CSSI: EPEXA
Ecosystem for Programming and Executing eXtreme-scale Applications

Robert. J. Harrison
Mohammad Mahdi Javanmard
IACS, Stony Brook University
https://iacs.stonybrook.edu/
Award #1931387

George Bosilca
Thomas Herault
ICL, University of Tennessee
http://www.icl.utk.edu/
Award #1931384

Edward F. Valeev
Department of Chemistry, Virginia Tech
https://bit.ly/3aZmM4G
Award #1931347

rharrison@stonybrook.edu

http://tesseorg.github.io/

Introduction to EPEXA

EPEXA is creating a production-quality, general-purpose, community-supported, open-source software ecosystem to attack the twin challenges of programmer productivity and portable performance for advanced scientific applications. Through application-driven codesign we focus on the needs of irregular and sparse applications that are poorly served by current programming and execution models on massively-parallel, hybrid systems.

Challenges from Modern Computational Science

- Advances in predictive, high-fidelity simulation characterized by increasingly irregular and dynamic computation (block sparse, low-rank, mixed representations, etc.).
- Ongoing technology trend in heterogeneous architectures with dynamically changing performance, and need to increase concurrency at all scales.

Specific Aims

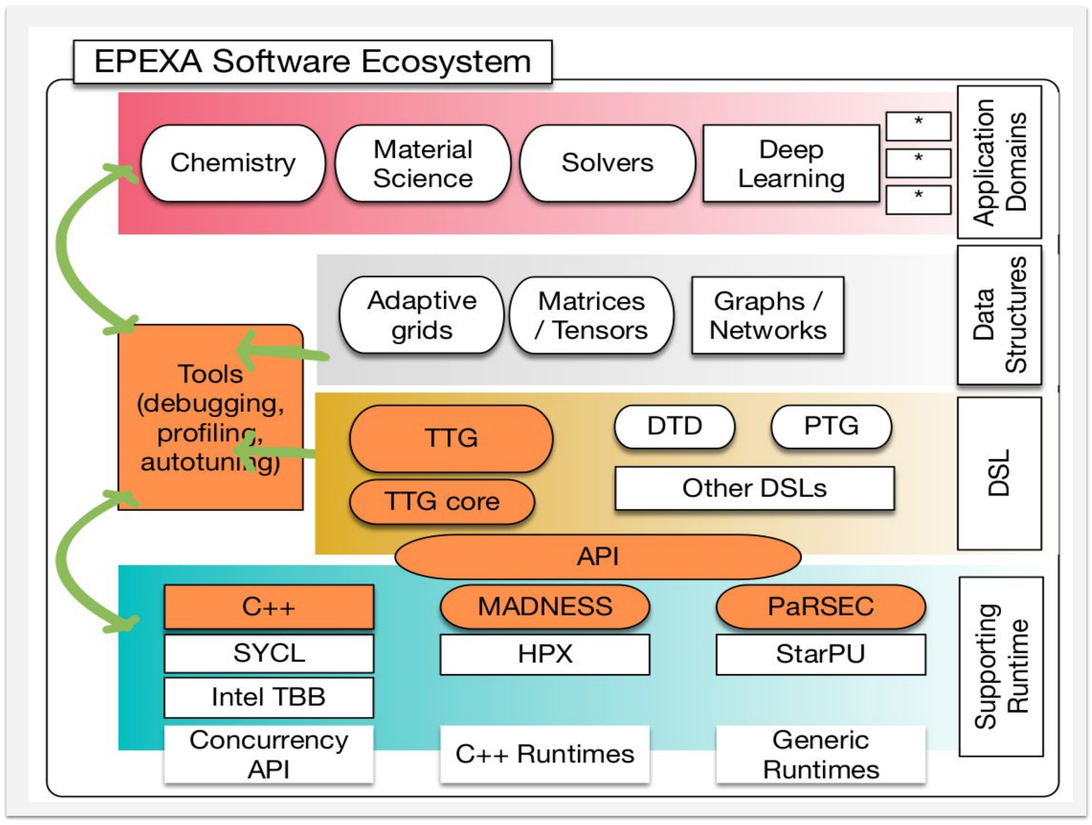
- To extend, complete, and harden the successful TESSE research prototype providing its first production quality implementation using a community-based, science-driven approach.
- To grow and support the user community, associated applications, and research use cases.
- To create a community to design, maintain, support and to grow EPEXA in the future.
- To transform the scalability of key parts of new and existing numerical simulation codes, including enabling the development of new DSLs by utilizing the API of the EPEXA runtime, and by migration paths for both applications and application programmers to follow.

