



Results from Santa Fe

Pete Beckman

Optimism, Excitement

Good agreement on several themes

- ❑ International collaboration is required to address scale of challenge
- ❑ Research community and vendors must work together
- ❑ Open Source is vital component
- ❑ Large changes in platforms expected

	A	B	C	D	E	F	G
1	Software Requirements for HPC						
2							
3	Site:		ORNL				
4	System:		Cray XT3				Note that this list is best viewed as a datab
5	Submitted		8/25/06				
6	Contact:		Jeff Vetter, vetter@ornl.gov				
7							
8	Site	System	Node Type	L1 Category	L2 Type	L3 Function	Package
9	ORNL	Cray XT3	All	App Support	Library	I/O & Storage	HDF5_PAR
10	ORNL	Cray XT3	All	App Support	Library	I/O & Storage	HDF5_SERI
11	ORNL	Cray XT3	All	App Support	Library	I/O & Storage	netCDF
12	ORNL	Cray XT3	All	App Support	Library	I/O & Storage	netCDF_para
13	ORNL	Cray XT3	Compute	App Support	Library	Math	PetSC
14	ORNL	Cray XT3	Compute	App Support	Library	Math	Aztec
15	ORNL	Cray XT3	Compute	App Support	Library	Math	BLAS
16	ORNL	Cray XT3	Compute	App Support	Library	Math	FFTWack
17	ORNL	Cray XT3	Compute	App Support	Library	Math	FFTW
18	ORNL	Cray XT3	Compute	App Support	Library	Math	LAPACK
19	ORNL	Cray XT3	Compute	App Support	Library	Math	MUMPS
20	ORNL	Cray XT3	Compute	App Support	Library	Math	Scalapack
21	ORNL	Cray XT3	Compute	App Support	Library	Math	SPRNG
22	ORNL	Cray XT3	Compute	App Support	Library	Math	Support_U_DR
23	ORNL	Cray XT3	Compute	App Support	Library	Math	libsci
24	ORNL	Cray XT3	Compute	App Support	Library	Math	acml
25	ORNL	Cray XT3	Compute	App Support	Library	Math	Hypr
26	ORNL	Cray XT3	Compute	App Support	Library	Mesh	Metis
27	ORNL	Cray XT3	Compute	App Support	Library	Mesh	ParMetis
28	ORNL	Cray XT3	Service	App Support	Tool	File Transfer	BCDP
29	ORNL	Cray XT3	Service	App Support	Tool	File Transfer	FFP
30	ORNL	Cray XT3	Service	App Support	Tool	File Transfer	FFP
31	ORNL	Cray XT3	Service	Prog Env	Language	Compiler	C/C++
32	ORNL	Cray XT3	Service	Prog Env	Language	Compiler	C/C++
33	ORNL	Cray XT3	Service	Prog Env	Language	Compiler	C
34	ORNL	Cray XT3	Service	Prog Env	Language	Compiler	FORTRAN77
35	ORNL	Cray XT3	Service	Prog Env	Language	Compiler	FORTRAN90
36	ORNL	Cray XT3	Service	Prog Env	Language	Compiler	UPC

Exascale Target Schedule

- One possible platform schedule, based on strong and continued investment:
 - 2012: 10-30pf
 - 2015: 100-300pf
 - 2018: 1000pf

- What kind of Exaflop?
 - ▣ One that can do large multi-scale problems
 - ▣ And great dense linear algebra
 - ▣ A possible heterogeneous core

Software Barriers



- Evolution or Revolution
- Key Challenges:
 - ▣ ***Concurrency***
 - ▣ ***Energy (Power)***
 - ▣ ***Resilience (Fault tolerance)***

Exploration (1 / 4)

- Revolutionary storage class memory is coming:
 - 1000x < latency

- A power envelope will become part of design framework

Actions we can take

- Value of storage class memory: Need to have the HPC community united in articulating the value proposition associated with storage class memory. The critical break points in terms of bandwidth, density, cost and latency need to be understood to help guide the technology development.
- Power : This is somewhat a mindset change. Applications will eventually need to think of their computing resource as a total energy budget and they need to optimize within this. Fortunately much of performance tuning also drives toward energy efficiency... but not always. Tools and reports that detail the energy usage need to be accessible to users.
- Reliability: The realistic adoption, cost and risk of fault tolerant algorithms must be assessed and these should be traded off against hardware cost and risk. The systems can not move in a direction that "might" be acceptable from a reliability perspective. This makes a software solution very difficult.

