Toward the Embedded Petascale Architecture: The Dual Roles of Middleware and Software Engineering in Future HPEC

Anthony Skjellum, PhD
tony@cis.uab.edu
University of Alabama at Birmingham

Introduction

For the past 15 years, HPEC has emphasized performance, portability, productivity, quality of service (QoS), reliability, and size-weight-power design requirements and trade-offs. Large performance gains will be achieved over the next 15 years. The first teraflop supercomputer (ASCI Red) was commissioned 15 years ago, and gigascale embedded systems were soon created (*e.g.*, PowerPC 7400 with Altivec). Analogously, future HPEC systems will incorporate petascale devices in today's terascale power-size-weight envelope, while supercomputers achieve exascale. Petascale embedded systems will be complex and difficult to program and use in production without reliable middleware and software engineering strategies to manage complexity and support developers.

This paper addresses the strengths and limitations of current middleware and software engineering technologies (*e.g.*, MPI, VSIPL, aspect oriented computing, design patterns) for use on petascale embedded application development. Specific examples of HPEC operations are used to identify capability gaps for petascale HPEC computing.

Emerging HPEC Vs. Existing Methodologies

Systems developed for the next 15 years will certainly incorporate some of the following features/concerns: massive concurrency, heterogeneity (CPU, GPU, FPGA, PIM), computation/network hierarchy, high transient fault rates, morphability and/or reconfigurability, asynchronous VLSI, 3D packaging and chip composition, advanced DMA, non-cache-coherent cores, advanced power management, multi-level hardware virtualization support, peer-oriented protocols, new data types (*e.g.*, intervals, analog signals), scratch pads, as well as COTS components**.** These technologies and their interactions will pose new, complex requirements on applications and by implication on the middleware, API standards, compilers, and software engineering methodologies that are used in order to support development, including tradeoffs in the performance-portability-productivity-QoS. Today's software ecosystem will need to be better integrated, generalized, and updated.

Legacy HPEC applications must evolve for petascale; such applications currently depend on standards, APIs, and standard practices that are often rooted in the 1980's and earlier. Petascale hardware concerns will lessen the effectiveness of legacy approaches. Developers will need new programming abstractions to handle petascale complexity and support practical applications. Existing middleware for high performance operations, data transfer, synchronization, and fault recovery, and others, will need to evolve to support performance, portability, predictability, QoS and other application requirements. Compilers will need to be "middleware aware" in order to reduce the runtime overheads for high productivity programming abstractions. New requirements may result in the complete or partial displacement of existing standards, since existing middleware such as VSIPL (sequential/parallel), MPI, DRI, PGAS languages, OpenMP, Pthreads, OpenGL, OpenCL, dataflow, and Verilog/VHDL each individually capture only parts of the architectural concepts, primitive operations, and operations of fully heterogeneous, real-time, fault-tolerant, embedded HPC systems as of now.

By considering petascale hardware, together with examples of operations representative of HPEC applications, we will be able to suggest a framework for petascale HPEC, while relating it to current and future practice. We will determine where current middleware can support these goals, what features future compilers will have to minimally provide, what new middleware and/or APIs are plausibly needed, as well as where use of software engineering methodologies could help significantly. The intent is to define the framework as specifically as possible. HPEC-relevant operations will therefore be considered and mapped to petascale hardware via current middleware. Operations include data transfer between different memory spaces with high overlap of communication and computation, selecting "where" to run a mathematical operation, managing faults, reconfiguring for low power or high performance, data independent computation (*e.g.*, 2D FFT followed by target recognition), and addressing jitter systemically.

References

[1] The MPI Forum, *MPI-2 Standard*, http://www.mpi-forum.org, accessed May 27, 2011.

[2] The VSIPL Forum, *VSIPL 1.03 Standard*, http://www.vsipl.org, accessed May 27, 2011.

[3] Khronos Group, OpenCL 1.1 Standard, http://www.khronos.org/opencl , accessed May 27, 2011.

[4] Open MP Standards Group, OpenMP 3 Standard, http://openmp.org/wp/ , accessed May 27, 2011.

[5] Model Driven Architecture, OMG, http://www.omg.org/mda/, accessed May 27, 2011.

[6] Anonymous, Aspect Oriented Development Resources, http://www.aosd.net, accessed May 27, 2011.

[7] Vivek Sarkar, et al, DARPA *ExaScale Software Study: Software Challenges in Extreme Systems*, http://users.ece.gatech.edu/mrichard/ExascaleComputingStudyReports/ECSS%20report%20101909.pdf , accessed May 27, 2011.

[8] Skjellum A., Cain K, et al., Data Reorganization Initiative (DRI), http://www.data-re.org , accessed May 27, 2011.

.