cloudera®

Cloudera Data Science

Important Notice

Cloudera, Inc.
395 Page Mill Road
Palo Alto, CA 94306
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

Version: Cloudera Data Science 1.0.x
Date: January 28, 2019

Table of Contents

Cloudera Data Science

Welcome to the Guru How-To's for Cloudera Data Science!

Only Cloudera offers a modern enterprise platform, tools, and expert guidance to help you unlock business value with machine learning and AI. Cloudera's modern platform for machine learning and analytics, optimized for the cloud, lets you build and deploy AI solutions at scale, efficiently and securely, anywhere you want. Cloudera Fast Forward Labs expert guidance helps you realize your AI future—faster!

Use these Cloudera Data Science Guru How-To's to get started building your AI solutions.

Cloudera Data Science POWERED BY…	
Cloudera Data Science Workbench (CDSW)	Accelerate data science from research to production on a collaborative platform for machine learning and AI. CDSW provides on-demand access to runtimes for R, Python, and Scala, plus high-performance integration with Apache Spark with secure connectivity to CDH. For deep learning and other demanding data science techniques, CDSW supports GPU-accelerated computing, so data scientists can use deep learning frameworks like TensorFlow, Apache MXNet, Keras. and more.
Apache Spark	Provides flexible, in-memory data processing, reliable stream processing, and rich machine learning tooling for Hadoop.
Cloudera Fast Forward Labs	Cloudera Fast Forward Labs helps you design and execute your enterprise machine learning strategy, enabling rapid, practical application of emerging machine learning technologies to your business. In addition, Cloudera Professional Services offer proven delivery of scalable, production-grade machine learning systems.

Using Native Math Libraries to Accelerate Spark Machine Learning Applications

by Zuling Kang

Spark ML is one of the dominant frameworks for many major machine learning algorithms, such as the Alternating Least Squares (ALS) algorithm for recommendation systems, the Principal Component Analysis algorithm, and the Random Forest algorithm. However, frequent misconfiguration means the potential of Spark ML is seldom fully utilized. Using native math libraries for Spark ML is a method to achieve that potential.

This article discusses how to accelerate model training speed by using native libraries for Spark ML. It also discusses why Spark ML benefits from native libraries, how to enable native libraries with CDH Spark, and provides performance comparisons between Spark ML on different native libraries.

Native Math Libraries for Spark ML

Spark's MLlib uses the Breeze linear algebra package, which depends on `netlib-java` for optimized numerical processing. `netlib-java` is a wrapper for low-level BLAS, LAPACK, and ARPACK libraries. However, due to licensing issues with runtime proprietary binaries, neither the Cloudera distribution of Spark nor the community version of Apache Spark includes the `netlib-java` native proxies by default. So if you make no manual configuration, `netlib-java` only uses the F2J library, a Java-based math library that is translated from Fortran77 reference source code.

To check whether you are using native math libraries in Spark ML or the Java-based F2J, use the Spark shell to load and print the implementation library of `netlib-java`. For example, the following commands return information on the BLAS library and include that it is using F2J in the line, `com.github.fommil.netlib.F2jBLAS`, which is bolded below:

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

scala> println(BLAS.getInstance().getClass().getName())
18/12/10 01:07:06 WARN netlib.BLAS: Failed to load implementation from:
com.github.fommil.netlib.NativeSystemBLAS
18/12/10 01:07:06 WARN netlib.BLAS: Failed to load implementation from:
com.github.fommil.netlib.NativeRefBLAS
com.github.fommil.netlib.F2jBLAS
```

Adoption of Native Libraries in Spark ML

Anand Iyer and Vikram Saletore showed in their engineering blog post that native math libraries like OpenBLAS and Intel's Math Kernel Library (MKL) accelerate the training performance of Spark ML. However, the range of acceleration varies from model to model.

As for the matrix factorization model used in recommendation systems (the Alternating Least Squares (ALS) algorithm), both OpenBLAS and Intel's MKL yield model training speeds that are 4.3 times faster than with the F2J implementation. Others, like the Latent Dirichlet Allocation (LDA), the Primary Component Analysis (PCA), and the Singular Value Decomposition (SVD) algorithms show 56% to 72% improvements for Intel MKL, and 10% to 50% improvements for OpenBLAS.

However, the blog post also demonstrates that there are some algorithms, like Random Forest and Gradient Boosted Tree, that receive almost no speed acceleration after enabling OpenBLAS or MKL. The reasons for this are largely that the training set of these tree-based algorithms are not vectors. This indicates that the native libraries, either OpenBLAS or MKL, adapt better to algorithms whose training sets can be operated on as vectors and are computed as a whole. It is more effective to use math acceleration for algorithms that operate on training sets using matrix operations.

How To Enable Native Libraries

The following sections show how to enable `libgfortran` and MKL for Spark ML. CDH 5.15 on RHEL 7.4 was used in the example and to produce the performance comparisons. The same functionality is available in CDH 6.x.

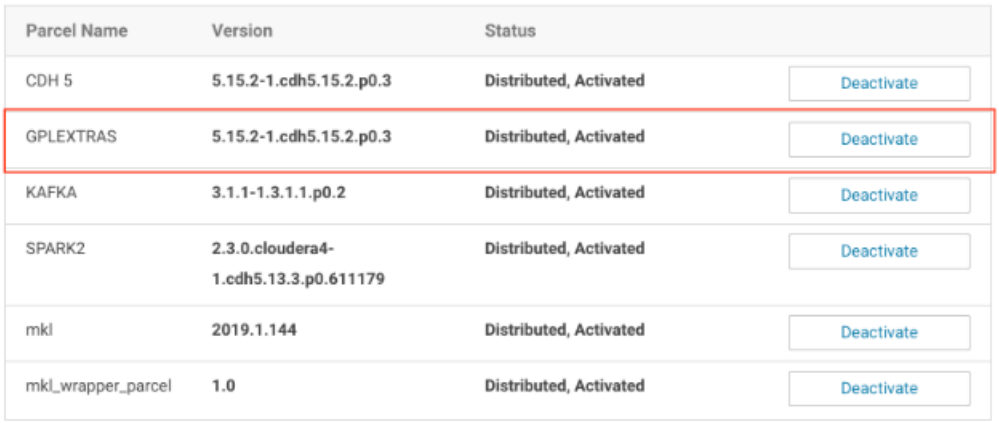
Enabling the libgfortran Native Library

> **Important:** The following instructions work for Spark 1.6x in CDH 5.x and for Spark 2.x in CDH 6.x. In those versions, GPLEXTRAS automatically adds the classpath of JAR files needed to use the native libraries to `/etc/spark/conf/classpath.txt`. Then the `spark-env.sh` loads the extra Java libraries during bootstrap. However, GPLEXTRAS cannot do this for Spark 2.x in CDH 5.x. If you want to use Spark 2.x, you must upgrade to CDH 6.x, which automatically performs this configuration for Spark 2.x.

1. Enable the `libgfortran` 4.8 library on every CDH node. For example, in RHEL, run the following command on each node:

```
yum -y install libgfortran
```

2. Install the GPLEXTRAS parcel in Cloudera Manager, and activate it:
   a. To install the GPLEXTRAS parcel, see Installing the GPL Extras Parcel in the Cloudera Manager documentation.
   b. To activate the package, see Activating a Parcel.
   c. After activating the GPLEXTRAS parcel, in Cloudera Manager, navigate to **Hosts** > **Parcels** to confirm that the GPLEXTRAS parcel is activated:



   The GPLEXTRAS parcel acts as the wrapper for `libgfortran`.

3. Restart the appropriate CDH services as guided by Cloudera Manager.
4. As soon as the restart is complete, use the Spark shell to verify that the native library is being loaded by using the following commands:

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

scala> println(BLAS.getInstance().getClass().getName())
18/12/23 06:29:45 WARN netlib.BLAS: Failed to load implementation from:
com.github.fommil.netlib.NativeSystemBLAS
18/12/23 06:29:45 INFO jni.JniLoader: successfully loaded
/tmp/jniloader6112322712373818029netlib-native_ref-linux-x86_64.so
```

```
com.github.fommil.netlib.NativeRefBLAS
```

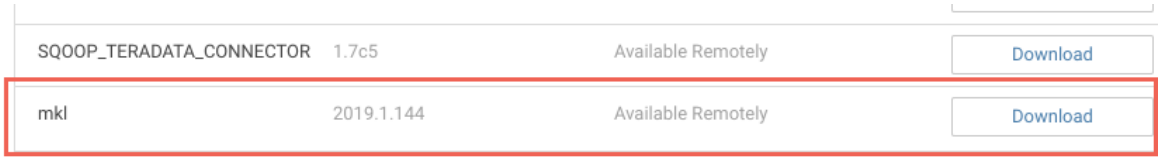
You might be wondering why there is still a warning message about `NativeSystemBLAS` failing to load. Don't worry about this. It is there because we are only setting the native library that Spark uses not for system-wide use. You can safely ignore this warning.

