The Distributed Tasking for Exascale (DTE) project extends the capabilities of ICL’s Parallel Runtime and Execution Controller (PaRSEC) project—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.

**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 the option to declare typed dependencies (e.g., read, write, and atomic update), including on hybrid distributed environments.

**C++ / SLATE**

DTE also features an extension for DTD in C++ that maps the Software for Linear Algebra Targeting Exascale (SLATE) project’s multi-level algorithms to multi-level DAGs. Sets of embarrassingly parallel tasks are gathered in containers, and the dependencies are expressed between these containers at the higher level. Explicit communication happens inside the progress of these containers and in between.

**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-decides 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**

**Problem and Node Scaling of a Matrix Multiply (DGEMM)**

Summit: 2-72 nodes (40 cores each with 6 V100s) with a 1024 x 1024 tile size

**Tile, Low-Rank, Cholesky Factorization for Large Matrices**

Shaheen II: 4096 nodes (32 cores each @ 2.30 GHz [Intel Haswell])

**Energy Consumption Solving a Linear Least Square Problem (DGEQRF)**

---

**Sponsored By**

U.S. Department of Energy, Office of Science, ECP, NIS

Find out more at [https://icl.utk.edu/dte](https://icl.utk.edu/dte)

---

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.
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**
- 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, UNIX (depends on MPI, hloc)
- Fine-grain Composition of Operations

**Future Plans**
- Two-sided Factorizations
- Distributed Sparse Solver
- More GPU Sparse Solver
- 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 x 1024 tile size

- **Problem Scaling of a Cholesky Factorization (DPOTRF)**
  - Shaheen II: 912 nodes (32 cores each) with a 400 x 400 tile size

- **Energy Consumption Solving a Linear Least Square Problem (DGEQRF)**
  - ScALAPACK
  - DPLASMA

---

**In Collaboration With**
- University of Colorado Denver
- INRIA
- KAUST

**With Support From**
- Microsoft

**Find Out More At**
- https://icl.utk.edu/dplasma

**SPONSORED BY**
- U.S. Department of Energy Office of Science
- ECP
- NIFA
- National Science Foundation