EPEXA Community and Driving Applications

- Robert Harrison (PI) - multiresolution numerical analysis (MADNESS; https://github.com/m-a-d-n-e-s-s/madness)
- Mahdi Javanmard - dynamic programming
- Edward Valeev (co-PI) - block-sparse tensor algebra (TiledArray; https://github.com/ValeevGroup/tiledarray)
- George Bosilca and Thomas Herault (co-PIs) - parallel programming and runtime (PaRSEC; http://icl.utk.edu/parsec/)
- Scott Thornton - geospatial AI (Descartes Labs; https://www.descarteslabs.com/)
- Florian Bischoff - chemistry (Humboldt University; https://bit.ly/2RZG2q0)
- Wolfgang Bangerth - deal.II open-source finite-element engineering and physics (Colorado State University; https://www.dealii.org)

What is TESSE?

- An extensible, robust and scalable directed acyclic graph (DAG) execution model supported by an intelligent and dynamic runtime that can adapt to changing requirements presented by the evolving numerical theories and HPC platforms.
- TTG (template task graph - see RHS panel) is the main initial C++ API.
- Multiple runtimes are supported (MADNESS and PaRSEC currently, plans for UCX, HPX, native C++, and cloud stacks)
- Plans for multiple interoperable DSLs and algorithms on distributed data

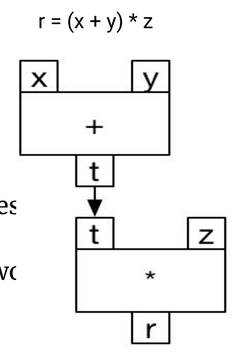


Templated Task Graph (TTG)

The Templated Task Graph API / DSL has been developed to enable a straightforward expression of the parallelism for algorithms that work on irregular and unbalanced data sets. Combining our experience with MADNESS, TiledArray and PaRSEC, the DSL employs C++ templates to build an internal representation of the Distributed DAG of Tasks.

Overall Objective

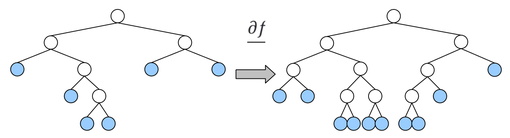
- Provide an intermediate-level expression of data-dependent irregular algorithms while leveraging a powerful micro-task runtime to manage dependencies, scheduling, and data motion within the data flow.
- Encourage programs that avoid non-essential barriers and intermediates express available parallelism without drowning the developer in detail, and reap most benefits of fusion within a more general framew



Key Concepts

1) Parameterize each task that will execute the operation by a key or index (e.g. a loop index making a separate task for each iteration; the label of each node in a tree being traversed; a pair of indices labelling a matrix sub-block).
2) Avoid describing / observing the entire task graph at once (avoid memory clogging)
3) Data labeled by a key to match with consuming task (all inputs of a task must have the same key, while outputs may target different keys)
4) Through each output, a task can send data to a specific successor (identified by its key), or broadcast to multiple successors (keys).