Enabling the Intel MKL Native Library

1. Intel provides the MKL native library as a Cloudera Manager parcel on its website. You can add it as a remote parcel repository in Cloudera Manager. Then you can download the library and activate it:
   a. In Cloudera Manager, navigate to **Hosts** > **Parcels**.
   b. Select **Configuration**.
   c. In the section, **Remote Parcel Repository URLs**, click the plus sign and add the following URL:

```
http://parcels.repos.intel.com/mkl/latest
```

   d. Click **Save Changes**, and then you are returned to the page that lists available parcels.
   e. Click **Download** for the `mkl` parcel:



   f. Click **Distribute**, and when it finishes distributing to the hosts on your cluster, click **Activate**.

2. The MKL parcel is only composed of Linux shared library files (`.so` files), so to make it accessible to the JVM, a JNI wrapper has to be made. To make the wrapper, use the following MKL wrapper parcel. Use the same procedure described in Step 1 to add the following link to the Cloudera Manager parcel configuration page, download the parcel, distribute it among the hosts and then activate it:

```
https://raw.githubusercontent.com/Intel-bigdata/mkl-wrappers-parcel-repo/master/
```

3. Restart the corresponding CDH services as guided by Cloudera Manager, and redeploy the client configuration if needed.
4. In Cloudera Manager, add the following configuration information into the **Spark Client Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-defaults.conf**:

```
spark.driver.extraJavaOptions=-Dcom.github.fommil.netlib.BLAS=com.intel.mkl.MKLBLAS
-Dcom.github.fommil.netlib.LAPACK=com.intel.mkl.MKLLAPACK
spark.driver.extraClassPath=/opt/cloudera/parcels/mkl_wrapper_parcel/lib/java/mkl_wrapper.jar
spark.driverEnv.MKL_VERBOSE=1
spark.executor.extraJavaOptions=-Dcom.github.fommil.netlib.BLAS=com.intel.mkl.MKLBLAS
-Dcom.github.fommil.netlib.LAPACK=com.intel.mkl.MKLLAPACK
spark.executor.extraClassPath=/opt/cloudera/parcels/mkl_wrapper_parcel/lib/java/mkl_wrapper.jar
spark.executorEnv.MKL_VERBOSE=1
```

This configuration information instructs the Spark application to load the MKL wrapper and use MKL as the default native library for Spark ML.

Important:

- By setting `MKL_VERBOSE=1`, MKL logs what computational functions are called, what parameters are passed to them, and how much time is spent to execute the functions. This information can be useful for implementation, but it consumes large amounts of space on HDFS in your cluster. In my experimental cases that are discussed in the following section, the logs for each job could consume hundreds of GBs of space.
- If the `UnsatisfiedLinkError` message is returned when verifying the native library being used as shown below, add the `/opt/cloudera/parcels/mkl/linux/mkl/lib/intel64` directory to the `LD_LIBRARY_PATH` environment variable for each cluster node.

```
Native code library failed to load.
java.lang.UnsatisfiedLinkError:
/opt/cloudera/parcels/mkl_wrapper_parcel-1.0/lib/native/mkl_wrapper.so:
 libmkl_rt.so: cannot open shared object file: No such file or
directory
```

5. Open the Spark shell again to verify the native library, and you should see the following output:

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

scala> println(BLAS.getInstance().getClass().getName())
com.intel.mkl.MKLBLAS
```

Performance Comparisons

In this section, we use the ALS algorithm to compare the training speed with different underlying math libraries, including F2J, `libgfortran`, and Intel's MKL.

The hardware we are using are the `r4.large` VM instances from Amazon EC2, with 2 CPU cores and 15.25 GB of memory for each instance. In addition, we are using CentOS 7.5 and CDH 5.15.2 with the Cloudera Distribution of Spark 2.3 Release 4. The training code is taken from the core part of the ALS chapter of *Advanced Analytics with Spark* (2nd Edition) by Sandy Ryza, et al, O'Reilly (2017). The training data set is the one published by Audioscrobbler, which can be downloaded at:

```
https://storage.googleapis.com/aas-data-sets/profiledata_06-May-2005.tar.gz
```

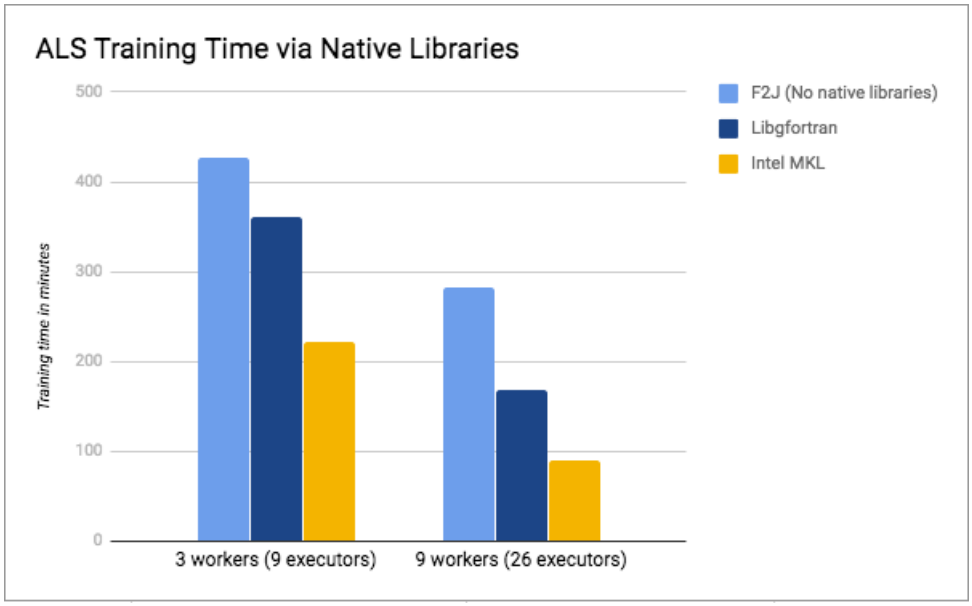
Usually the rank of the ALS model is set to a much larger value than the default of 10, so we use the value of 200 here to make sure that the result is closer to real world examples. Below is the code used to set the parameter values for our ALS model:

```
val model = new ALS().
     setSeed(Random.nextLong()).
     setImplicitPrefs(true).
     setRank(200).
     setRegParam(0.01).
     setAlpha(1.0).
     setMaxIter(20).
     setUserCol("user").
     setItemCol("artist").
     setRatingCol("count").
```

```
    setPredictionCol("prediction")
```

The following table and figure shows the training time when using different native libraries of Spark ML. The values are shown in minutes. We can see that both `libgfortran` and Intel's MKL do improve the performance of training speed, and MKL seems to outperform even more. From these experimental results, `libgfortran` improves by 18% to 68%, while MKL improves by 92% to 213%.

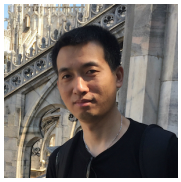
# of Workers/Executors	F2J	libgfortran	Intel MKL
3 workers (9 executors)	426	360	222
9 workers (26 executors)	282	168	90



Acknowledgement

Many thanks to Alex Bleakley and Ian Buss of Cloudera for their helpful advice and for reviewing this article!

About the Author



Zuling Kang is a Senior Solutions Architect at Cloudera, Inc., and holds a Ph.D. in Computer Science. Before joining Cloudera, he worked as an architect of big data systems at China Mobile Zhejiang Co., Ltd. Currently, he has published nine academic/technical papers, of which seven are indexed by the Science Citation Index (SCI)/Ei Compendex (formerly the Engineering Index). One of these papers, "Performance-Aware Cloud Resource Allocation via Fitness-Enabled Auction," is published in the "IEEE Transactions on Parallel and Distributed Systems." Zuling's current research and engineering interests include architectures for big data platforms, big data processing technologies, and machine learning.

Appendix: Apache License, Version 2.0

SPDX short identifier: Apache-2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims

licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```