Applications & Libraries

Facilitators

John Turner

Sriram Swaminarayan

Technical Committee Report
to the Hybrid Multicore Consortium

First HMC Roadmap Workshop
January 19-22, San Francisco

OAK RIDGE National Laboratory

BERKELEY LAB

Los Alamos NATIONAL LABORATORY

BREAKOUT PARTICIPANTS

- John Turner, ORNL
- Sriram Swaminarayan, LANL
- Erich Strohmaier, LBL
- Thomas Schulthess, ETH
- Milton Halem, UM Baltimore
- Johnny Chang, NASA Ames
- Hatem Ltaief, UT(ennessee)
- Stephen Lee, LANL
- Hemant Shukla, LBL
- Vasily Volkov, UCB
- Jamal Moh'd Yusof, LANL
- Ann Johnson, Reservoir Labs
- Kalyan Kumaran , ANL
- Wen-Mei Hwu
- Pieter Swart, LANL
- Peter Messmer, Tech-X
- Dan Hitchcock, DoE

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.

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

OAK RIDGE National Laboratory

BERKELEY LAB

Los Alamos NATIONAL LABORATORY

BREAKOUT SUMMARY

Topic	Urgency	Duration	Responsive	Applicability	Timeline
Math & I/O Libraries	Critical	Medium	Moderate	Broad	Immediate
Novel algorithm research	*Critical*	*Long*	*High*	*Broad*	*Soon*
~~Intra-node Data motion Libs~~	~~Critical~~	~~Medium~~	~~High~~	~~HPC~~	~~Immediate~~
profiling tools	Important	Long	High	HPC	Eventually
Generic Scientific Toolkits	Useful	Long	High	Broad	Eventually
~~Architecture-aware Compiler / build systems~~	~~Important~~	~~Long~~	~~Moderate~~	~~Broad~~	~~Soon~~
~~Debugging~~	~~Important~~	~~Long~~	~~Moderate~~	~~HPC~~	~~Soon~~
Fault tolerance tools	Important	Long	High	HPC	Eventually

Libraries

- Description
  - numerical libraries
    - BLAS, LAPACK, Trilinos, FFTW, BGL, grid operators, AMR
  - I/O libraries
- Notes from Discussion
  - building-blocks of apps
  - scalable from desktop to HPC
  - diffusion of knowledge beyond specific libs
  - portability critical
- Relations to other TCs
  - Performance
  - Programming models
  - Architecture
- Related Projects
  - MAGMA
  - cuBLAS
  - Trilinos
  - PETSc
  - Adios
  - PVFS, PLFS, GPFS, etc.

Urgency	Duration	Responsive	Applicability	Timeline
Critical	Medium	Moderate	Broad	Immediate

OAK RIDGE National Laboratory

BERKELEY LAB

Los Alamos NATIONAL LABORATORY

Novel Algorithm Research

- Description
  - Methods development
  - Algorithm is some version of above method that we can implement
  - Implementation is a specific instantiation of that method
- Notes from Discussion
  - Implementations need to be architecture aware
  - Spatial and temporal locality is key
  - Time to solution should be kept in mind in addition to complexity and flops.
- Relations to other TCs
  - Programming models
- Related Projects
  - CFDNS on Cell
  - FEAST-GPU

Urgency	Duration	Responsive	Applicability	Timeline
Critical	Long	High	Broad	Soon

ERASED: Intra-node Data Motion Libraries

- Description
  - libs to facilitate data motion across platforms
- Notes from Discussion
  - analysis & performance feedback
  - expose memory model
  - low-level access to memory hardware
- Relations to other TCs
  - Programming models
  - Architecture
- Related Projects
  - OpenCL
  - Sequoia
  - Thrust
  - DaCS

Urgency	Duration	Responsive	Applicability	Timeline
Critical	Medium	High	HPC	Immediate

Profiling tools

- Description
  - data motion feedback
  - data location
  - Time to solution is critical
  - Energy to solution is critical
- Notes from Discussion
  - Equal ownership with performance
  - cache hits/misses
  - retired operations
  - dual-issue
  - bus contention
  - latency
  - packet size.
  - Ops/load can be useful
- Relations to other TCs
  - Performance
  - Architecture
- Related Projects
  - OpenSpeedshop
  - VTUNE
  - VAMPIR
  - Oprofile
  - gprof
  - Tau

Urgency	Duration	Responsive	Applicability	Timeline
Important	Long	High	HPC	Eventually

Abstract Scientific Toolkits

- Description
  - high-level expression of math / physics
  - Physics resides in Applications, CS resides in Programming models
- Notes from Discussion
  - Grid operation libraries
  - PDE libraries
  - Graph libraries
  - Success requires strong interaction between CS and Physics experts
- Relations to other TCs
  - Programming models
- Related Projects
  - SCOUT
  - libMesh
  - netCDF
  - Toolkits within matlab
  - BGL / PBGL

Urgency	Duration	Responsive	Applicability	Timeline
Useful	Long	High	Broad	Eventually

ERASED: Architecture-aware compilers

- Description
  - optimizing compilers with knowledge of underlying architecture
  - build system / tools
- Notes from Discussion
  - assume basic compiler available
  - assume MPD compiler will never exist
  - desire something in between (e.g. directives)
  - feedback, auto-tuning
- Relations to other TCs
  - Programming models
  - Architecture
- Related Projects
  - PGI
  - CAPS / HMPP
  - CUDA
  - R-Stream
  - GPUSS
  - CellSs
  - Scout

