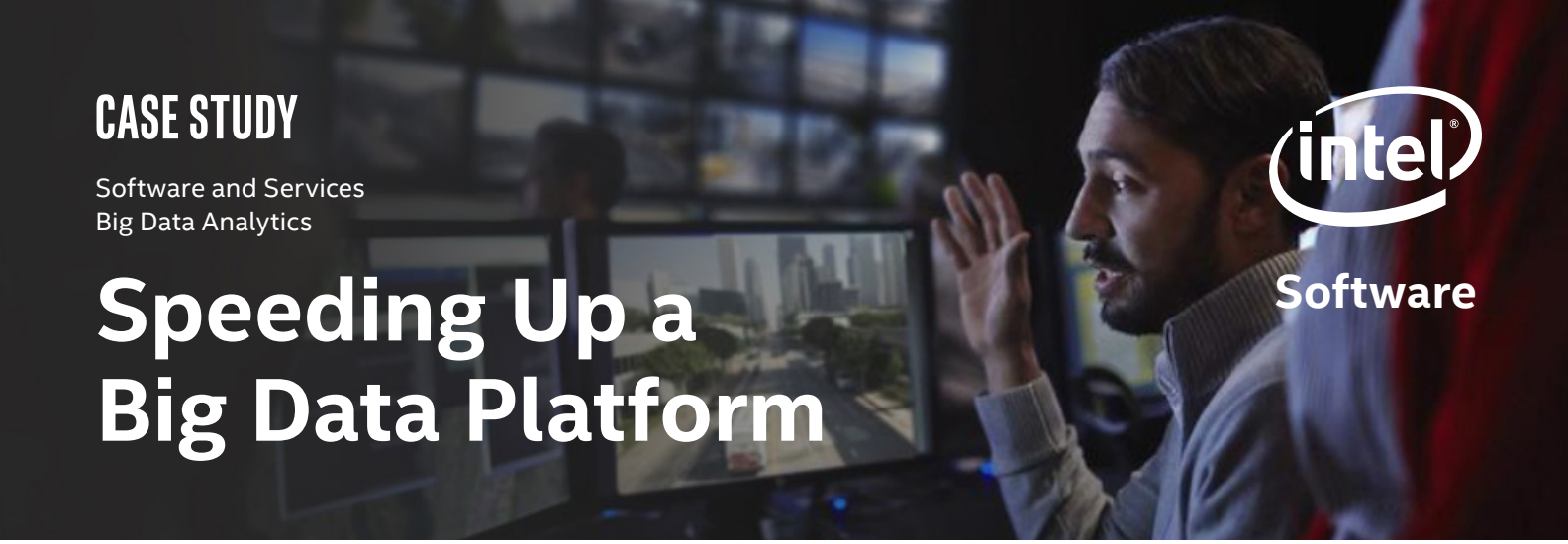


CASE STUDY

Software and Services
Big Data Analytics

Speeding Up a Big Data Platform

intel Software

MeritData Inc. improves performance—and the potential for big data algorithms and visualization—with high-performance Intel® libraries

"Through close collaboration with Intel engineers, we adopted the Intel® Data Analytics Acceleration Library and Intel® Math Kernel Library for algorithm optimization in our big data analysis platform Tempo*. The performance and customers' experience are improved signifcantly. We are looking forward to more collaboration."

—Jin Qiang
Data Mining Algorithm Architect
MeritData Inc.

MeritData, Inc. is a top big data analysis technology and service provider in China. Its Tempo* big data platform is widely used in industries like power, manufacturing, and financial services by well-known enterprises and cloud service providers. Fusing high-performance computing technology, leading data analysis algorithms, high-dimensional visualization, and creative data visualization language, MeritData helps its customers explore and exploit data values—and ultimately make value flow effectively for both internal and external users— through a suite of data processing, data mining, and data visualization solutions.

To analyze data quickly and precisely, MeritData must optimize its algorithms from all perspectives of algorithm principles, computing, and programming. Intel worked with MeritData algorithm engineers to optimize the company's multiple data mining algorithms including Extreme Learning Machines (EML), the in-house L1/2 Sparse Iteration Algorithm, Linear Regression (LR), QR matrix factorization, and Principle Component Analysis (PCA). After optimizing the algorithms with Intel® Data Analytics Acceleration Library (Intel® DAAL) and Intel Math Kernel Library (Intel® MKL) performance improved by an average of 3x, with peak performance gains of 14x.