Revolutionary Software Areas for Exascale:

- Storage class memory is coming: Technology will offer 1000x less latency but there are many other dimensions to this. Need to think through the possible directions to use this technology.
- Systems are transitioning to being power optimized. Application developers are still focused on performance optimization regardless of power. In a world where there is a power budget, software should play a role in optimizing performance through optimization of Perf/ Watt. (with total power being a hard facility constraint)
- Reliability: This is not an issue of what will we do when systems can not be made reliable. This issue is making the best trade-offs between hardware, system software and fault tolerant applications.

Evolutionary Software Areas for Exascale:

- Extend current program models through single node threading of messaging. Eliminate "per task" scaling terms in messaging layer to allow for higher "flat scaling".
- Allow for mixed programming models to coexist. We need a bridge to new programming models that is not an all or nothing proposition.
- Enhance job flow to enable many concurrent capability scale jobs. (similar to the emerging approach at LLNL) This is likely to be a common early usage model for Exascale.
- Open source can be a very good thing for vendors and end users but we need to find a way share the responsibility and risk.
- Educate young people in parallel programming.

Exploration (2/4)

- Revolution in multi-scale/multi-physics simulation frameworks
- Revolution in visualization and post-processing
- Need new programming models for hybrid programming
- Need unification for heterogeneous programming models

How do we get it done to develop community-supported open source software to address these 6 areas, what is needed?

Need for International Task Forces on:

1. Parallel visualization tool. The community should focus on a small number of tools. VISIT and Paraview seems good candidates
2. Remote and collaborative post-treatment tools
3. Meshing tools. Need for an international joint effort between academic, commercial companies and end users
4. Common data model and associated libraries. Providing an international standard model for mesh and fields exchange and services (localization, projection, interpolation, arithmetic operations, ...)
5. Supervising and code coupling tool. Unifying the software developments often driven by end users communities (climate, energy, ...)
6. Uncertainties Quantification. Uncertainty analysis framework, Uncertainties referential (methodologies and tools, Open TURNS)
7. Algorithms/solvers and data structures, solver interface
8. Fault tolerance. Need a joint effort involving OS, compilers, middleware/libraries, numerical solvers/algorithm researchers and engineer communities

Research policy:

How do we get it done?

A software research/development process model

New problems are defined (from new hardware and demands)

divergence

Many ideas are proposed

convergence

Standard development activity

Deployment for application end-users

admaps, cost roadmap for HPC software

forum structured around large (i.e. Transport, Defense, ...)

co-funding, single country funding

EDF

where "revolution" is

treatment tools

mesh healing, CAD healing for meshes (AMR like 2020?)

and associated services, adapted processing, calculation distribution and

14

Proposal for "Parallel Programming Languages" area

mesh and fields) and associated

Many parallel programming languages have been proposed, but ...

- Many people still use MPI ...
- OpenMP is now "standard" for programming multi-cores
- What about distributed memory programming?

NOTE: Restrict us parallel extension of existing languages (C/F95) for end-users.

- NOT HPCS languages and Java-based.

How about PGAS (UPC and CAF)?

- Local view parallel programming
- Already standard?

Global view parallel programming

- We should learn from HPF history
- Locality, efficient communication ...
- Any way to Combine to local view programming?

Cost to other partners

Cost to end users

Programming cost

HPF

PGAS

MPI

community-supported

se 6 areas, what is

EDF

XcalableMP

<http://www.xcalablemp.org>

15

it may help to succeed and to

- one active leader and the recognition by key players
- A roadmap and a validated business model (at least for the leader)
- An ecosystem of partners for the software development, diffusion and associated services (installation, deployment, maintenance, specific developments)

Using Open Source software, some issues to be aware of: how they are supported, deployed, visibility of the roadmap, associated risks (as an example, moving from Qt3 to Qt4 cost 400 days of development to the SALOME project).

Suggestion:

- Identification of existing HPC Open Source software (cf. P. Beckman list)
- Promotion of an international HPC source forge for Open Source software diffusion?

9 June 27, 2009 EDF R&D

EDF

Exploration (3/4)

□ Revolution Needed

- Reliability
- Expressing Parallelism
- Programming Tools

□ Other issues

- Full SW life cycle
- Product differentiation
- Testing and development practices

Where is evolution required?