Example: Differentiation

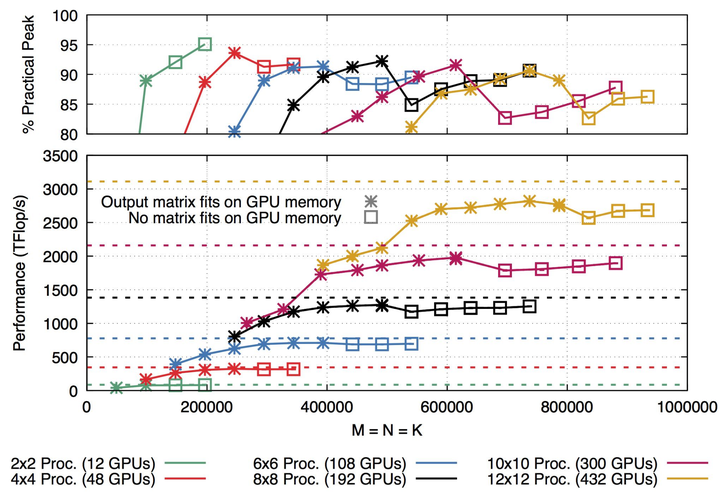



```
void diff(const Key& key, const Node& left, const Node& center,
          const Node& right,
          tuple<nodeOut,nodeOut,nodeOut,nodeOut>& out)
{
    nodeOut &L=get<0>(out), &C=get<1>(out),
            &R=get<2>(out), &result=get<3>(out);
    if (!(left.has_children || center.has_children ||
          right.has_children)) {
        double derivative = (right.s - left.s)/(4.0*::L*pow2(-key.n));
        result.send(key,Node(key,derivative,0.0,false));
    }
    else {
        result.send(key,Node(key,0.0,0.0,true));
        if (!left.has_children) L.send(key.left_child(), left);
        if (!center.has_children) {
            auto children = {key.left_child(),key.right_child()};
            L.send(key.right_child(),center);
            C.broadcast(children,center);
            R.send(key.left_child(), center);
        }
        if (!right.has_children) R.send(key.right_child(),right);
    }
}
```

Dense Linear Algebra over multi-GPUs/node using PaRSEC

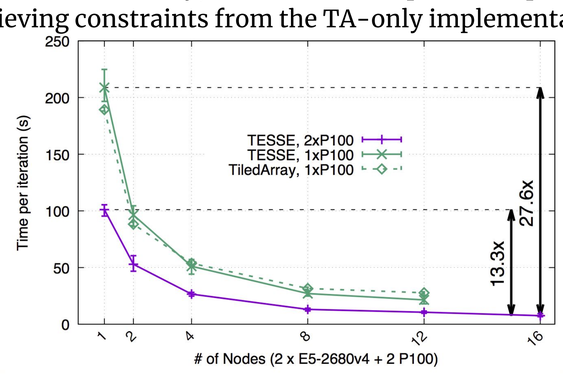
Extracting high performance out of hybrid systems like the ones developed for exascale computing can be challenging because of the complexity of the platforms. For example, Summit, the ORNL pre-exascale system from the DOE features 6 Pascal P-100 accelerators per node, each with a limited memory (16GB). 97% of the computing capability is located on the GPUs, but data movement must be orchestrated to remain within the bounds of memory available on the node and within the bounds of memory available on each GPU.

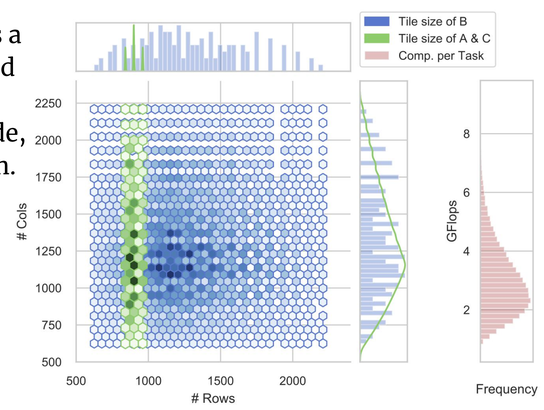
Using PaRSEC, we were able to provide a scalable approach that achieves more than 85% of the peak performance, and that sustains this performance even when the problem size outgrows the memory capacity of the accelerators. In that case, the runtime system moves the data transparently and in the background, while computations continue on the GPUs.



Irregularly-tiled ABCD term using TiledArray, PaRSEC and MADNESS (a crucial step towards block-sparse computation)

In the context of TiledArray, each tile of the GEMM has a different size, generating large variation in memory and load. Integrating TA, PaRSEC and MADNESS, enabled excellent scalability, and use of multiple-GPUs per node, relieving constraints from the TA-only implementation.





Distribution of tile sizes and amount of computations per task for the ABCD term in the coupled-cluster doubles equation for $(H_2O)_{12}$ in aug-cc-pVDZ basis set.

ICL INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY OF TENNESSEE KNOXVILLE
iACS INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE
Stony Brook University
VIRGINIA TECH
NSF