NUMERICAL LINEAR ALGEBRA ON EMERGING ARCHITECTURES: THE PLASMA AND MAGMA PROJECTS

Emmanuel Agullo
Jim Demmel
Jack Dongarra
Bilel Hadri
Jakub Kurzak
Julien Langou
Hatem Ltaief
Piotr Luszczek
Rajib Nath
Stanimire Tomov
Vasily Volkov
Asim YarKhan

The emergence and continuing use of multi-core architectures and graphics processing units require changes in the existing software and sometimes even a redesign of the established algorithms in order to take advantage of now prevailing parallelism. Parallel Linear Algebra Software for Multi-core Architectures (PLASMA) and Matrix Algebra on GPU and Multi-core Architectures (MAGMA) are two projects that aim to achieve high performance and portability across a wide range of multi-core architectures and hybrid systems respectively.

INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE
http://icl.eecs.utk.edu/

A COLLABORATION OF
THE UNIVERSITY of TENNESSEE
Berkeley UNIVERSITY OF CALIFORNIA
University of Colorado Denver

SPONSORED BY
WITH SUPPORT FROM
The MathWorks
Microsoft
NVIDIA

PARALLEL LINEAR ALGEBRA SOFTWARE FOR MULTICORE ARCHITECTURES

PLASMA

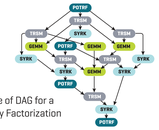
DOWNLOAD THE LIBRARY AT http://icl.eecs.utk.edu/plasma/

THE PARALLEL LINEAR ALGEBRA SOFTWARE FOR MULTICORE ARCHITECTURES (PLASMA) PROJECT aims to address the critical and highly disruptive situation that is facing the Linear Algebra and High Performance Computing community due to the introduction of multicore architectures. PLASMA's ultimate goal is to create software frameworks that enable programmers to simplify the process of developing applications that can achieve both high performance and portability across a range of new architectures. PLASMA uses a programming model that allows asynchronous, out-of-order scheduling of operations in order to achieve a scalable yet highly efficient software framework for Computational Linear Algebra applications.

TILE ALGORITHMS

Unlike LAPACK, which uses block algorithms, PLASMA relies on tile algorithms to enable the use of fine grained parallelism.

Tile algorithms of Linear Algebra operations can be represented as Directed Acyclic Graphs (DAG) where nodes represent the tasks in which the operation can be decomposed and the edges represent the dependencies among them. As long as the task execution order does not violate the dependencies, the result will be correct.



Example of DAG for a Cholesky Factorization

PLASMA 2.1.0

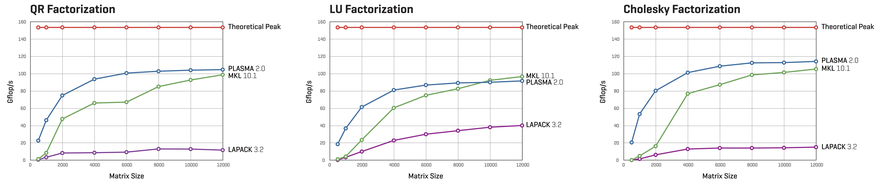
- Solution of Linear Equations
- Linear Least Squares Problems
- Multiple Precision Support
- Mixed-Precision Iterative Solver
- Static Scheduling
- LAPACK Interface / Native Interface
- LAPACK-Compliant Error Handling
- LAPACK-Derived Testing Suite
- Thread Safety
- Windows, Linux, AIX, Mac OS
- PLASMA Users' Guide

CURRENT RESEARCH

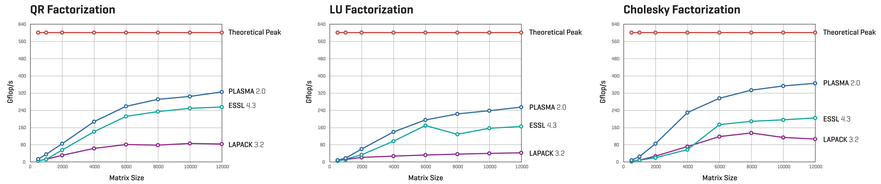
- Singular Value Decomposition
- Symmetric and Non Symmetric Eigenvalue Problems
- Dynamic Scheduling
- Communication Avoiding Algorithms
- Autotuning
- Distributed Memory Machines
- Hardware Accelerators

