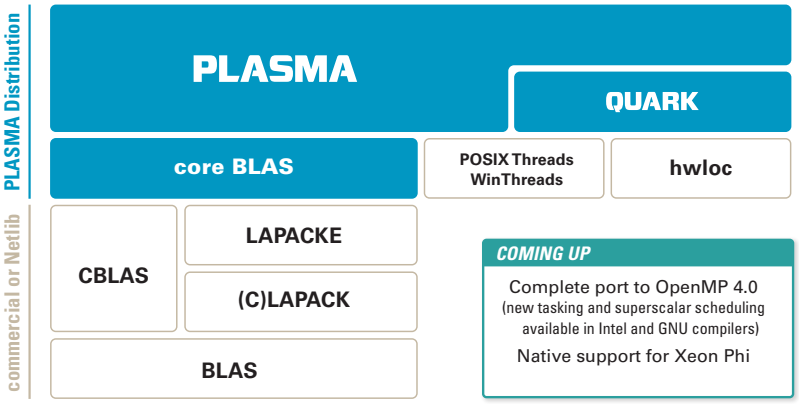
PLASMA

INNOVATIVE COMPUTING LABORATORY

PLASMA (Parallel Linear Algebra Software for Multicore Archtectures) is a dense linear algebra package at the forefront of multicore computing. PLASMA is designed to deliver the highest possible performance from a system with multiple sockets of multicore processors. PLASMA achieves this objective by combining state of the art solutions in parallel algorithms, scheduling, and software engineering. PLASMA currently offers a collection of routines for solving linear systems of equations, least square problems, eigenvalue problems, and singular value problems.

MODERN SOFTWARE STACK



HIGHLIGHTS

State-of-the-art LU factorization with multithreaded and cache-efficient panel factorization

State-of-the-art hierarchical QR factorization for tall-and-skinny matrices

State-of-the-artSVD and symmetric EVP solvers based on three-stage factorization and multithreaded Divide and Conquer algorithm

State-of-the-art, parallel and cache efficient, matrix layout translation and transposition routines

STATE-OF-THE-ART SOLUTIONS

Tile Matrix Layout

PLASMA lays out matrices in square tiles of relatively small size, such that each tile occupies a continuous memory region. Tiles are loaded to the cache memory efficiently with little risk of eviction while being processed. The use of tile layout minimizes conflict cache misses, TLB misses, and false sharing, and maximizes the potential for prefetching. PLASMA contains parallel and cache efficient routines for converting between the conventional LAPACK layout and the tile layout.

Tile Algorithms

PLASMA introduces new algorithms redesigned to work on tiles, which maximize data reuse in the cache levels of multicore systems. Tiles are loaded to the cache and processed completely before being evicted back to the main memory. Operations on small tiles create fine grained parallelism providing enough work to keep a large number of cores busy.

Dynamic Scheduling

PLASMA relies on runtime scheduling of parallel tasks. Runtime scheduling is based on the idea of assigning work to cores based on the availability of data for processing at any given point in time, and thus is also referred to as data-driven scheduling. The concept is related closely to the idea of expressing computation through a task graph, often referred to as the DAG (Directed Acyclic Graph), and the flexibility of exploring the DAG at runtime. This is in direct opposition to the fork-and-join scheduling, where artificial synchronization points expose serial sections of the code and multiple cores are idle while sequential processing takes place.

SUPPORT FROM THE SILAS NSF AWARD

ICL INNOVATIVE COMPUTING LABORATORY

IN COLLABORATION WITH Berkeley UNIVERSITY OF CALIFORNIA, University of Colorado Denver

WITH SUPPORT FROM FUJITSU, MathWorks, Microsoft

SPONSORED BY U.S. DEPARTMENT OF ENERGY, NSF