PLASMA

http://icl.cs.utk.edu/plasma/

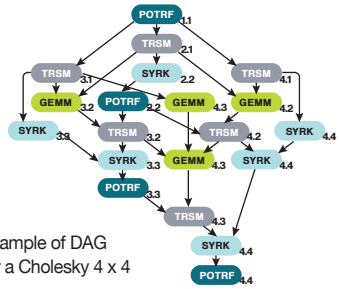
PARALLEL LINEAR ALGEBRA FOR SCALABLE MULTI-CORE ARCHITECTURES

The Parallel Linear Algebra for Scalable Multi-core Architectures (PLASMA) project aims to address the critical and highly disruptive situation that is facing the Linear Algebra and High Performance Computing communities due to the introduction of multi-core 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. The development of programming models that enforce asynchronous, out of order scheduling of operations is the concept used as the basis for the definition of a scalable yet highly efficient software framework for Computational Linear Algebra applications.

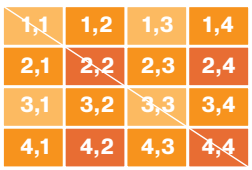
SOFTWARE/ALGORITHMS FOLLOW HARDWARE EVOLUTION IN TIME

LINPACK (70's) **(Vector operations)**		Rely on - Level-1 BLAS operations
LAPACK (80's) **(Blocking, cache friendly)**		Rely on - Level-3 BLAS operations
ScaLAPACK (90's) **(Distributed Memory)**		Rely on - PBLAS Message Passing
PLASMA (00's) **New Algorithms** **(many-core friendly)**		Rely on - a DAG-based scheduler - a block data layout - some extra kernels

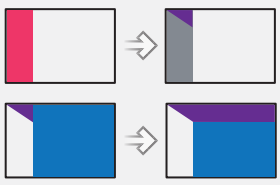
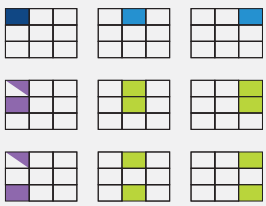
Tile algorithms of Linear Algebra operations can be represented as Directed Acyclic Graphs (DAG) where nodes represent the tasks in which the operation can be decomposed and the edges represent the dependencies among them. As long as the task execution order does not violate the dependencies, the result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



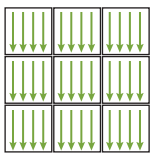
Example of DAG for a Cholesky 4 x 4



Contrary to block algorithms like in LAPACK, PLASMA relies on tile algorithms to allow fine granularity parallelism.

Block algorithms - LAPACK	Tile algorithms - PLASMA
	

BLOCK DATA LAYOUT



Novel data formats like Block Data Layout (BDL) improve locality of reference to memory and higher reuse of data in memories that are closer to cores (caches or scratchpad memories).

University of California Berkeley
University of Colorado Denver Downtown Campus
The University of Tennessee

SPONSORED BY The MathWorks Microsoft NSF

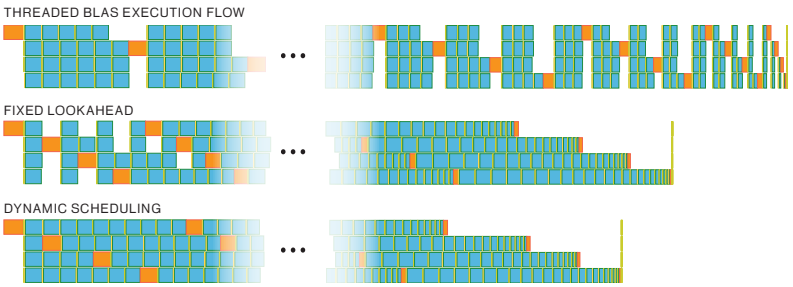
PLASMA

http://icl.cs.utk.edu/plasma/

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. The development of programming models that enforce asynchronous out of order scheduling of operations is the concept used as the basis towards the definition of a scalable yet highly efficient software framework for Computational Linear Algebra applications.

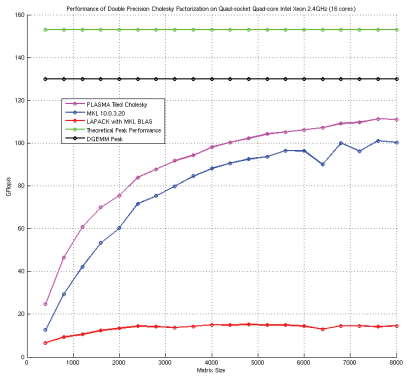
It is difficult to overestimate the magnitude of the discontinuity that the high performance computing (HPC) community is about to experience because of the emergence of next generation of multi-core and heterogeneous processor designs. For at least two decades, HPC programmers have taken for granted that each successive generation of microprocessors would, either immediately or after minor adjustments, make their old software run substantially faster. But three main factors are converging to bring this "free ride" to an end. First, system builders have encountered intractable physical barriers - too much heat, too much power consumption, and too much leaking voltage - to further increases in clock speeds. Second, physical limits on the number and bandwidth of pins on a single chip mean that the gap between processor performance and memory performance, which was already bad, will get increasingly worse. Finally, the design trade-offs being made to address the previous two factors will render commodity processors, absent any further augmentation, inadequate for the purposes of tera- and petascale systems for advanced applications. This daunting combination of obstacles has forced the designers of new multi-core and hybrid systems, searching for more computing power, to explore architectures that software built on the old model are unable to effectively exploit without radical modification. Currently available Linear Algebra software packages rely on parallel implementations of the Basic Linear Algebra Subroutines (BLAS) to take advantage of multiple execution units. This solution is characterized by a fork-join model of parallel execution, which may result in suboptimal performance on current and future generations of multi-core processors since it introduces strict dependencies due to the presence of non parallelizable portions of code. The Parallel Linear Algebra for Scalable Multicore Architectures (PLASMA) project aims to overcome the shortcomings of this approach introducing a pipelined model of parallel execution.

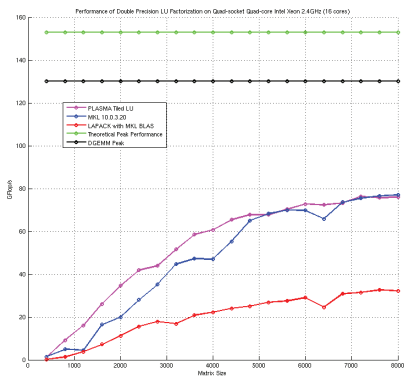
The dynamic asynchronous scheduling of the tasks allows the overlapping of slow serial tasks with more efficient ones and removes all the synchronizations introduced by the fork-join execution model.

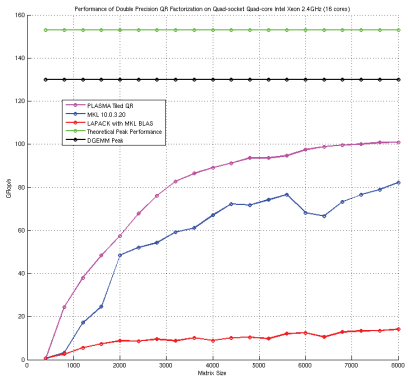


RESULTS

Performance results on Intel Xeon 2.4 GHz Quad-socket Quad cores (16 cores total) for the LU, Cholesky and QR factorizations.







The graphs report the comparison of the LAPACK block algorithm with multithreaded MKL BLAS, the PLASMA implementations and implementations from vendor libraries.

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

COMING SOON Two-Sided Matrix Bidiagonalization for Multicore Architectures