PERFORMANCE RESULTS DOUBLE PRECISION

CPU Intel Xeon 2.4 GHz Quad-socket Quad cores (16 cores total)



CPU IBM Power6 4.7 GHz 16 Dual cores (32 cores total)



MATRIX ALGEBRA FOR GPU AND MULTICORE ARCHITECTURES

MAGMA

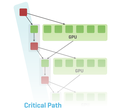
FIND OUT MORE AT http://icl.eecs.utk.edu/magma/

THE MATRIX ALGEBRA FOR GPU AND MULTICORE ARCHITECTURES (MAGMA) PROJECT aims to create a new generation of linear algebra libraries that achieve the fastest possible time to an accurate solution on hybrid/heterogeneous architectures, starting with current multicore+multiGPU systems. To address the complex challenges stemming from the heterogeneity of these systems, their massive parallelism, and the gap in computation vs. CPU-GPU communication speeds, MAGMA research is based on the idea that optimal software solutions will themselves have to hybridize, combining the strengths of different algorithms within a single framework.

HYBRID ALGORITHMS

MAGMA relies on hybrid algorithms that match algorithmic requirements to the architectural strengths of the system's hybrid components.

Small non-parallelizable tasks, often on the critical path, are scheduled on the CPU, and larger more parallelizable ones, often Level 3 BLAS, are scheduled on the GPU.



Hybrid Algorithms as DAGs

MAGMA 0.2

- Solution of Linear Equations
- Linear Least Squares Problems
- Multiple Precision Support
- Mixed-Precision, Iterative Solvers
- Hessenberg Reduction
- CPU / GPU Interfaces
- LAPACK-Compliant Accuracy
- Testing Suite and Examples
- Current system requirements: 1 CUDA-enabled GPU, Linux

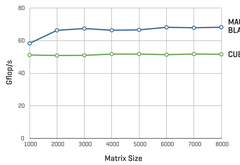
MAGMA BLAS

A complementary to CUBLAS subset of CUDA BLAS that are crucial for the performance of MAGMA routines.

FEATURING:

- Auto-tuned kernels
- Removed performance "dips" for "bad" matrix sizes
- GEMM tuned for rectangular matrices
- SYRK, GEMV, and SYMV
- TRSM of high parallelism/performance (trade-off for numerical stability)

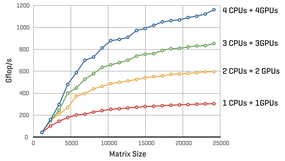
DGEMM tuned for rectangular matrices
(e.g.,Matrix Size Nx32)



CURRENT RESEARCH

- Symmetric and Non Symmetric Eigenvalue Problems
- Singular Value Decomposition
- Multicore + MultiGPU Algorithms

Cholesky Factorization SINGLE PRECISION

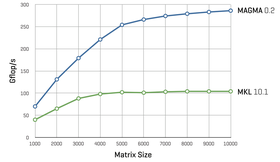


CPU AMD Opteron 1.8GHz, 4 cores **GPU** Tesla C1070 1.44GHz, 4 GPUs

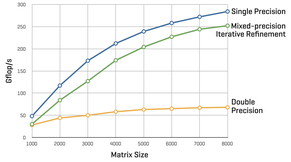
PERFORMANCE RESULTS

CPU Intel Xeon 2.33 GHz, 8 cores, s/d gemm peak 128/65 GFlop/s **GPU** NVIDIA GTX280 1.33 GHz, s/d gemm peak 375/75 GFlop/s

QR Factorization SINGLE PRECISION



Solving A x = b using LU factorization



Reduction to Hessenberg Form for Eigenvalue computations SINGLE PRECISION