# 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 / COVERAGE
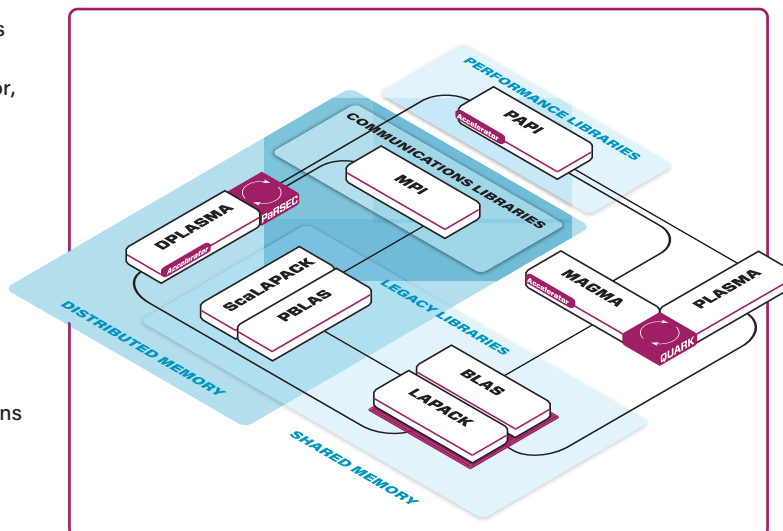
FUNCTIONALITY	COVERAGE
Linear Systems of Equations	Cholesky, LU (inc. pivoting, PP), and LDL (prototype)
Least Squares	QR and LQ
Symmetric Eigenvalue Problem	Reduction to Band (prototype)
Level 3 Tile BLAS	GEMM, TRSM, TRMM, HEMM / SYMM, HERK / SYRK, and HER2K / SYR2K
Auxiliary Subroutines	Matrix generation (PLRNT, PLGHE / PLGSY, PLTMG), Norm computation (LANGE, LANHE / LANSY, LANTR), Extra functions (LASET, LACPY, LASCAL, GEAD, TRADD, PRINT), and Generic Map functions

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)
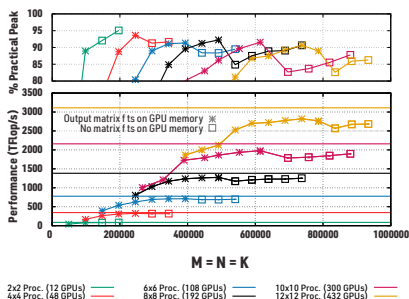
Fine-grain Composition of Operations

Future Plans

Two-sided Factorizations

Distributed Sparse Solver

More GPU kernels integration

LU+RBT

BLR Solver

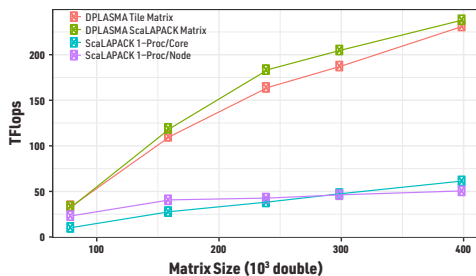Eigenvalue Decomposition and Singular Value Decomposition

Performance Results

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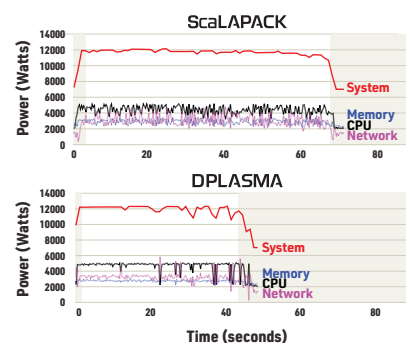Summit: 2–72 nodes (40 cores each with 6 V100s) with a 1024 × 1024 tile size

M = N = K

2x2 Proc. (12 GPUs)
4x4 Proc. (48 GPUs)
6x6 Proc. (108 GPUs)
8x8 Proc. (192 GPUs)
10x10 Proc. (300 GPUs)
12x12 Proc. (432 GPUs)

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

DPLASMA Tile Matrix
DPLASMA ScaLAPACK Matrix
ScaLAPACK 1-Proc/Core
ScaLAPACK 1-Proc/Node

Matrix Size (10³ double)

ENERGY CONSUMPTION SOLVING A LINEAR LEAST SQUARE PROBLEM (DGEQRF)

Time (seconds)

