LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-MI-692819

DOE-COE Breakouts

J. R. Neely, M. W. Epperly

May 23, 2016

Disclaimer

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

Breakout session: Tools/Compiler/System Requirements
Thursday, April 21
Lead: Edgar A. Leon

This session of about 20 participants included representatives from Cray, Intel, and IBM as well as two centers of excellence: LLNL and SNL/LANL. Six questions were posed to the audience: (1) What tools exist today that help achieve performance portability? (2) Is memory management the most important area to achieve performance portability? Should vendors provide an interface for tools to extract information of interest? (3) How can an application and libraries share resources amiably? Can an application specify what percentage of resources are willing to give up to a library? (4) What role should compiler technology play in performance portability? What features do we already have? What features are needed? (5) Will machine learning play a significant role for performance portability? and (6) What are some guidelines for performance portability with respect to I/O patterns (e.g., burst buffers)? How about memory layouts (for HBM, large shared caches, etc.)? From these questions four of them were discussed and are summarized below.

1. Existing tools that help with performance portability.
Several areas were identified including compiler-based tools (e.g, Cray's Reveal, OpenARC), libraries (e.g, SCR, MACSio), profiling interfaces (e.g, KokkosP, OMPT), and DSLs (CHiLL, AMRStencil). The group also identified a need for tools to work across architectures and provide and "integrated" view on heterogeneous systems. There is a clear desire to have common interfaces between tools across different platforms. For application developers it is time consuming to have to learn a new tool when moving to a new vendor's platform and thus even though the tool may help address a certain need, it is not used. Totalview and ScoreP are good examples of tools that work well across multiple platforms.

The DWARF debugging standard (http://dwarfstd.org/) was brought up since vendors spend a significant amount of time modifying/extending it to address its shortcomings. However, none of this work is contributed back to the community. It would be helpful to have the vendors work together to update the DWARF standard to include the extra information that tools developers are needing to create executables and extract relevant information from them. For example, the Intel compiler adds extra information that Intel tools consume; it would be useful if that information was standardized in DWARF version 5 (future) so that other tools could use that information.

Tools that provide a more unified view of a heterogeneous system seem to be lacking. One promising approach is IBM's HPM. Another promising tool to add/identify parallelism in codes is Cray's Reveal but it does not work with GPUs (OpenMP 4.5). Having Reveal work with GPUs would be extremely useful.

2. Tools for memory management on multi-level memories.

There is a strong desire for tools that give application and tool developers visibility into where data is moving as the application executes. Based on Karlin's presentation "Fundamental Cross Architecture Multi-Level Memory Support," the following features were identified to help application and tool developers understand data movement and improve the placement of data on a multi-level memory system: (1) Ability to mark user (subset) data structures and track them in the memory hierarchy throughout a code region. Associate location, effective access latency and bandwidth. Latency, in particular, was emphasized. For example, having a histogram of what data structures experience the highest latencies would be helpful; (2) Ability to query memory properties/attributes: available space (dynamically), latency, bandwidth (numactl-like); (3) Control placement using an open interface that would work across architectures, i.e., KNL and GPUs. KNL uses libNUMA, which is supported in Linux but libNUMA does not allow access to the features described here; (4) Track data transferred from one memory to another and correlate to application's objects/data structures. And, importantly, what entity moved the data (e.g., runtime system, user initiated, OS).

PAPI was brought up as a useful tool in the past but does not provide the memory information desired and PAPI is CPU centric. It would be useful to expand PAPI to be node-based but it may include work throughout software/hardware/OS stack. Another issue with PAPI is that the maintainers, apparently, have lost most of their funding and thus the tool, in many cases, is not accurate as brought up by Jeanine Cook in the presentation "The Importability of Performance Tools." There may be an opportunity for DOE to support PAPI or perhaps investigate other tools like Perfminer. There are also tools like Memspy, which seem to provide standard interfaces to get information about the memory subsystem but it is not clear whether this would be enough to support multi-level memories.

A key observation regarding these type of tools is that we need them to be accessible at the user-level, otherwise cannot be used by regular application or tool developers on DOE clusters. Finally, it is also desirable to identify different application uses cases of intended use of a multi-level memory system.

3. Applications and libraries playing nicely

There is a significant concern due to contention for resources between applications and libraries: How can application and libraries share resources effectively? What happens when the application needs all the memory, but then a library or other tool needs to execute and use resources? For example, an OpenMP application may need to call an MPI-only library or a library with pthreads. We need a mechanism to orchestrate friendly coordination between these. This may result in primitives or a "contract" to coordinate the needs of the application and libraries. Libraries could be parameterized based on this contract but the interface would need to be defined. Cray does this implicitly with their optimized libraries, e.g., BLAS. There is an implicit handshake between the app and the library. The library may move data between memories or decide, based on problem size, which device to execute on (e.g, CPU or GPU). It would be helpful to further understand what decisions are

made in these libraries and also what "application state" was changed after a library was executed.

4. Compiler technology and performance portability.
The group focused on two areas: code-to-code generation and descriptive vs prescriptive programming abstractions (e.g., OpenACC and OpenMP). First, code-to-code generation (source-to-source compiler transformations) can be of significant value to achieve performance portability. The transformed code can retain important semantic information that can then be harvested by architecture-specific compilers and achieve good performance. During the workshop several examples were demonstrated including OpenARC and the importance of high-level intermediate representations and Domain Specific Languages regarding stencils through frameworks such as CHiLL and ROSE. There are some disadvantages though mainly the ability to map back to the original source code. This is a major issue when debugging for example. In addition, it takes many months for a code team to "trust" a compiler and compiler options.

Second, descriptive approaches such as OpenACC give the compiler more flexibility to optimize code and reduces application developer's burden to tell the compiler exactly what to do to optimize their codes. In many cases, however, the compiler fails to optimize code because of many constraints that ultimately the application developer can easily resolve. Thus, the group proposed a combined approach that would allow the use of a descriptive interface first and depending on the results a prescriptive approach may be necessary.

OpenACC was developed to fill the gap between OpenMP and device computing capabilities. Now that OpenMP 4.5 has similar device support, OpenACC is in maintenance mode but feature frozen. It seems that some vendors will continue to support OpenACC with bug fixes but are targeting OpenMP for future work and improvements. Considering that OpenMP is a prescriptive approach, there is a need for the OpenMP forum to have a discussion about the role of descriptive approaches within OpenMP.