The Need for Speed in Big Data Analysis Platforms

Globally, the volume of data is growing exponentially—and with it, the market for big data analysis and vigorous big data services. Almost every large company is investing resources in big data analysis and expects to integrate years of both historic and newly generated data and to quickly and accurately mine its value.

MeritData's Tempo big data analysis platform stores and processes multiple data types for its customers. To analyze all this data effectively, and to process increasingly large data sets faster, the company needed to significantly improve algorithm performance.

Algorithm modeling can help with a high computation load that needs to perform repeated computation iterations on input data. When the data volume is small, operating time is usually acceptable. But as data volumes surge, the operating time of some algorithms increases exponentially—until they can no longer meet customer requirements.

By working closely with Intel, and using Intel MKL and Intel DAAL, MeritData was able to accelerate Tempo's core algorithm library, giving its customers a powerful data analysis solution. Compared with the original hardware-independent implementation, the new scheme can quickly and accurately analyze huge amounts of data processing and modeling, enabling customers to quickly explore and realize the full value of their data.

MERITDATA

Solution: Tempo Big Data Analysis Platform

Intel worked with MeritData to build a Tempo big data analysis platform, based on Intel MKL and Intel DAAL, to accelerate the core algorithm libraries on Intel® architecture. Through the cloud computing architecture, the team implemented a big data analysis solution for fast modeling and analysis. At the same time, it provided integrated services to satisfy customers with different fields and levels of data analysis to achieve the value of their data, data visualization exploration, and in-depth analysis.

The system architecture of the Tempo platform includes the data access layer, analysis and modeling, result display, and access layer, providing unifed cloud service access, cloud resource scheduling, and cloud platform management.

The data access layer needed to be able to accept different sources of data, including intelligently adapting to both SQL* and non-SQL databases, docking with Kafka*, flume streaming data sources, and unstructured text data sources.

Analysis and Modeling Layer

Based on Intel MKL and Intel DAAL, MeritData accelerated the core analysis algorithms of Tempo. The intelligent data analysis platform:

- Manages node scheduling
- Analyzes assigned compute nodes
- Writes the related job log

Result Display and Access Layer

After analysis modeling is completed, the modeling results are displayed as statistics and can generate a result report to improve resource utilization and support agile decision-making. They also allow the customer to build a new model based on pretrained results, directly accepting the new data sources and making accurate predictions. This layer provides an access interface in the form of an API call, supporting future development.

The Technical Advantage

Intel MKL provides a wide range of matrix, vector, and math processing routines that are highly optimized for multilevel parallelism on Intel architecture. These routines make efficient use of available resources in multicore, many-core, and clustered architectures and automatically balance workloads across Intel® Xeon® processors. The highly optimized functions (Figure 2) are widely used in performance-demanding applications for science, engineering, and financial services. These industries can then make full use of Intel® multicore processors to maximize the performance of applications and reduce development time.

Like Intel MKL, Intel DAAL is a high-performance library. However, it has more functionality than Intel MKL for big data, helping speed big data analytics by providing highly optimized

