Programming Models

Facilitators

Paul Henning (LANL)

Sadaf Alam (CSCS)

Jonathan Carter (LBL)

Technical Committee Report
to the Hybrid Multicore Consortium

First HMC Roadmap Workshop
January 19-22, San Francisco

Breakout Participants

- Sadaf Alam, CSCS, (alam@cscs.ch)
- Jonathan Carter, LBL, (jtcarter@lbl.gov)
- Alan Coppola, OptNgn, (ajjc@optngn.com)
- Richard Graham, ORNL, (rlgraham@ornl.gov)
- Paul Henning, LANL (phenning@lanl.gov)
- Wen-mei Hwu, UIUC (hwu@crhc.uiuc.edu)
- John Levesque, ORNL, (levesque@ornl.gov)
- Al McPherson, LANL (mcpherson@lanl.gov)
- Piyush Mehrota, NASA Ames (piyush.mehrota@nasa.gov)
- Dave Norton, PGI, (dave.norton@pgroup.com)
- Philip Roth, ORNL, (rothpc@ornl.gov)
- Sonia Sachs (soniasachs@gmail.com)
- John Shalf, LBL, (jshalf@lbl.gov)
- Aniruddha Shet, ORNL, (shetag@ornl.gov)
- John Thorp, LANL (thorp@lanl.gov)
- Kathy Yelick, LBL, (yelick@eecs.berkeley.edu)
- Shujia Zhou, NASA, (shujia.zhou@nasa.gov)

CHARGE TO BREAKOUT SESSIONS

- Goal of Roadmap:
  - Identify technologies that need to be developed to make next generation, large-scale, accelerator-based systems "production ready"
  - Provide community input needed to prioritize and support activities
- Focus is near term, while keeping an eye toward to long term (avoid box canyons)
- Work with the other TCs to support the overall co-design of applications, architectures, programming, and performance and to build ties with and provide feedback to vendors.
- Develop strategies for early and broader access to these accelerator-based or future hybrid multicore systems.

CHARGE TO PROGRAMMING MODELS

- Identify and report on programming models for developing applications on large-scale (accelerator-based) hybrid computer systems in the near term and in the future.
- *Identify the types and degrees of parallelism provided by hybrid cores and to define key architectural metrics of this class of hybrid machine.*

SUMMARY OF PROGRAMMING MODELS TC

- Areas of interest:
  - Code and performance portability
  - Developer productivity: tools, programming for “mere mortals”
  - Data layout & motion, multiple disjoint address spaces, SIMD length, etc.
- Relation to other TCs
  - Relation to applications: algorithm design/selection
  - Relation to architectures: design roadmaps
  - Relation to performance: data motion costs, system modeling

Review of Grading Criteria

Urgency	Duration	Responsive	Applicability	Timeline
Critical Needed as soon as possible	**Long** Applicable for the foreseeable future	**High** Additional funding would enable significant progress	**Broad** Applicable beyond HPC	**Immediate** Results within 1-2 years
Important Needs to be done within 3 years	**Medium** Will be applicable for Exascale	**Moderate** Additional funding would enable progress	**HPC** Applicable to all of HPC	**Soon** Results within 2-5 years
Useful Needed after 3 years	**Near** Only applicable for immediate systems	**Low** Additional funding will not help very much	**Narrow** Only applicable to Hybrid Multicore systems	**Eventually** Results after 5 years

HMC Programming: Best Practices and Knowledge Transfer

- Description
  - Provide independent assessment of technologies.
  - Match algorithms to hardware.
  - Influence future investments
- Notes from Discussion
  - Reference implementations
  - Best practices
  - White papers & books
  - Benchmark suites
  - Illustrate range of available technology options
- Relations to other TCs
  - Applications: collaborate on design of architecture-aware algorithms
  - Libraries: preserve best practices, but algorithms should be revisited!
  - Architecture: co-design
- Related Projects
  - CUDA Zone, motifs, MAGMA project

Urgency	Duration	Responsive	Applicability	Timeline
Important	Medium	High	Narrow (a plus!)	Immediate

OAK RIDGE National Laboratory

BERKELEY LAB

Los Alamos NATIONAL LABORATORY

Transition Tools

- Description
  - Tools to facilitate refactoring existing code bases to new programming paradigms.
  - Tools for identifying acceleration opportunities.
  - Choosing the right hardware for the application.
- Notes from Discussion
  - Language interoperability is crucial
- Relations to other TCs
  - Applications: requirements
  - Performance: modeling of systems
- Related Projects
  - Compiler directives (e.g. OpenMP)
  - Language translation (e.g. C-to-CUDA, C-to-FPGA)
  - Performance analysis & modeling tool extensions (e.g. ROSE, TAU)

Urgency	Duration	Responsive	Applicability	Timeline
Critical	Medium	High	HPC	Soon

Debugging and Performance Support

- Description
  - Capability to access debugging and performance data on HMC hardware and runtime
  - Correlating data from heterogeneous hardware components
  - Bridging the semantic gap between low-level data and high-level programming models
- Notes from Discussion
  - Goal: Uniform interface between tools and architectural features for portability
- Relations to other TCs
  - Architecture: collaboration on two-way exchange of information on debugging and performance
  - Performance: analysis tools
- Related Projects
  - Consumers: NVIDIA Nexus, vampir, oprofile, TAU, TotalView, Allinea DDT, Charm++
  - PAPI

Urgency	Duration	Responsive	Applicability	Timeline
Important	Long	High	Broad	Soon

HMC and Non-HMC Performance Portability

- Description
  - Single code base for performance on multiple architectures.
  - Addressing explicitly-managed memory hierarchies
- Notes from Discussion
  - What are the implications of maintaining multiple code bases (V&V, feature creep, etc)
  - What breadth of application space?
- Relations to other TCs
  - Applications: what is "acceptable" performance, when needed?
  - Architecture: compatibility or general-purpose feature additions
- Related Projects
  - MCUDA, OpenCL, CUDA-Fortran
  - Autotuning

Urgency	Duration	Responsive	Applicability	Timeline
Important	Long	Moderate	Broad	Eventually

Expressive Programming Environments

- Description
  - Reduce effort to utilize accelerator hardware
  - Capture developer's intent in a more declarative way, develop back-ends for HMC
- Notes from Discussion
- Relations to other TCs
  - Applications: co-design of declarative programming environments
- Related Projects
  - Thrust
  - MATLAB
  - Python (Copperhead, SciPy)
  - Domain specific languages
  - HPCS Languages
  - FPGA Workflow (LabVIEW, C2H, MATLAB-to-FPGA)

Urgency	Duration	Responsive	Applicability	Timeline
Useful	Long	Moderate	Broad	Eventually

Breakout Summary

Topic	Urgency	Duration	Responsive	Applicability	Timeline
HMC Programming: Best Practices…	Important	Medium	High	Narrow	Immediate
Transition Tools	Critical	Medium	High	HPC	Soon
Debugging and Performance Support	Important	Long	High	Broad	Soon
HMC & non-HMC Performance Portability	Important	Long	Moderate	Broad	Eventually
Expressive Programming Environments	Useful	Long	Moderate	Broad	Eventually

Notes and Recommendations

- Testbeds: a large variety of small systems to test cross-platform applicability
- Clusters: useful to evaluate programming models (e.g. PGAS), but only up to a point
- Stability of development and execution environments
- Cross-cutting collaboration is critical