spark MLlib

Wednesday, June 19, 2019

10:33 AM

Spark MLlib depends on netlib-java ([https://github.com/fommil/netlib-java](https://github.com/fommil/netlib-java#linux)), which searches for native optimized BLAS/LAPACK libraries and uses them if available. If not, it uses a reference implementation (low performance) BLAS/LAPACK. This recipe discusses how to configure MKL as the native library to be used by MLlib.

* 1. Download and install MKL. Intel BB employees can get MKL from <http://softwareproducts.intel.com/ILC/>. For the rest of this discussion, we assume the installation location is /opt/intel/mkl. Make sure the MKL installation is visible on each node of your cluster.
  2. Compile and install OpenBlas on each node of cluster:

git clone <https://github.com/xianyi/OpenBLAS>

cd OpenBLAS

make FC=gfortran

make install

* 1. Netlib-java native libs have a dependency on libgfortran. It requires GFORTRAN 1.4 or above. So make sure you have it on your system. Find where libgfortran is located on your system. Do this test on each node of your cluster.

strings /path/to/lib64/libgfortran.so.3.0.0|grep GFORTRAN\_1.4  
  
If you don't see GFORTRAN\_1.4 then stop! Update your GCC toolchain on all your nodes, then come back here to continue. You can also install it by yourself. Like yum install gcc-gfortran.

* 1. Create symbolic links. Do this on each node.

sudo update-alternatives --install /usr/lib64/libblas.so libblas.so /opt/intel/mkl/lib/intel64/libmkl\_rt.so 1000

sudo update-alternatives --install /usr/lib64/libblas.so.3 libblas.so.3 /opt/intel/mkl/lib/intel64/libmkl\_rt.so 1000

sudo update-alternatives --install /usr/lib64/liblapack.so liblapack.so /opt/intel/mkl/lib/intel64/libmkl\_rt.so 1000

sudo update-alternatives --install /usr/lib64/liblapack.so.3 liblapack.so.3 /opt/intel/mkl/lib/intel64/libmkl\_rt.so 1000

sudo update-alternatives --install /usr/lib64/libgfortran.so.3 libgfortran.so.3 /path/to/lib64/libgfortran.so.3.0.0 1000

also need to create symbolic links for openblas

sudo update-alternatives --install /usr/lib64/libblas.so libblas.so /opt/OpenBLAS/lib/libopenblas.so 1000

sudo update-alternatives --install /usr/lib64/libblas.so.3 libblas.so.3 /opt/OpenBLAS/lib/libopenblas.so 1000

sudo update-alternatives --install /usr/lib64/liblapack.so liblapack.so /opt/OpenBLAS/lib/libopenblas.so 1000

sudo update-alternatives --install /usr/lib64/liblapack.so.3 liblapack.so.3 /opt/OpenBLAS/lib/libopenblas.so 1000

To switch between mkl and openblas, firstly change the configuration by:

sudo update-alternatives --config libblas.so

sudo update-alternatives --config libblas.so.3

sudo update-alternatives --config liblapack.so

sudo update-alternatives --config liblapack.so.3

To check use correct config as expected, should run:

update-alternatives --display libblas.so

libblas.so - status is manual.

link currently points to /opt/OpenBLAS/lib/libopenblas.so

Then update spark-env.sh like:

export LD\_LIBRARY\_PATH=$(echo $LD\_LIBRARY\_PATH):/opt/OpenBLAS/lib

and

export LD\_LIBRARY\_PATH=$(echo $LD\_LIBRARY\_PATH):/opt/intel/mkl/lib/intel64/

* 1. Take care of ldconfig. Do this on each node.
     + Go to /etc/ld.so.conf.d/, add a new file mkl-x86\_64.conf. This file should contain these 2 lines:

/opt/intel/mkl/compiler/lib/intel64\_lin

/opt/intel/mkl/lib/intel64

* 1. Save the file and then:

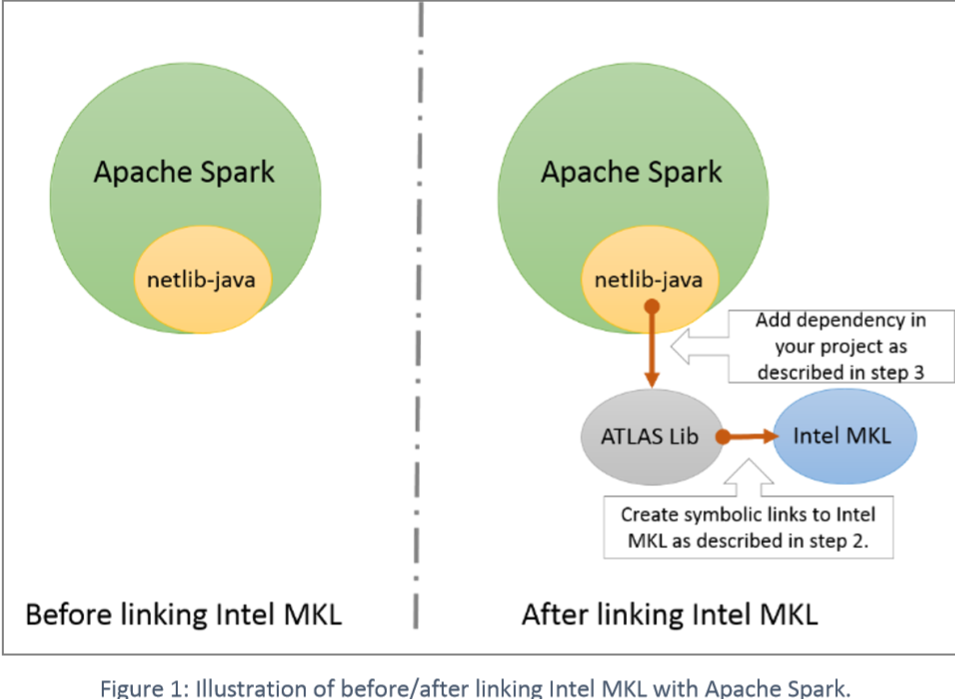
sudo ldconfig

* 1. Check if libmkl\_rt.so is the default system BLAS/LAPACK:

update-alternatives --display libblas.so

update-alternatives --display liblapack.so

If you don't see libmkl\_rt.so then stop, fix the problem before continue.

1. 
2. Build Spark (version 1.5.0 or above) from source with -Pnetlib-lgpl profile.

Download the sources of Spark and build it.

1. Spark configuration. Add the following to spark-env.sh:

export LD\_LIBRARY\_PATH=$(echo $LD\_LIBRARY\_PATH):/path/to/hadoop/lib/native

export LD\_LIBRARY\_PATH=$(echo $LD\_LIBRARY\_PATH):/opt/intel/mkl/lib/intel64

* 1. A simple sanity check to see if native BLAS/LAPACK libs can be properly loaded:

/opt/spark/bin/spark-shell

scala> import com.github.fommil.netlib.BLAS;

scala> System.out.println(BLAS.getInstance().getClass().getName());

If it is correctly loaded,then it should print:

Com.github.fommil.netlib.NativeSystemBLAS

If it is not loaded,then it should print:

WARN BLAS: Failed to load implementation from:com.github.fommil.netlib.NativeSystemBLAS

 WARN BLAS: Failed to load implementation from:com.github.fommil.netlib.NativeRefBLAS

com.github.fommil.netlib.F2jBLAS

One reason for the problem “WARN BLAS”:Failed to load implementation

From:com.github.fommil.netlib.NativeSystemBLAS” is that the GCCv4.8.2 or higher is missing.GCCv4.8.2 installation also installs gfortranv4.8.2

Another reason is that:your Spark have not been builded and netlib could’t be used.

So check if your spark is well builded.