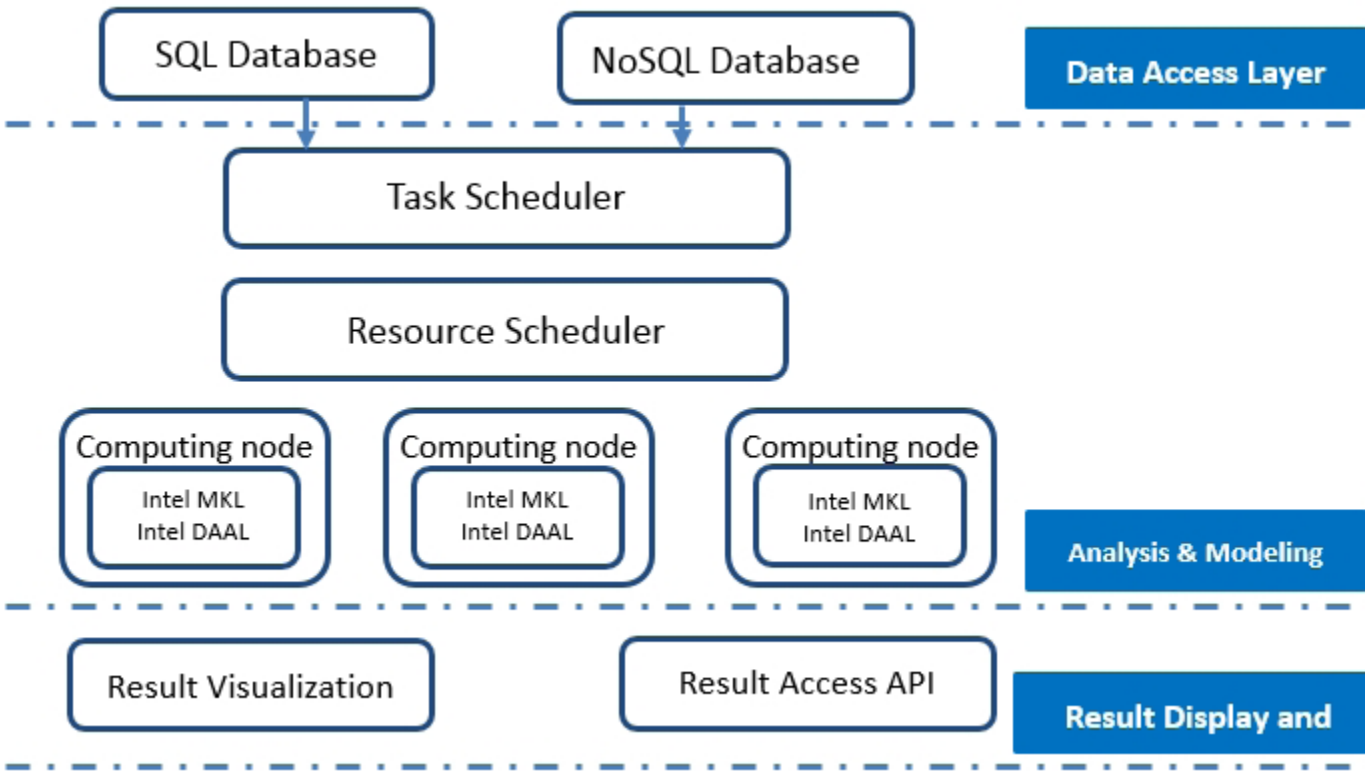


Figure 1. Tempo* big data analysis platform for data-mining system architecture

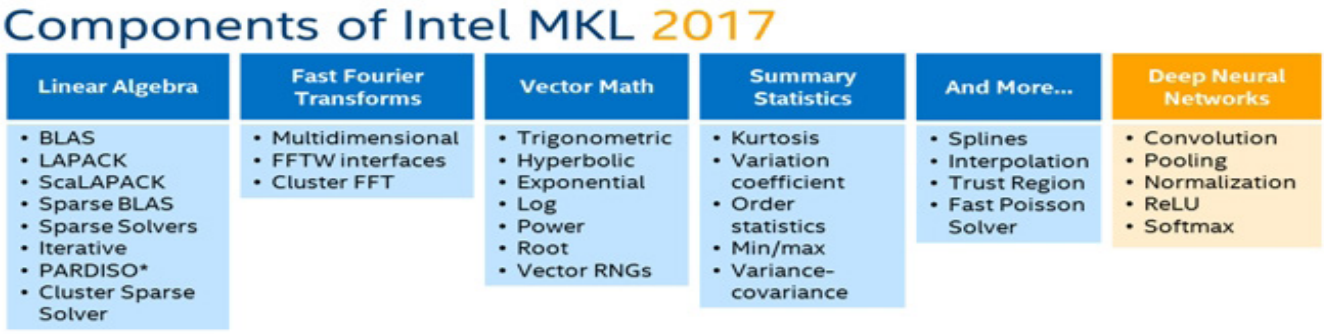


Figure 2. Intel® Math Kernel Library (Intel® MKL) components

algorithmic building blocks for all data analysis stages (preprocessing, transformation, analysis, modeling, validation, and decision-making) for ofine, streaming, and distributed analytics usages. It is designed for highly efficient data access for popular data platforms including Hadoop*, Spark*, R*, and MATLAB*.

