Workshop on Resource Arbitration for Dynamic Runtimes (RADR)

Pete Beckman, Emmanuel Jeannot, Swann Perarnau

Workshop on Resource Arbitration for Dynamic Runtimes (RADR)

Pete Beckman
Argonne National Laboratory
Northwestern University
Argonne, IL, USA
beckman@anl.gov

Emmanuel Jeannot
TADaaM Team, Inria
Talence, France
emmanuel.jeannot@inria.fr

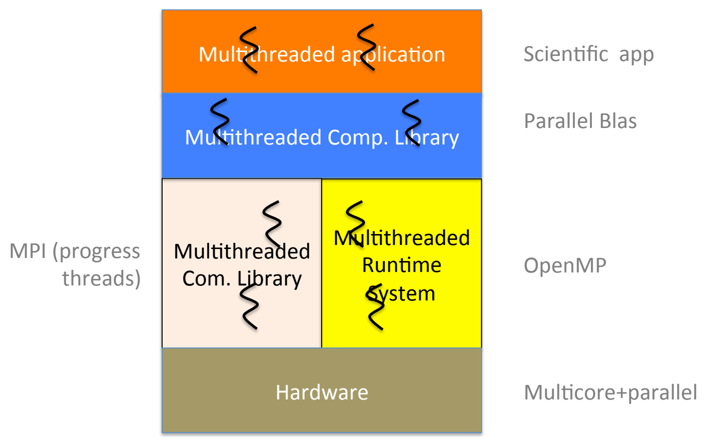
Swann Perarnau
Argonne National Laboratory
Argonne, IL, USA
swann@anl.gov

Abstract

The question of efficient dynamic allocation of compute-node resources, such as cores, by independent libraries or runtime systems can be an nightmare. Scientists writing application components have no way to efficiently specify and compose resource-hungry components. As application software stacks become deeper and the interaction of multiple runtime layers compete for resources from the operating system, it has become clear that intelligent cooperation is needed. Resources such as compute cores, in-package memory, and even electrical power must be orchestrated dynamically across application components, with the ability to query each other and respond appropriately. A more integrated solution would reduce intra-application resource competition and improve performance. Furthermore, application runtime systems could request and allocate specific hardware assets and adjust runtime tuning parameters up and down the software stack.
The goal of this workshop is to gather and share the latest scholarly research from the community working on these issues, at all levels of the HPC software stack. This include thread allocation, resource arbitration and management, containers, and so on, from runtime-system designers to compilers. We will also use panel sessions and keynote talks to discuss these issues, share visions, and present solutions.

Scope

Over the last five years, the number of nodes in large supercomputers has remained largely unchanged. In fact, the Oak Ridge National Laboratory computer leading the Top500 list, Summit, has fewer nodes than its predecessor, which is 20 times slower. Machines are getting faster not by adding nodes, but by adding parallelism, cores, and hierarchical memory to each compute node. This shift in how computers are scaled up makes it imperative that parallel computer resources within a node be carefully orchestrated to achieve maximum performance. Dynamically allocating and managing threads and the mapping of these threads to cores is a challenge that requires cooperation and coordination between the different components of the software stack.



Multithreaded application
Scientific app
Multithreaded Comp. Library
Parallel Blas
MPI (progress threads)
Multithreaded Com. Library
Multithreaded Runtime System
OpenMP
Hardware
Multicore+parallel

Figure 1: Software stack with different parts using threads

At the application level, a software component might use pthreads to express and coordinate concurrency. The application might also be linked to computational libraries, such as PETSc or Intel's MKL that could be multithreaded. Moreover, other parts of the application may use OpenMP parallel section (which are implemented with threads). Furthermore, the runtime system may need its own parallel resources, for example, to spawn progress engines for message libraries or remote invocation handlers. Currently, each component in this complex software stack is unaware of the other pieces. Therefore, threads can compete for cores and cause profound slowdowns to intranode collective operations and barriers. Moreover, currently, no mechanisms exist to query for unused cores, to reserve some of them, or to check which part of the application is using them. Resource allocation and partitioning by the operating system must be adaptive and well connected to user-level software components.
There has been research progress in this field. Tools such as hwloc can provide information on systems, such as topology. However, hwloc is not designed to handle direct allocation and partitioning resources. Likewise, it does not provide the interfaces required for software components to negotiate how to improve performance of the application through cooperative sharing of resources. Other approaches have been proposed, such as application composition, dynamic topology management or topology-aware core selection. Resource partitioning, enforced by the operating system using containers or within multi-kernels are also being investigated. Each of those approaches brings its own set of benefits and challenges, that need to be discussed within the community, compared with each other, and evaluated against benchmarks and use cases yet to be identified. As a relatively new and specific research area, it is difficult for researchers to find a place where to submit papers and discuss solution with the whole community. Therefore, we think it is of great interest for the HPC community to provide a venue to present these work in all their specificity and foster new discussions.

Program Committee

Program Chairs:

- Pete Beckman
- Emmanuel Jeannot

Publicity Chair:

- Swann Perarnau

Program Committee:

- Dorian Arnold, Emory University.
- Siegfried Benkner, University of Vienna.
- George Bosilca, Univ Of Tennessee.
- Hal Finkel, Argonne Ntl Lab.
- Karl Fuerlinger, LMU, München.
- Balazs Gerofi, U. Tokyo – Riken.
- Brice Goglin, Inria.
- Raymond Namyst, Univ. Of Bordeaux.
- Stephen Olivier, Sandia Ntl Lab.
- Tapasya Patki, Lawrence Livermore Ntl Lab.
- Marc Perache, CEA.
- Swann Perarnau, Argonne Ntl Lab.
- Rolf Riesen, Intel.
- Sameer Shende, U. of Oregon.
- Christian Terboven, RTW Aachen.

Program

This year, we have selected four papers to be presented at this workshop. They all deal about cooperation in the software stack: between MPI and TBB, for core allocation, for different execution models and for multithreaded applications. However, due to the COVID-19 crisis, we have been forced to implement a virtual meeting where speakers were able to present their contributions through a video-conferencing system.