LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-739012

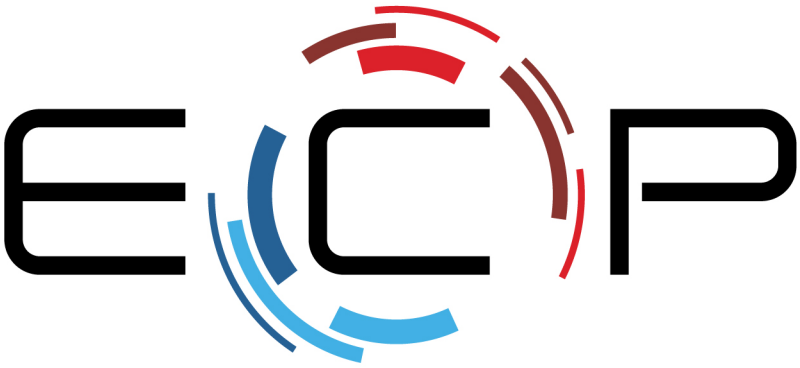
Milestone Completion Report STCO04-1 AAPS: engagements with code teams, vendors, collaborators, developers

E. W. Draeger

September 27, 2017

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.



Milestone Completion Report

1.3.8.04 ATDM LLNL AAPS Engagements
STCO04-1 AAPS: engagements with code teams, vendors, collaborators, developers

Erik W. Draeger
Lawrence Livermore National Laboratory

September 30, 2017

EXECUTIVE SUMMARY

The Advanced Architecture and Portability Specialists team (AAPS) worked with a select set of LLNL application teams to develop and/or implement a portability strategy for next-generation architectures. The team also investigated new and updated programming models and helped develop programming abstractions targeting maintainability and performance portability. Significant progress was made on both fronts in FY17, resulting in multiple applications being significantly more prepared for the next-generation machines than before.

1. INTRODUCTION

The Advanced Architecture and Portability Specialists team (AAPS) at LLNL was formed to help facilitate the transfer and dissemination of hands-on advanced architecture expertise to code teams across ASC. Rather than relying upon each code team to independently stay abreast of the latest developments in architecture, programming models and kernel optimization needed to make efficient use of new hardware, ASC code developers have access to an agile labor pool of computational scientists and computer scientists skilled in scaling applications on new, cutting-edge hardware. The team shares knowledge through a code repository and wiki as well as seminars, hackathons, and publications. The team includes specialists in key areas such as GPGPU programming, many-core programming, I/O, and parallel application development.

2. MILESTONE OVERVIEW

The AAPS team will work with select LLNL application teams to evaluate their codes, help to develop a portability strategy for next-generation architectures and help implement the chosen strategy. For FY17, the target application teams are the Ares multiphysics hydrodynamics code, deterministic transport codes Teton and ARDRA, and the MSlib EOS package. The team will also investigate new and updated programming models and will develop programming abstractions targeting maintainability and performance portability. Lastly, the team will perform co-design activities by developing and evaluating proxy applications in collaboration with vendors.

Milestone execution plan:

- Assist Ares team in their refactoring effort to target heterogeneous architectures using RAJA with Unified Memory. Work with RAJA developers to improve performance and robustness of performance-critical operations, e.g. reductions. Evaluate performance on early-access Sierra hardware and optimize Unified Memory performance where possible.
- Work with deterministic transport team to identify and implement portability strategy for Fortran code Teton. Collaborate with Nvidia to implement and compare OpenMP, OpenACC and CUDA performance for key kernels. Apply nested-loop RAJA constructs developed within Kripke proxy application to full ARDRA code.
- Assist MSlib team in profiling code and developing a refactoring strategy for heterogeneous architectures. Initial performance assessments will be carried out on Sierra early-access hardware. Different algorithms will be tested in a simplified proxy application to measure the extent to which existing code structure can be maintained.
- As limitations in the existing model are identified, the team will work with RAJA developers to extend the programming model to better support application needs. The CHAI interface will continue to be developed to simplify and automate data locality and movement within heterogeneous environments. Regular interactions with vendors will continue to identify bottlenecks and limitations in hardware and programming models/compilers.

3. TECHNICAL WORK SCOPE, APPROACH, RESULTS

In FY17, AAPS worked with the ARES code team to help implement their unified memory porting strategy. Team members worked to provide RAJA support and developed better error handling and unified reductions. Interoperability issues surrounding OpenMP 3, OpenMP 4.5, MPI, CUDA and RAJA were also explored and detailed performance analysis performed, working with tool developers in some cases to get access to alpha versions for tools not yet available on the Minksy architecture. An AAPS team member found a critical build error that was preventing compilation on early access Minksy nodes. Clang compiler plugins were developed to detect and warn on otherwise opaque RAJA and ARES implementation errors to speed development and limit debugging, as code developers are able to encode expert knowledge into the build process to inform future programmers of best practices through clear compiler warnings rather than obscure template compilation errors or run-time faults. These plugins were developed to be easily distributed and have already been extended to other codes, including ALE3D and ARDRA. AAPS worked in collaboration with the ATDM ProTools team to integrate the newly-developed Umpire hierarchical memory abstraction into Ares. All Ares allocations are now made through Umpire, including a device memory pool hidden behind the Umpire interface.

AAPS also assisted the deterministic transport codes ARDRA and Teton to help develop and implement porting strategy. For ARDRA, we continued to develop and improve the nested loop abstraction within RAJA, ultimately developing a new C++ 11 metaprogramming library CAMP that supports more compilers (including nvcc) than existing capabilities. AAPS contributed significantly to a major refactoring of the ARDRA code to make use of RAJA and CHAI. This required modification of the data structures throughout the code and modifying all loops to be compatible with the RAJA nested-loop construct, as well as rewriting the large problem setup portion of the code to consolidate and compartmentalize the data allocation and re-allocation. For Teton, we worked in coordination with Nvidia and IBM to resolve compiler and programming model (CUDA Fortran and OpenMP 4.5) interoperability issues between KULL (C++) and Teton (Fortran) on the early access Minsky nodes, then helped devise and begin implementing a strategy for parallelism and data movement. New GPU kernels have been integrated into the Teton mainline branch, including a non-linear solve kernel that yields a 3x speedup.

The AAPS team continues to work with code teams devise a heterogeneous architecture strategy for physics packages used by multiphysics applications. Working with the MSlib team and Nvidia and IBM vendor representatives, the team ported multiple versions of MSlib to use OpenMP 4.5 on the Minsky nodes in order to evaluate the performance and identify the current suitability of OpenMP 4.5 as a portable programming model. Numerous compiler and OpenMP 4.5 bugs were identified and reported to IBM, and initial profiling helped identify several kernels that can be optimized for improved performance. The team also completed development of the Cheetah proxy application Cheep. In addition to serving as a proxy app, Cheep will allow prototyping and evaluation of major refactoring strategies for several different single-physics applications to determine the extent to which the existing code base can be preserved and estimate performance tradeoffs associated with different approaches.

4. CONCLUSIONS AND FUTURE WORK

The AAPS model of multi-team engagements continues to prove successful. In addition to providing expertise and effort to code teams to help prepare them for new architectures, the team is gaining valuable experience in real-world multiphysics use cases and issues. The team is also closely involved with the development of the RAJA, CHAI and Umpire programming and data abstractions used to help large codes achieve maintainable portability. This combination is proving highly valuable to code teams and the broader ECP community.

ACKNOWLEDGMENTS

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. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.