Intel DAAL (Figure 3) provides a rich set of algorithms, ranging from the most basic descriptive statistics for data sets to more advanced data mining and machine learning algorithms. It can help big data developers create highly optimized code for many big data algorithms with relatively little effort.

Boosting Tempo's Performance

With Intel MKL and Intel DAAL, MeritData accelerated core algorithms of the Tempo large data analysis platform over the original method. Performance was boosted dramatically, effectively helping customers maximize their data value, meet business needs, and quickly analyze huge amounts of new data.

Specifcally, the team was able to optimize Tempo's extreme learning machines (EML) and in-house L1/2 sparse iterative algorithms.

Extreme Learning Machines Algorithm

ELM is one of MeritData's data mining algorithms. The team profled the code of the ELM algorithm and found the performance bottleneck was matrix computation, including inversion of functions such as matrix and matrix multiply. Originally, MeritData used the Eigen* library, which provided multiple matrix operations but offered less performance optimization than IntelMKL. MeritData replaced the related functions with Intel MKL functionality. The code replacement was straightforward (Table 1).

Results for using different amounts of data on the ELM algorithm for testing are shown in Figure 4. The optimized algorithm performance boost averaged about 12x.

In-House L1/2 Sparse Iterative Algorithm

After test and analysis, the team found the performance bottleneck in this algorithm was the kernel function and basic matrix operations. For basic matrix operations, they used the Intel MKL BLAS function. For kernel function, they used the Intel DAAL Kernel functionality. Table 2 shows the exact optimization before and after switching to Intel DAAL.

