Matrix factorization routines on heterogeneous architectures

Nikita Shustrov, Nadya Mozartova, Tamara Kashevarova, Konstantin Arturov, Intel Corporation

Introduction

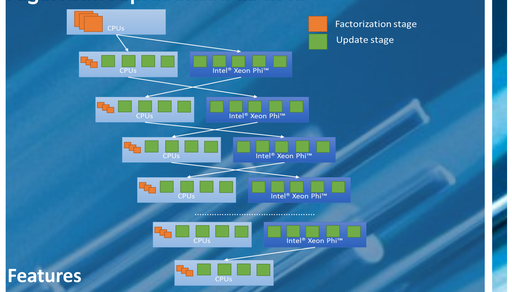
- New heterogeneous systems consisting of a multicore CPU with coprocessors introduce new challenges to designing efficient parallel algorithms.
- Simultaneous usage of all computational resources of such systems for solving one large problem requires uneven distribution of data and computations that leads to more complex parallelization methods.
- We present a new parallelization method for matrix factorization that efficiently utilizes all computational resources of a heterogeneous system consisting of a multicore CPU with coprocessors.

We show how this method can be applied for parallelization of the key linear algebra factorization algorithms (QR, LU and Cholesky) on systems with Intel® Xeon Phi™ coprocessors.

Algorithm - Overview

- Algorithm is based on the panel factorization method. It has advantages over the communication-avoiding method, the tile method, and their combination because it does not require:
  - additional computational cost
  - additional memory consumption
- Panel factorization has the same computational cost and memory usage as classic LAPACK algorithms.
- Implementation preserves LAPACK standard interfaces and data layout.
- Algorithm can be applied to any matrices.
- Implementation is DAG-based and uses panel factorization kernels that were redesigned and rewritten for new Intel Xeon Phi architectures.

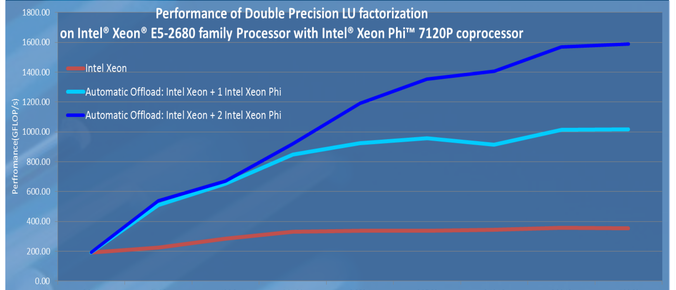
Algorithm represented as DAG



Features

- At the beginning, CPUs produce a number of factorized panels and send to coprocessors as many panels as needed to maximize coprocessor utilization.
- Coprocessors perform “update” stages in parallel.
- CPUs perform both “factorization” stages and “update” stages in parallel.
- To achieve the best load balance, a coprocessor may send a panel back to the CPU side on any process stage.

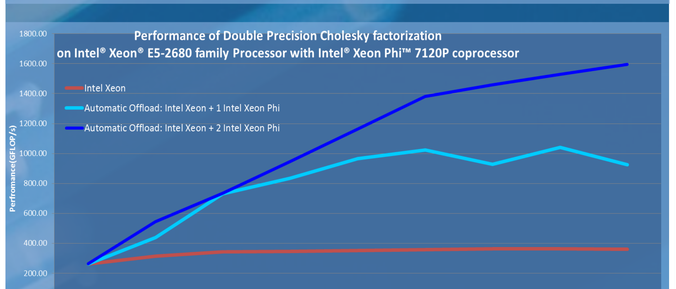
Performance results



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.0.5, MPSS 2.1.6720-13, Intel® Compiler 13.0, Hardware: Intel® Xeon® Processor E5-2680, 2 Eight-Core CPUs; Intel® Xeon Phi™ 7120P ; Benchmark Source: Intel Corporation.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to www.intel.com/performance/resources/benchmark_limitations.htm.

Optimization Notice: 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



Configuration Info - Versions: Intel® Math Kernel Library (Intel® MKL) 11.0.5, MPSS 2.1.6720-13, Intel® Compiler 13.0. Hardware: Intel® Xeon® Processor E5-2680, 2 Eight-Core CPUs; Intel® Xeon Phi™ 7120P ; Benchmark Source: Intel Corporation.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, refer to www.intel.com/performance/resources/benchmark_limitations.htm.

Optimization Notice: 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

Algorithm - Features

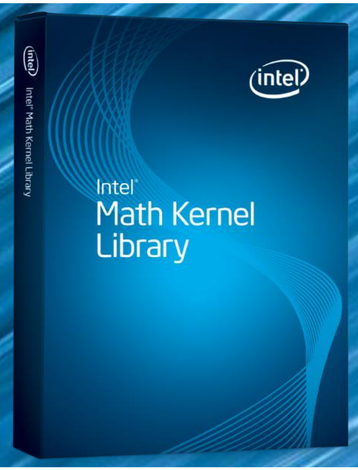
- Adaptive data/task distribution on the fly between CPUs and coprocessors to improve load balancing.
- Efficient utilization of all available computational units in heterogeneous systems.
- No limit on number of coprocessors on heterogenous systems.
- Scalability: A system with CPUs and 1 coprocessor shows >**3x** speed up.
  A system with CPUs and 2 coprocessors shows >**5x** speed up.

The algorithm provides a high degree of parallelism with minimal requirements for synchronization and communication

Algorithm - Implementation

- The algorithm is implemented within the framework of Intel® Math Kernel Library (Intel® MKL) LU, QR, and Cholesky factorization routines.
- The implementation detects the presence of Intel Xeon Phi coprocessors and automatically offloads the computations that can benefit from additional computational resources.
- The usage model hides the complexity of heterogeneous systems from the user, providing ease of use and providing the same API as usual Intel MKL routines.

Users do not need to change application code.



http://www.intel.com
http://software.intel.com/en-us/intel-mkl