DPLASMA

EFFICIENT DENSE LINEAR ALGEBRA ON DISTRIBUTED HYBRID MANYCORE SYSTEMS

Distributed Parallel Linear Algebra Software for Multicore Architectures (DPLASMA) is the leading implementation of a dense linear algebra package for distributed heterogeneous systems. Unlike any predecessor, DPLASMA depicts algorithms using data flow principles as pure data dependencies between BLAS kernels. The resulting dataflow depiction takes advantage of the state-of-the-art distributed runtime, PaRSEC, to achieve portable and sustained performance never seen before on heterogeneous distributed systems.

User Defined Data Placement
In addition to traditional ScaLAPACK (block-cyclic) data distribution, DPLASMA provides interfaces to define arbitrary data collections with unrestrained distributions. The DPLASMA data flow algorithms transparently operate on local data, or introduce implicit communications to resolve dependencies, thereby removing the burden of initial data re-shuffle and providing the user a novel approach to address load balance.

FUNCTIONALITY

<table>
<thead>
<tr>
<th>Coverage</th>
<th>Linear Systems of Equations</th>
<th>Cholesky, LU (inc. pivoting, PP), and LDL (prototype)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Least Squares</td>
<td>QR and LQ</td>
<td>Reduction to Band (prototype)</td>
</tr>
<tr>
<td>Symmetric Eigenvalue Problem</td>
<td>GEMM, TRSM, TRMM, HEMM / SYMM, HERK / SYRK, and HER2K / SYR2K</td>
<td></td>
</tr>
<tr>
<td>Level 3 Tile BLAS</td>
<td>Matrix generation (PLRNT, PLGHE / PLGSY, PLTMG), Norm computation (LANGE, LANGE / LANHE / LANSY, LANTR), Extra functions (LA taxi, LACPY, LASCAL, GEADD, TRADD, PRINT), and Generic Map functions</td>
<td></td>
</tr>
<tr>
<td>Auxiliary Subroutines</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Features
Recursive DAG Instantiation, allowing heterogeneous tile size executions to tune for heterogeneous devices
Covering four precisions: double real, double complex, single real, and single complex (D, Z, S, C)
Providing ScaLAPACK-compatible interface for matrices in F77 column-major layout
Supporting: Linux, Windows, macOS, UN*X (depends on MPI, hwloc)
Fine-grain Composition of Operations

Future Plans
Two-sided Factorizations
Distributed Sparse Solver
More GPU kernels integration
LU+RBT
BLR Solver
Eigenvalue Decomposition
and Singular Value Decomposition

Performance Results

PROBLEM AND NODE SCALING OF A MATRIX MULTIPLY (DGEMM)
Summit: 2–72 nodes (40 cores each with 6 V100s) with a 1024 × 1024 tile size

PROBLEM SCALING OF A CHOLESKY FACTORIZATION (DPOTRF)
Shaheen II: 512 nodes (32 cores each) with a 400 × 400 tile size.

ENERGY CONSUMPTION SOLVING A LINEAR LEAST SQUARE PROBLEM (DGEOQR)

IN COLLABORATION WITH

INNOVATIVE COMPUTING LABORATORY

https://github.com/iclsdisco/dplasma

SPONSORED BY

U.S. DEPARTMENT OF ENERGY
Office of Science

National Science Foundation
ICL’s Parallel Runtime and Execution Controller (PaRSEC) project is a generic framework for architecture-aware scheduling and management of microtasks on distributed, many-core, heterogeneous architectures. The PaRSEC environment also provides a runtime component for dynamically executing tasks on heterogeneous distributed systems along with a productivity toolbox and development framework that supports multiple domain-specific languages (DSLs) and extensions and tools for debugging, trace collection, and analysis.

**A Generic Runtime for Domain-specific Language/Extensions**

The PaRSEC engine enables the domain scientist to implement a DSL to efficiently interact with the runtime, thereby improving productivity and portability.

With PaRSEC, applications are expressed as a direct acyclic graph (DAG) of tasks with edges designating data dependencies. This DAG dataflow paradigm attacks both sides of the exascale challenge: managing extreme-scale parallelism and maintaining the performance portability of the code.

The ECP Distributed Tasking at Exascale (DTE) effort is a vital extension that ensures that PaRSEC meets the critical needs of ECP application communities in terms of scalability, interoperability, and productivity.

**Accelerate your Application with PaRSEC**

Write once, execute on any hardware: adding distributed memory and GPU acceleration to a PaRSEC code is simple, and performance portable, thanks to implicit data movement.

With PaRSEC, your main code in C, Template C++, Fortran, Python, etc., your PaRSEC application is modular, and you can accelerate critical routines only, and use Open MP, Kokkos, Cuda etc. as the main body for your tasks. The PaRSEC ecosystems comes with tools for debugging, performance analysis as well as documentation.

Installing PaRSEC on leadership class hardware and workstations alike is simple with CMake, Spack, PkgConfig integrations.

**Performance Results**

- **Gordon Bell Finalist Run:**
  - Exascale Tile Low-Rank: Strong correlation on 4096 and 48384 Fugaku nodes

- **Tile, Low-Rank, Cholesky Factorization for Large Matrices**
  - Shaheen II: 4096 nodes
  - (32 cores each @ 2.30 GHz [Intel Haswell])

- **Performance Portability on AMD ROCM Hardware:**
  - Early results for Cholesky Factorization on pre-Frontier systems

---

**SPONSORED BY**

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

[https://github.com/icldisco/parsec](https://github.com/icldisco/parsec)