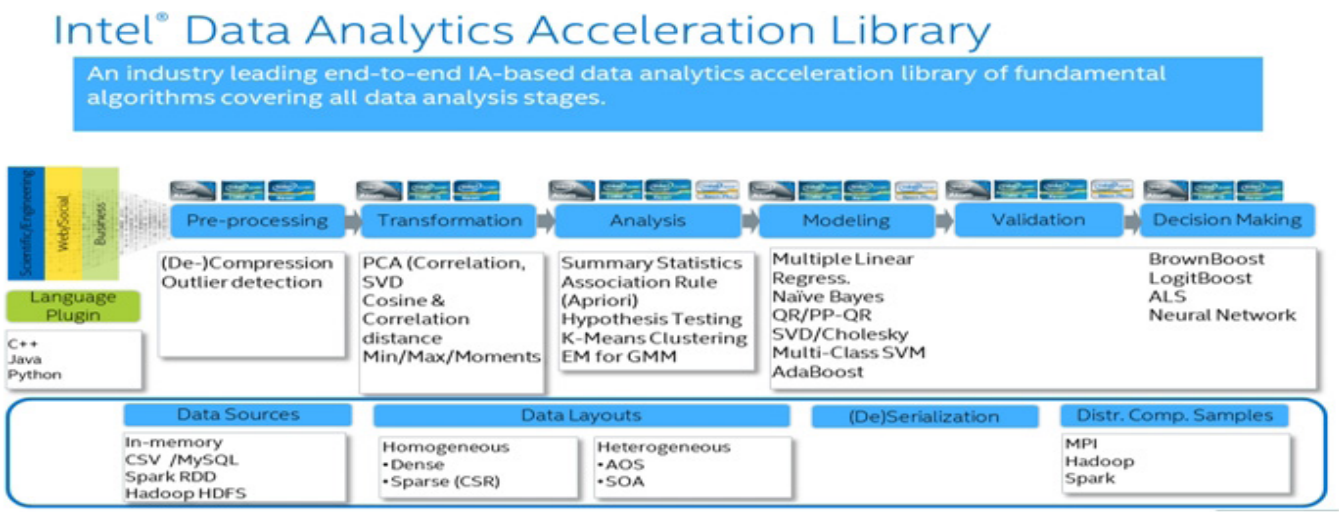


Figure 3. Intel® Data Analytics Acceleration Library components

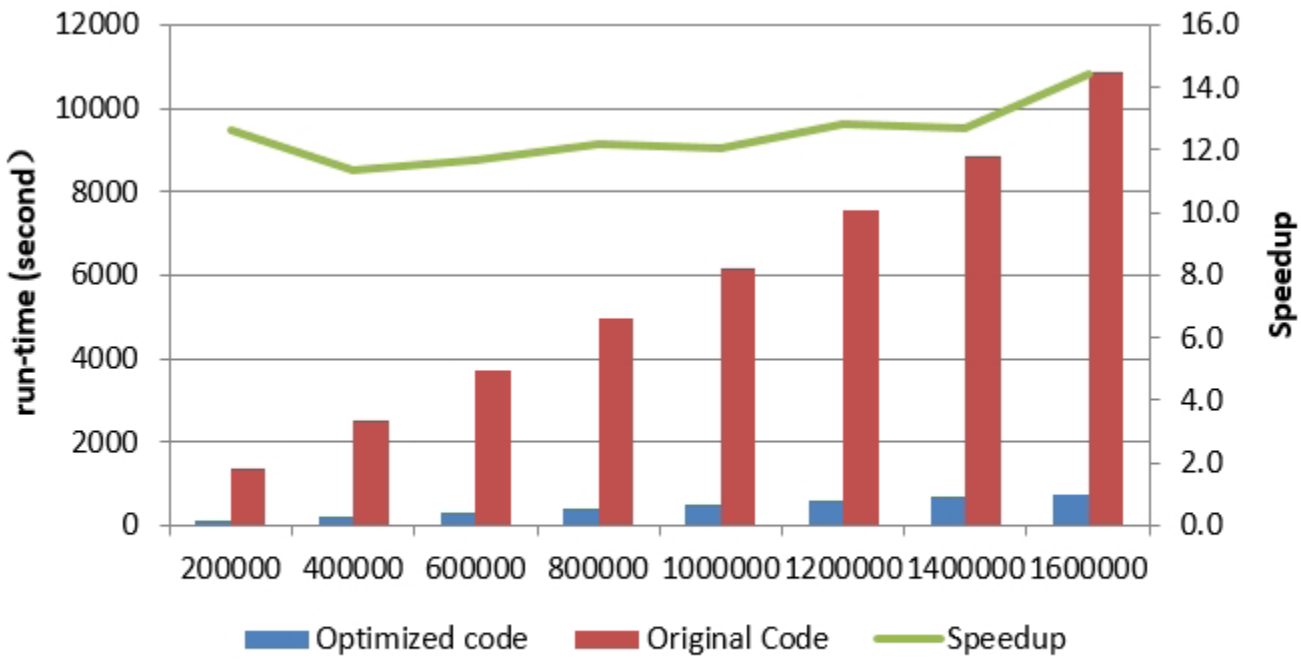
Original Version

```
Eigen::MatrixXd Pk;
Eigen::MatrixXd Bk;
jobjectArray     update(JNIEnv*
env,jobject,jobjectArray
HM,jobjectArray TM){
// get row num of data.
int height = env->GetArrayLength(HM);
// get column num of data.
…
int width = env->GetArrayLength(tmp);
// internal matrix
Eigen::MatrixXd mat(height,width);
…
// TM matrix
Eigen::MatrixXd Tm(row2,col2);
…
// do caculations
Eigen::MatrixXd     matTran     =
mat.transpose();
Eigen::MatrixXd matTmp = matTran*mat;
...
// inverse matrix
Pk = matTmp.inverse();
Bk = Pk*matTran*Tm;
…
}
```

