Using Coroutines to Support Accelerators in TTG

–

Discussion on programming models risks and benefits

Joseph Schuchart

HiHAT Monthly Review

July 16, 2024

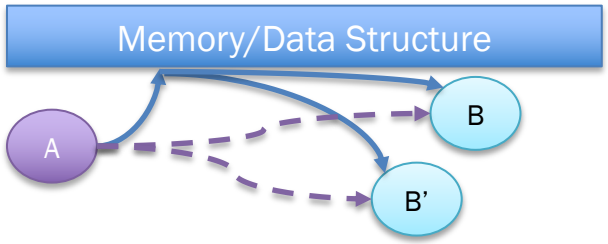
ICL INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Composition Granularity

Backward-looking models

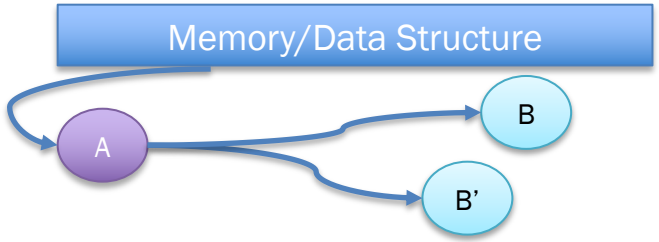
- Based on integration points:
  - Data structures (StarPU, Legion)
  - Memory locations (OpenMP, OmpSs)
  - Futures/promises (HPX, C++)
- Task-wise composition:
  - Distributed data structures
  - Based on (*implicit*) contracts



Task B accesses some data and thus depends on Task A, who produced that data.

Forward-looking models

- Pure data-flow
- Task-graphs similar to functions:
  - Black-boxes with well-defined inputs & outputs
  - Computational structure known within
- Algorithms composed through Edges as integration points



Task A produces data and sends it to Task B, who consumes it.

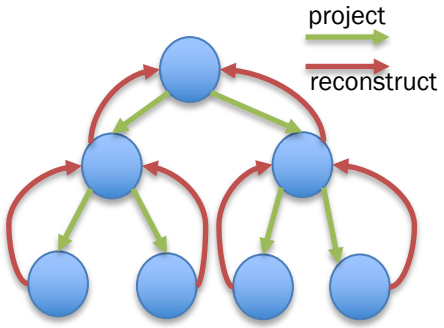
Target Applications

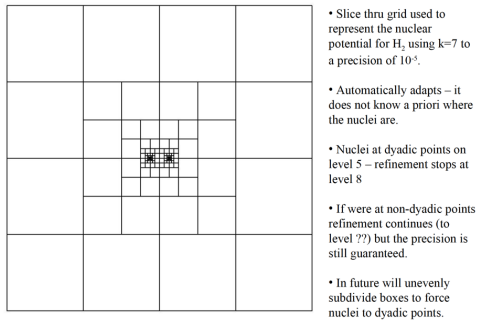
Block-sparse matrix algorithms for Sparse Tensor Algorithms in TiledArray

- Forwarding of tiles between operators
- Forking off tiles to preserve versions from subsequent changes
- Composition of operators
  - Merging basic operators in TA
  - Flow between operators

Multi-Resolution Function Analysis in MADNESS

- Tree traversal: refinement and reconstruction of functions in 3D/6D
- Central part of MADNESS
- Variable tree-depths, dynamic refinement
- No distributed data structure





- Slice thru grid used to represent the nuclear potential for H_2 using k=7 to a precision of 10^{-5}.
- Automatically adapts – it does not know a priori where the nuclei are.
- Nuclei at dyadic points on level 5 – refinement stops at level 8
- If were at non-dyadic points refinement continues (to level ??) but the precision is still guaranteed.
- In future will unevenly subdivide boxes to force nuclei to dyadic points.

Acknowledgements

This research was supported partly by NSF awards #1931347 and #1931384, and 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. We gratefully acknowledge the provision of computational resources by the Oak Ridge National Laboratory (ORNL) and the High-Performance Computing Center (HLRS) at the University of Stuttgart, Germany.

Resources

- ECP Tutorial: https://www.exascaleproject.org/event/ttg-2022/
- Paper:
  - J. Schuchart *et al.*, "Generalized Flow-Graph Programming Using Template Task-Graphs: Initial Implementation and Assessment," *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS).*
  - J. Schuchart, P. Nookala, T. Herault, E. F. Valeev and G. Bosilca, "Pushing the Boundaries of Small Tasks: Scalable Low-Overhead Data-Flow Programming in TTG," *2022 IEEE International Conference on Cluster Computing (CLUSTER).*
  - T. Herault, J. Schuchart, E. F. Valeev and G. Bosilca, "Composition of Algorithmic Building Blocks in Template Task Graphs," *2022 IEEE/ACM Parallel Applications Workshop: Alternatives To MPI+X (PAW-ATM).*
- Github: https://github.com/TESSEorg/ttg/

This research was supported partly by NSF awards #1931347 and #1931384, and 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. We gratefully acknowledge the provision of computational resources by the High-Performance Computing Center (HLRS) at the University of Stuttgart, Germany.