IN COLLABORATION WITH

Inria · KAUST · University of Colorado Denver

ICL INNOVATIVE COMPUTING LABORATORY

https://github.com/icldisco/dplasma
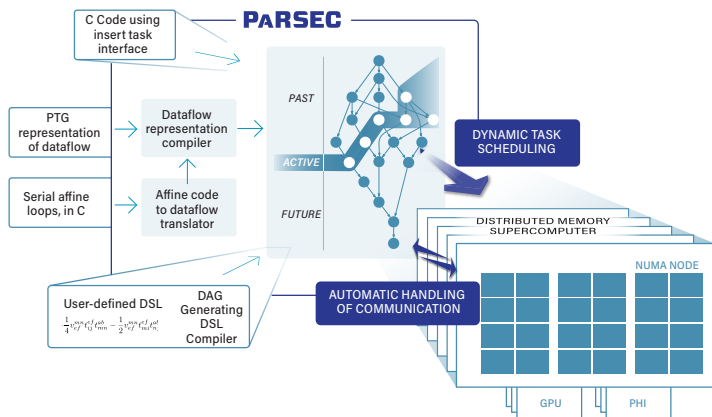
PaRSEC

PARALLEL RUNTIME AND EXECUTION CONTROLLER

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.

Write your main code in C, Templated 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.

DOMAIN SPECIFIC LANGUAGES (DSLs)

Dynamic Task Discovery (DTD)

DTDs enable a sequential description of application data and tasks dependencies similar to OpenMP. Tasks are presented using an insert_task directive, with an option to declare typed dependencies (e.g., read, write, and atomic update), including on hybrid distributed environments.
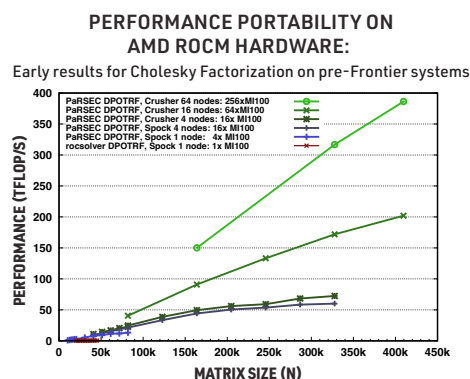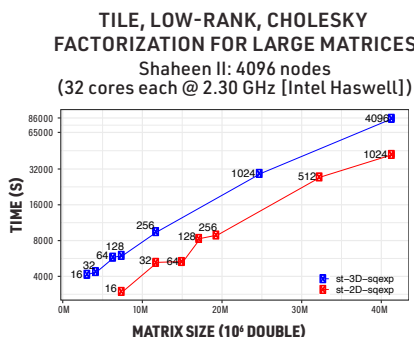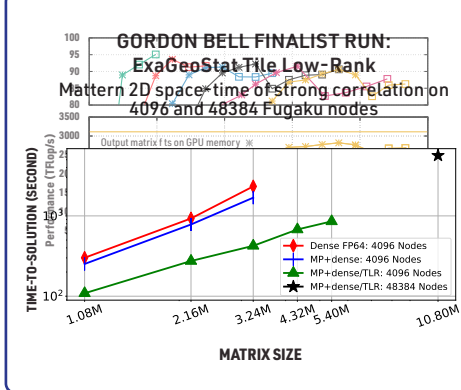
Templated Task Graph (TTG)

DTE includes a set of C++ Template classes to express dynamic DAGs for heterogeneous datasets. At the heart of TTG lie the Operand class (which represents Tasks) and the Terminal class (which connects Operands together). In the Operand body, the programmer explicitly transmits data to output terminals to trigger the input terminals of destination tasks. The language is heavily templated, moving all compiler-decidable decisions at compile time and uses the Standard Template Library to encapsulate communications between Operands.

Parameterized Task Graph (PTG)

A PTG is a concise, symbolic, problem size-independent task graph representation, with implicit data movements that supports hybrid architectures via multiple task incarnation. In PTG, the developer expresses all flows of data between tasks in an analytical way using the tasks parameters. This representation is then used by PaRSEC to track dependencies and schedule tasks and data movement.

PERFORMANCE RESULTS

GORDON BELL FINALIST RUN:
ExaGeoStat Tile Low-Rank
Mattern 2D space-time of 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

ICL INNOVATIVE COMPUTING LABORATORY

https://github.com/icldisco/parsec