Optimized by Intel® MKL

```
double* Pk;
double* Bk;
jobjectArray update(JNIEnv*
env,jobject,jobjectArray HM,jobjectArray TM){
// get row num of data.
int rowNum = env->GetArrayLength(HM);
// get column num of data.
…
int colNum = env->GetArrayLength(tmp);
// internal matrix
double* mat = (double
*)mkl_calloc(rowNum*colNum, sizeof( double ),
64);…
// TM matrix
double* Tm = (double
*)mkl_calloc(rowNumT*colNumT,
sizeof( double ), 64); …
// do caculations
double* matTmp = (double
*)mkl_calloc(colNum*colNum, sizeof( double ),
64);
MKL_INT lda = colNum,ldc = colNum;
cblas_dgemm(CblasRowMajor,CblasTrans,CblasNoT
rans,colNum,colNum,rowNum,1.0,mat,lda,mat,co
Num,0.0,matTmp,ldc);
...
// inverse matrix
MKL_INT* ipiv = (int *)mkl_calloc(colNum,
sizeof( int ), 32);
MKL_INT info =
LAPACKE_dgetrf(LAPACK_ROW_MAJOR,colNum,colNum
,matTmp,lda,ipiv);
info =
LAPACKE_dgetri(LAPACK_ROW_MAJOR,colNum,matTmp
,lda,ipiv);
…
}
```

Table 1. Comparison showing before and after the machine learning algorithm was used for code optimization



Configuration Info - Versions: Intel® Math Kernel Library (Intel MKL) 11.3.2; Hardware: Intel® Xeon CPU E5-2699 V3 2.30GHz, 2 sockets x 18 cores, AVX 2.0Supported. 45MB Cache, 128 GB Memory; Operating System: Centos6.7; 4 node cluster with spark-1.5.1, hadoop-2.6.0, jdk-7u79-linux-x64, scala-2.10.4. Benchmark Source: MeritData test code and test data set

Figure 4. Time comparison before and after the optimization with Intel® MKL

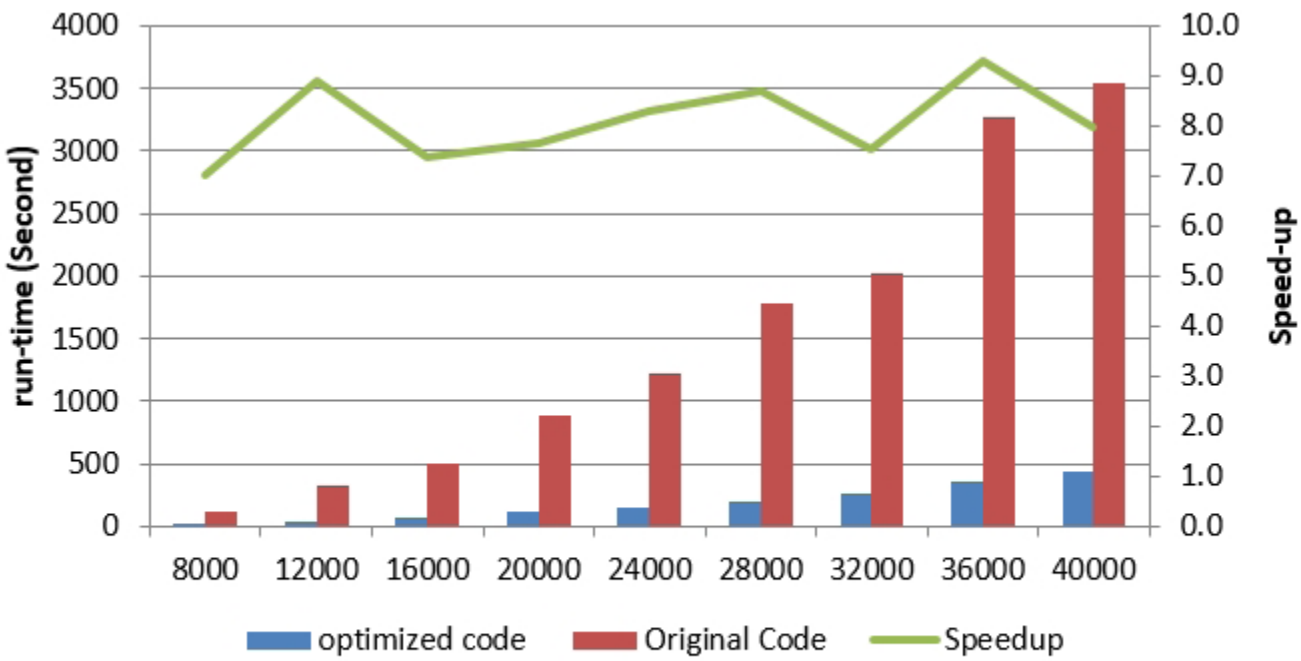
Original Code

```
public Matrix getKernelMatrix()
throws Exception {
Matrix result = new
Matrix(m_data.numInstances(),
m_data.numInstances(), 0);
for (int i = 0; i <
m_data.numInstances() - 1; i++) {
for (int j = i + 1; j <
m_data.numInstances(); j++) {
result.set(i, j, evaluate(i, j,
m_data.instance(0)));
}
}
result = result.plus(
result.transpose().plus(
Matrix.identity(m_data.numInstances()
,
m_data.numInstances()))).copy();
return result;
}
```