* 1. To verify that MKL is being used for your application, you need to set spark.executorEnv.MKL\_VERBOSE=1. Then, you should look the log dir on each Worker node (Not on the master node!) for MKL\_VERBOSE messages. These messages tell you which MKL functions are used, and information associated to each call (such as matrix/vector size and threading information). If you don't see any MKL\_VERBOSE message, then there are two possibilities:
     1. The ML algorithms in your application do not use BLAS/LAPACK. MKL has no benefit in this case.
     2. Native libs are not loaded properly. See step 7 above.

Besides, you can also add “export MKL\_VERBOSE=1” to spark-env.sh insteaded.

When you run a spark job, if MKL is invoked, one should see something similar as follows:

MKL\_VERBOSE Intel(R) MKL 2019.0 Update 4 Product build 20190411 for Intel(R) 64 architecture Intel(R) Advanced Vector Extensions (Intel(R) AVX) enabled processors, Lnx 2.70GHz lp64 intel\_thread

MKL\_VERBOSE DSPR(U,10,0x7f93adc9cb40,0xeeb21f78,1,0xf6411428) 1.72ms CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:24

MKL\_VERBOSE DSPR(U,10,0x7f93adc9cb40,0xeeb381e0,1,0xf6411428) 3.63us CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:24

MKL\_VERBOSE DSPR(U,10,0x7f93adc9cb40,0xeeb38ff8,1,0xf6411428) 1.18us CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:24

MKL\_VERBOSE DSPR(U,10,0x7f93adc9cb40,0xeeb39e18,1,0xf6411428) 946ns CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:24

MKL\_VERBOSE DSPR(U,10,0x7f93adc9cb40,0xeeb3ac40,1,0xf6411428) 931ns CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:24

MKL\_VERBOSE DSPR(U,10,0x7f93adc9cb40,0xeeb3ba88,1,0xf6411428) 1.61us CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:24

* 1. Remember to remove spark.executorEnv.MKL\_VERBOSE (or set it to 0) before benchmarking! This setting is for debugging purposes and can lead to serious overhead.
  2. There are other MKL specific env-variables that users may need to tweak to get optimal performance. The most important one is MKL\_NUM\_THREADS. This defines the number of threads used by each MKL function call. When left unset, each MKL call by default uses the number of threads equal to the number of physical cores on a node. This may or may not be desired. Users need to experiment with this to find proper settings for different applications.
  3. To use MKL AVX2, you can add “export MKL\_ENABLE\_INSTRUCTIONS=AVX2” in spark\_env.sh on each node. By default, MKL will use AVX512. You can check library tree(lsof -p jobprocessID|grep AVX) to verify which AVX used or through emon data.
  4. If you want install MKL as a dependency in your project:

Follow instructions listed here: https://github.com/fommil/netlib-java#installation

Troubleshooting tips:

* + Are you using 64-bit or 32-bit libs? Make sure MKL libs and libgfortran are both 64-bit, or both 32-bit.
  + libgfortran version must be GLIBFORTRAN\_1.4 or above.
  + MKL libs must be visible on all nodes. And MKL path must be identical on all nodes.

Problems you may meet:

/home/spark01/app/jdk1.8.0\_201/bin/java: symbol lookup error: /tmp/jniloader94322606877265804netlib-native\_system-linux-x86\_64.so: undefined symbol: cblas\_dspr

you can try:

nm /usr/lib/libblas.so.3 | grep cblas\_dspr

if your lib have cblas\_dspr ,you can see:

000000000018cc60 T cblas\_dspr

000000000018cc60 T cblas\_dspr\_

000000000018cd40 T cblas\_dspr2

000000000018cd40 T cblas\_dspr2\_

If your libblas.so.3 have this message, you may need add the following configure to spark-env.sh:

export LD\_LIBRARY\_PATH=$(echo $LD\_LIBRARY\_PATH):/usr/lib

If not,please see step 7 above.

Error: libmkl\_intel\_thread.so: undefined symbol: omp\_get\_num\_procs refer <https://software.intel.com/en-us/forums/intel-system-studio/topic/611682>

This error is because OpenMP runtime is not found on LD\_LIBRARY\_PATH. Adding the location of libiomp5.so to LD\_LIBRARY\_PATH will fix the problem.