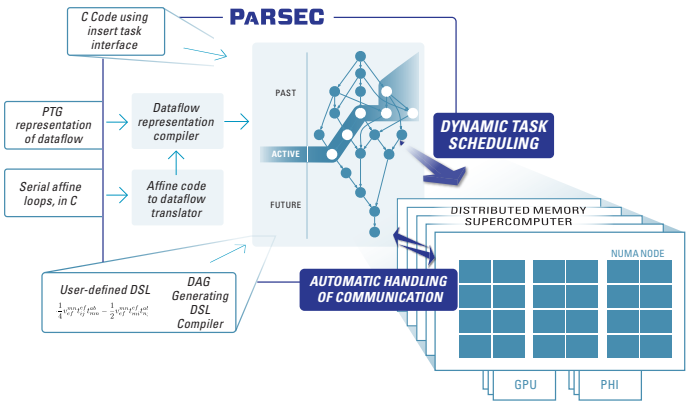
DTE

DISTRIBUTED TASKING FOR EXASCALE

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 and extensions and tools for debugging, trace collection, and analysis.

PARSEC A GENERIC RUNTIME FOR DOMAIN SPECIFIC LANGUAGE / EXTENSIONS



The PaRSEC engine enables the domain scientist to implement a Domain Specific Language (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 DTE effort is a vital extension and continuation of this effort and will ensure that PaRSEC meets the critical needs of ECP application communities in terms of scalability, interoperability, and productivity.

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, 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 decidable decisions at compile time and uses the Standard Template Library to encapsulate communications between Operands.

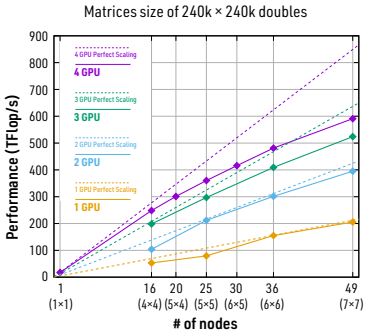
PARAMETERIZED TASK GRAPH (PTG)

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

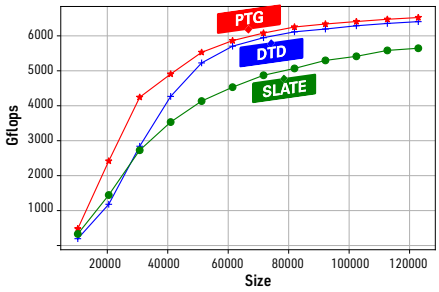
Strong Scaling of a Matrix Multiply (DGEMM)

Summit-dev, 1 to 49 nodes, 20 cores and 4 P100 per node



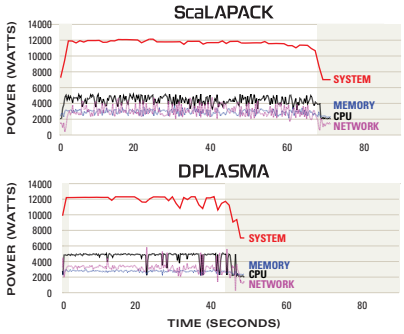
Problem Scaling of a Cholesky Factorization (DPOTRF)

64 nodes, Intel Xeon X5660@2.8GHz, 12 core / node



Solving Linear Least Square Problem (DGEQRF)

System G Virginia Tech, 32-node, 256-core, Intel Xeon 2.8GHz, IB20G



SPONSORED BY

U.S. DEPARTMENT OF ENERGY | Office of Science

ECP

NNSA National Nuclear Security Administration

FIND OUT MORE AT **http://icl.utk.edu/dte**

CITR CENTER FOR INFORMATION TECHNOLOGY RESEARCH

THE UNIVERSITY OF TENNESSEE KNOXVILLE

INNOVATIVE COMPUTING LABORATORY