Optimized by Intel® DAAL

```
jobject getKernelMatrix(JNIEnv*
env,jobject,jdouble param,jint rows,jint
cols,jobject byteBuffer,jobject dstBuffer){
…
kernel_function::linear::Batch<>
linearKernel;
/* Set the kernel algorithm parameter */
linearKernel.parameter.k = 1.0;
linearKernel.parameter.b = 1.0;
linearKernel.parameter.computationMode =
kernel_function::matrixMatrix;
/* Set an input data table for the algorithm
*/
linearKernel.input.set(kernel_function::X,
data);
linearKernel.input.set(kernel_function::Y,
data);
/* Compute the linear kernel function */
linearKernel.compute();
/* Get the computed results */
services::SharedPtr<kernel_function::Result>
lkResult = linearKernel.getResult();
/* Get the results */
services::SharedPtr<NumericTable> lkMat =
lkResult->get(kernel_function::values);
BlockDescriptor<double> block;
lkMat->getBlockOfRows(0, rows, readOnly,
block);
...
}
```

Table 2. L1/2 sparse code before and after optimization with the iteration algorithm



Configuration Info - Versions: Intel Data Analysis Acceleration library from Parallel_studio_xe_2017_beta; Hardware: Intel® Xeon CPU E5-2699 V3 2.30GHz, 2 sockets x 18 cores, AVX 2.0Supported. 45MB Cache, 128 GB Memory; Operating System: Centos6.7; Benchmark Source: MeritData test code and test data set

Figure 5. L1/2 sparse iterative algorithm running-time comparison before and after the transformation

Using a different amount of data on the L1/2 sparse iterative algorithm for testing, optimized algorithm performance was boosted by a factor of approximately 8x (Figure 5).

Summary

With more and more need for in-depth understanding of data, MeritData's customers want to integrate years worth of both historic and newly generated data, and to mine the value in it using limited resources. The MeritData Tempo platform provides a fast and highly efficient solution for its customers' data mining tasks.

Through close collaboration with Intel engineers, MeritData adopted Intel DAAL and Intel MKL for algorithm optimization in Tempo—enabling customers to accurately analyze big data fast and model different types and levels of data analysis.

The solution greatly improved data analysis processing capacity, performance, and the user experience.

MeritData will continue working with Intel to optimize the Tempo large data analysis platform to maximmize the computation capability provided by Intel architecture. This will help MeritData developers deliver higher performance with less effort, so they can continue to provide the best possible user experience for their customers.

Learn More

Intel® Data Analytics Acceleration Library >

Intel® Math Kernel Library >

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation.

Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer, or learn more at www.intel.com.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to www.intel.com/performance.

Intel does not control or audit the design or implementation of third party benchmark data or Web sites referenced in this document. Intel encourages all of its customers to visit the referenced Web sites or others where similar performance benchmark data are reported and confirm whether the referenced benchmark data are accurate and reflect performance of systems available for purchase.

This document and the information given are for the convenience of Intel's customer base and are provided "AS IS" WITH NO WARRANTIES WHATSOEVER, EXPRESS OR IMPLIED, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. Receipt or possession of this document does not grant any license to any of the intellectual property described, displayed, or contained herein. Intel® products are not intended for use in medical, lifesaving, life-sustaining, critical control, or safety systems, or in nuclear facility applications.

Copyright © 2016 Intel Corporation. All rights reserved. Intel, Xeon, and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others. Printed in USA 1016/SS Please Recycle