PLASMA (Parallel Linear Algebra Software for Multicore Architectures) 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. Currently PLASMA offers a collection of routines for solving linear systems of equations, least square problems, eigenvalue problems and singular value problems.

**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.
FUNCTIONALITY

**Linear Systems of Equations**
- Cholesky, LU (partial pivoting), LDL (RBT), LDL (Aasen’s pivoting)

**Condition Number Estimation**
- GE, PO, SY, TR

**Explicit Matrix Inversion**
- Cholesky, LU (partial pivoting)

**Least Squares**
- QR & LQ, QR with column pivoting

**Mixed Precision Iterative Refinement**
- Cholesky, LU (linear systems); OR, LQ (least squares)

**Symmetric Eigenvalue Problem**
- eigenvalues, all eigenvectors, subset of eigenvectors

**Generalized Symm Eigen Problem**
- eigenvalues, all eigenvectors, subset of eigenvectors

**Nonsymmetric Eigenvalue Problem**
- singular values, all singular vectors, subset of singular vectors

**Level 3 Tile BLAS**
- GEMM, HEMM, HER2K, HERK, SYMM, SYR2K, SYRK, TRMM, TRSM

**In-Place Layout Translation**
- CM, RM, CCRB, CRBB, RCRB, RRBB

**FEATURES**

- Cholesky, LU (partial pivoting), LDL (RBT), LDL (Aasen’s pivoting)
- GE, PO, SY, TR
- Cholesky, LU (partial pivoting)
- QR & LQ, QR with column pivoting
- Cholesky, LU (linear systems); OR, LQ (least squares)
- eigenvalues, all eigenvectors, subset of eigenvectors
- eigenvalues, all eigenvectors, subset of eigenvectors
- singular values, all singular vectors, subset of singular vectors
- GEMM, HEMM, HER2K, HERK, SYMM, SYR2K, SYRK, TRMM, TRSM
- CM, RM, CCRB, CRBB, RCRB, RRBB

**PERFORMANCE RESULTS**

**Solving Linear System (DGE SV)**
48-core, 2.1 GHz AMD Magny-Cours System

**Solving Symmetric EVP (DSYEV)**
48-core, 2.1 GHz AMD Magny-Cours System

**Solving Least Squares Problem (DGELS)**
48-core, 2.1 GHz AMD Magny-Cours System

**Solving Singular Value Problem (DGESVD)**
48-core, 2.1 GHz AMD Magny-Cours System

**FEAT URES**

- Covering four precisions: double real, double complex, single real, single complex (D, Z, S, C)
- Providing LAPACK-compatible interface for matrices in F77 column-major layout
- Supporting:
  - Linux, MS Windows, Mac OS, AIX