- MPI
 - It's portable
 - It's ubiquitous
 - It isn't going away
- Thread Packages
 - Stable, portable, general user-level
 - Enable easier implementation of OpenMP
 - Allow oversubscription of HW threads
- Operating System
 - Extremely lightweight with global functionality (memory management, communication, etc.)
 - Heavily multithreaded locally for latency tolerance

Where is revolution required?

- Software to enable reliable systems built with unreliable parts
 - Infrastructure to enable application resiliency
 - Programming Models
 - System Software
 - APIs

Finding and Expr

- User perspective
- Compiler perspec
- Make extremely f
- Exploit heteroger

Programming Tool

- Intelligently collec
- Provide space eff
- Collapse, reduce

How do we get it done?

- Define the overall architecture
 - Can we converge on a common architecture?
 - Establish well-defined interfaces between SW layers
 - Dedicated architects throughout the effort
- Establish a community for key projects
 - Dedicated maintainers
 - Research + Industrial partnerships with funding for both
 - User community participation
- Avoid "Design by Committee"
 - HPF, Ada are examples to avoid
 - Respected leaders make the tough calls

How do we get it done?

- Focus on the full SW life-cycle, not just the initial development
 - Test and integration
 - Maintenance
 - Management of the rate of change
- Provide a common **exascale** test and integration platform
 - All components tested at scale on a reference platform
 - Strong focus on:
 - Mainline testing
 - Error-path testing
 - Edge-condition/interface testing
- Resolve Differentiation Needs vs. Commonality Needs
 - Hardware has been commoditizing over time
 - Can common SW provide opportunities for vendor differentiation?

Exploration (4/4)

- Revolution:
 - Fine-grain Async parallelism
 - Locality
 - Co-design HW/SW
- Follow best-practices from community software development
- Validation and Uncertainty Quantification important

Three Software areas where Revolution is Required

1. Fine-grained Asynchronous Parallelism

- Weak scaling and bulk-synchronous parallelism will not deliver billion-way concurrency needed in Exascale systems
- Instead require unified abstractions of asynchrony and concurrency for multi-core & cluster parallelism
 - Subsumes threads, shared memory, message-passing, active messages, ...

2. Locality Models

- Data movement will be major contributor to energy consumption in Exascale systems
- Need locality models that enable programmer, compiler, and runtime to handle multiple levels of memory hierarchy

Exascale systems
Interfaces that are critical bottlenecks, require support for software-hardware co-

Revolution x 3



- Portability
 - Architecture dependent code-generation
 - Dynamic adaptation
 - Check out on one platform check in on another
- Programmability
 - Develop systems that let us drive the machine with the hood down – better abstractions
- Dependability
 - On this scale things will fail – but it shouldn't mean they're broken
- Validation
 - Garbage generated in milliseconds is still garbage
- What can we learn from our formal methods colleagues?

Hardware Interface

coherence



- Address ranges for which coherence will be managed by software
- Address ranges with values that are guaranteed to be read-only (immutable) for certain application phases
- Network bandwidth partitioning for different forms of data movement and communication
 - PGAS, RDMA, Message passing, Stream processing, ...
- Other network reconfigurable parameters
 - Topology, Packet size, ...
- Power management
 - Frequency scalars, Voltage scalars, ...

Playing together



- Collaborative development of a roadmap for exascale software – several such already underway at the national level
- We need better coordination at the international programme level including mechanisms for collaboratively funded research and development
- Integration of applications, numerical and system software – silos of activity will not achieve our aims – US is better at than UK at this.
- Better models for sustainability
 - Community support?
 - Industry take-up
 - Need to ensure exascale efforts are not for the few
- Shared knowledge base required.

Software-Hardware Co-Design

distributed tasks (async, spawn, ...)

data structures

and data (places, locales, ...)



- Hardware support for virtual-to-physical translation and inter-processor data transfers
- Collective and point-to-point synchronization with dynamic parallelism (barriers, phases, ...)
- Hardware support for intra-node & inter-node synchronization and communication
- Producer-consumer parallelism (single-assignment vars, futures, ...)
 - Hardware support for full-empty bits
- Isolation and mutual exclusion
 - Transactions, fine-grained locks
- Data parallelism
 - Vectors, SIMD, SIMT

Non-architectural Issues



- Science drivers must be closely tied to effort
- What funding and governance model are most supportive?
- How do we set priorities?
- Who assumes risk for hitting R&D targets?
 - ▣ How do we handle failures?
- More integrated testing and development vital
- Close communication with coordination with funding agencies is required for success