Urgency	Duration	Responsive	Applicability	Timeline
Important	Long	Moderate	Broad	Soon

ERASED: Debugging

- Description
  - something better than printf (and write)
- Notes from Discussion
  - luxury, not necessity
  - have survived with printf, but would love better
  - thread-awareness
  - non-intrusive
  - heterogeneous
  - aware of memory hierarchy
- Relations to other TCs
  - Architecture
  - Programming models
- Related Projects
  - compilers
  - PGI (pgdbg)
  - Totalview
  - gdb
  - Allinea
  - nvcc

Urgency	Duration	Responsive	Applicability	Timeline
Important	Long	Moderate	HPC	Soon

Resilience / Fault tolerance

- Description
  - system reports failures so app can continue
- Notes from Discussion
  - must move beyond checkpoint / restart
  - minimal impact on resources
  - Generic interaction with system
- Relations to other TCs
  - Architecture
  - Programming models
- Related Projects
  - compilers
  - Erlang
  - OpenMPI

Urgency	Duration	Responsive	Applicability	Timeline
Critical	Long	High	HPC	Eventually

SUPPLEMENTAL

Discussion with programming models
Need to develo methods that cross domains.
Algorithm development / design
    Implementation is usually tied to algorithm
Are algorithms different from libraries
Novel algorithm research that is architecture aware

What is a DSL?
    Is it just a library?
    Physics part of DSL belongs to Apps
    CS part of DSL belongs to Programming models

Memory hierarchies
    short term we need hierarchies exposed (DMAs etc)
    use transition tools when available
    transition tools get subsumed by compilers, etc.

Drop Intra node data libs
    Drop architecture aware compilers

Discussion with architectures section

What matters to applications?
    memory bandwidth

Impact of architecture changes to memory on apps

(a) syncronous behavior
    shared address space
    what is a good metric: ops/load?
    speed of light is not a limiting factor
    latency (hiding) is important

need to define a system interaction API

WE COULD deal with fault tolerance by automatic data migration, but
    HPC specific means higher cost
    not commodity, see above
    perhaps improve checkpoint performance instead
    MPI-3 has hooks for knowing that a node is about to go down
    system monitoring tools to let us know when a node is due to die
    could do redundant computations to deal with this
    Automated queing system to shrink / expand jobs (performance,

Is mixed (extended) precisionimportant
    Verification is next to impossible
    tools for determining precision needed
    FPGAs are a possible fit

Ken's notes
Applications & Architecture Pairing

Input from Apps about what metrics matter?

- Apps need more BW to memorieS (ultimately comes down to Ops/Load)

- Apps need tools to deal with memory heirarchies (abstract & portable
would be better)
    may be vendor desire to hide some private IP - "shim" like interfaces

- architecture features like moving memory on stack or die will help
latency which is goods for Apps

- latency hiding would be good; with enough parallelism this is all that
matters
    most archs don't seem to provide enough hooks for this
    - Cell/RR experience: user controlled local mem was good but with
issues, but automated (or teaching cache) would help
    more exposure from and control of asynch behavior
    desire ability to partition cache behavior into pools

Synchronous behavior vs. asynch?

Applications in the Scientific community are written to accomplish Science
and not to write an application.

CUDA vs OpenCL... (FOR Apps & Programming Models)
 - need portability; need/desire performance right away

Fault Tolerance greyed out by Apps!
 - what is unique about this area for HMC? (beyond Exascale Initiative)
 - this buck has been passed around so much it is worn out
 - need for architecture to expose faults to Apps to deal with
 - are some of the Apps desired architectural features possible?
 - fault prediction with migration (IBM example); heavy tax (performance,
and beyond HPC)
 - many types of faults may not provide sufficient time to "protect"
some can't be predicted, but many can
 - flash on nodes for faster local checkpoints
 - "Check Engine" light is a similar issue
 - Is Redundant computation necessary? (more discussion suggested)

memory capacity
issues arise
 - is there a need beyond just 32-bit & 64-bit

Performance and metrics
Need to evaluate architecture applicability for specific application

What is a good metric?
 ops/W is useless to applications
 ops/load is the arithmetic intensity, but is a crude indicator of performance expectation.
 time to solution is better
 energy to solution is better

Time to solution:
 make model of application
map to system characteristics
run 'what if' scenarios
 Leads to predictive modeling
    levels of accuracy / ease
    user level simple warm fuzzy
    professional level more accurate
    helps evaluate gains by chnaging to a different architecture

 A good lowerbound on expected performance gain is important

Autotuning:
 Not always the answer
 best practices need to be captured
 optimization can result from this
 but also need to explore different algorithms and methods
 genetic algorithms can help
 need to figure out how best to distribute physics across HMC

Integrated measurements
 How to measure performance?
    counters
    latency
    pipeline stalls
    memory hits/misses
Must be protable, calibrated, and usable

OAK RIDGE National Laboratory
BERKELEY LAB
Los Alamos NATIONAL LABORATORY

NOTES AND RECOMMENDATIONS

- Hardware simulators are useful before hardware is available
- As soon as hardware is available, we need a few prototype nodes per site, preferably one per developer
- Small testbeds of 10-100 nodes within a year
- Leadership platform that is 10x more powerful than today's fastest supercomputers within 2-3 years