The "State" and "Future" of Middleware for HPEC

Anthony Skjellum, PhD
Runtime Computing Solutions, LLC and
Department of Computer and Information Sciences
University of Alabama at Birmingham

During the last decade, a decided move to COTS computing has brought with it the adoption of standards for middleware – CORBA, MPI, and VSIPL principal among these, that have driven opportunities for optimization of performance, portability, and productivity. In this presentation, a discussion of the current state of MPI, VSIPL, VSIPL++, and related standards, and their impacts are discussed. Performance-oriented middleware standards that have been developed, but have not been widely used (*e.g.,* MPI/RT and DRI), are also mentioned, including opportunities for their renewed consideration, and how they have impacted other efforts. The continuing lack of comprehensive compiler solutions means that high performance middleware will continue to have to "pull its weight" for HPEC and defense programs over the next decade as well.

Several dimensions inform this discussion. First, an exploration of the performance-portability-productivity space and the so-called "price of portability" concept coined by Richard Games, remain important issues. The usefulness of reduced vendor lock associated with the use of middleware standards vs. the optimizability of the standards as they exist. The possibility to view middleware specifications not only as achievable through library implementations, but also via compilation and aspect-orientation is considered.

Second, the question of investment in further standardization is discussed. A decade ago, considerable efforts were underway to expand the options for high performance middleware, whereas today, these efforts are "shoe string" and highly focused on incremental improvement. Although MPI-3 has gotten underway, its consideration of areas that would enhance performance and productivity simultaneously remain limited… reviewing the existing standards for performance gaps is not central to the effort. The "price of portability" metric remains a concern for relatively naïve users of standards… advanced users must continue to use all the standards carefully to achieve high performance while conserving portability. This means a step in the wrong direction, typically, in the productivity space.

Third, we consider what is needed for broader adoption of MPI, VSIPL, VSIPL++, and potential follow-ons in the HPEC space. While there are clearly many legacy codes that use these standards and they have therefore been highly beneficial, discussing potential limitations to adoption is important. Is adoption limited because of the "cost of portability" or because of training, availability, or quality of implementation on specific platforms?

Fourth, we consider opportunities broadly to conserve existing API-type middleware standards, while driving higher performance and portability. This discussion comes in two areas: hardware-software co-design to enhance API performance on future systems, and software-software co-design based on aspect-oriented computing to help advance specifications from being strictly library based.

Fifth, we consider that as architectures evolve in the HPEC arena, there are important changes: new complexities have arisen, including more internal concurrency in processors, new classes of coprocessors (e.g., GPUs and FPGAs), and memory hierarchy will continue to grow in complexity. As such, we have to ask if existing middleware standards can survive, or if they have to be replaced. This is particularly a concern as there is a goal to optimize performance-portability for scalable architectures of the next decade. The abstractions of existing systems (MPI, VSIPL) codified architectural systems of the 1990's, which lacked several aspects of this complexity. On the other hand, COTS architectures of today are more similar to "clusters" and non-embedded systems of years ago.

Finally, we cover opportunities that have gone unmet in the last decade – and which could be met moving forward – to expand standardization on a "lean budget," consistent with a finite set of abstractions that developers can agree are manageable with middleware, and not directly by compilers. We give examples of what could and should be further standardized, and where should HPEC-specific profiles of standards be considered. For instance, are new classes of memory management libraries needed? Are there needs for heterogeneous computing interfaces that extend MPI or VSIPL? Are there classes of fault tolerant APIs needed for HPEC computing that can be standardized. Finally, are there cogent arguments against further standardization, prior to new practices arising in industry that begin their lifecycle as vendor-specific systems.

In order to make quantitative progress in middleware, it is important to understand how to map the requirements of defense programs, contractors, and stakeholders to computing systems in a way that leads to important cost reductions, improved software quality, and software cost reductions over long lifecycles. In the previous decade, we have seen resistance to certain standards. Can new and productive standards be developed that will significantly drive efficiencies and create dual-use scenarios? Arguments for and against will be offered. The concept of making "COTS" into "COTS+" without significant vendor lock will be discussed.

Overall, middleware, and principally MPI and VSIPL, have driven up the capability of defense and other HPEC applications to be performance-oriented and portable at the same time. Significant legacy applications have been developed, and these standards have proven useful. Questions remain: how do we get more benefit from middleware, where do we invest, what private and government stakeholders make those investments, when, and how? And, this paper will argue that investments, and efforts are long overdue and can be extremely beneficial, and even enhance the competitiveness of HPEC systems vendors.