Overview of the National Center for Computational Sciences

James B. White III (Trey)

trey@ornl.gov

NLCF Computational Chemistry Workshop

August 2, 2005

Acknowledgement

Research sponsored by the Mathematical, Information, and Computational Sciences Division, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

Outline

- Leadership Computing Facility (LCF)
- Jaguar (Cray XT3)
- Phoenix (Cray X1E)
- Ram (SGI Altix)
- LCF Proposals

Leadership Computing Facility

- High priority, high payoff computationally intensive experiments
- Enable new breakthroughs in science
- Enable DOE SC to be “first to market” with important scientific and technological capabilities, ideas, and software
- Limited set of scientific applications (perhaps 10 per year)

Leadership Computing Facility

- FY06 Call for Proposals
  - http://hpc.science.doe.gov/proposalCallFY06.do
  - http://nccs.gov/
- Closes August 12
- Announcements September 15
- Awards begin October 3

Leadership Computing Facility

- Jaguar
  - 5,294-processor Cray XT3
- Phoenix
  - 1,024-processor Cray X1E
- Ram
  - 256-processor SGI Altix

Jaguar

- Cray XT3 (Red Storm)
- 2.4 GHz AMD Opteron
  - 4.8 GF peak
  - 2 GB memory
  - 1 MB L2 cache
- 5,212 application procesors
  - 10 TB aggregate memory
- 120 TB scratch disk space
  - Lustre
  - Anticipated multiple GB/s

Jaguar upgrade path

Cabinets	Performance	Processors	Memory	Disk Space	I/O Bandwidth
56	25 TF	5,300	10 TB	120 TB	15 GB/s
120	54 TF	11,300	23 TB	240 TB	30 GB/s
120	100+ TF	22,700	46 TB	480 TB	60 GB/s

Cray XT3

- Each Opteron runs Catamount microkernel
  - Limited OS features
  - Distributed memory only (no SMP)
- Cray 3D mesh attached to Hypertransport
  - >1 GB/s each way in/out of an Opteron
  - Multiple GB/s on each link
  - Overpowered mesh compensates for contention
- Ping-pong latency not great
  - <10 μs soon, eventually <5
- Global latency scales
  - Microkernel eliminates OS jitter

Jaguar strengths & weaknesses

- Strengths
  - Memory bandwidth and latency
  - Interconnect bandwidth
  - Scalable latency
  - Aggregate memory (10 TB)
- Weaknesses
  - Ping-pong latency (getting better)
  - Cross-compiler environment
  - Lack of support for OS features

Catamount: What's missing

- Pipes, sockets, TCP/IP
- Dynamic process control
  - "*exec*", "*popen*", "*fork*", "*system*"
- Dynamic libraries
- Pthreads
- OpenMP
- "*/proc/cpuinfo*", "*/proc/meminfo*"
- "*mmap*"
- …

Applications for Jaguar

- Bigger experiments
  - More grid points
  - More memory
  - More processors
- Non-vector implementation
  - Deep scalar decision trees
  - Complex software architecture
  - Pointers abound
- Low to very heavy communication

Mismatches for Jaguar

- Large local memory
  - > 2 GB per processor
- Out of core?
  - No processor-local disk
  - Can Lustre handle it?
- Limited coarse-grain parallelism
  - Tens to low hundreds of processors
  - More time steps instead of more grid points
  - Latency bound
  - Time to consider Phoenix

Using Jaguar

- Portland Group Compilers
  - Fortran 95, minimal 2003 features
  - No C99, but ANSI C++
  - Recent performance improvements
  - Cray wrappers: "*cc*", "*ftn*"
- Cross compilation
  - Linux to Catamount
  - No Pathscale or Intel
  - Static libraries
  - "*<library call> is not implemented and will always fail*"

Using Jaguar

- Cray performance tools
  - "*pat_build*", "*pat_report*"
  - Apprentice^2 graphical tool
  - Good stuff!
- Totalview
  - Interactive
  - Attachment to running processes
  - Post-mortem
  - Command line or graphical

Using Jaguar

- Global Arrays
  - Shmem won't have AMOs until 2.0 release (months away)
  - MPI-2 one-sided fully supported in 1.2 release (to be installed this month)
  - "*MPI_Win_Lock/Unlock*" may not complete until there is an MPI call at the target (complies with spec)

Phoenix

- Cray X1E
- 1.13 GHz Multi-Streaming Processor (MSP)
  - 18 GF peak
  - 2 MB L2 cache
- 4-MSP SMP node
  - 8 GB of shared memory
- 994 application MSPs
- 2 TB aggregate memory
- 12 TB scratch disk space
  - XFS
  - ~1 GB/s

X1 to X1E

- 800 MHz to 1.13 GHz
  - Cache bandwidth scales with Hz
  - Improved memory management
- “Dual core” nodes
  - One 4-MSP node becomes two interleaved 4-MSP nodes
- Same memory system and interconnect
  - Memory is partitioned between interleaved nodes
  - 4 GB/MSP to 2 GB/MSP
  - Nodes contend for bandwidth

X1 to X1E

Benchmark*	X1	X1E
Single-MSP Stream Triad	24.02 GB/s	32.82 GB/s
Replicated Stream Triad	21.75 GB/s	13.23 GB/s
Single-MSP Random Access	208 Mup/s	249 Mup/s
Replicated Random Access	209 Mup/s	136 Mup/s

* http://icl.cs.utk.edu/hpcc/

Phoenix strengths & weaknesses

- Strengths
  - Powerful processors
  - Memory bandwidth (local and global, stride-1, strided, and random)
  - Latency hiding, local and global
  - Systematic tuning strategy
- Weaknesses
  - Slow scalar
  - “Some tuning required”
  - Limited memory per MSP

Applications for Phoenix

Applications for Phoenix

- Hard ones

Applications for Phoenix

- Cache unfriendly
- Growing cost per “grid point”
- More time steps
- Limited coarse-grain parallelism
- Tight synchronization
- Fixed startup cost

Why X1(E) is special

- Vectors
  - Provide fine-grained parallelism
  - Hide latency
  - Systematic tuning strategy
- Globally addressable memory
  - Minimize latency
  - Extend benefits of vector to remote access
  - Hide latency (as with local memory)

Mismatches for Phoenix

- Limited by memory (> 2 TB)
- Limited by memory per processor (> 2 GB)
- Limited by I/O bandwidth (out of core)
- Runs efficiently on general-purpose processors
- Easily scales to zillions of processors
- Non-vector implementation
  - Deep scalar decision trees
  - Complex software architecture
  - Pointers abound

Using Phoenix

- Build on Intel-Linux cross compiler
  - X1E has poor scalar performance
  - Building on X1E is 5-10x slower
  - X1E supports fewer tools (no E-macs, minimal Python, *etc*.)
  - "*configure*" can be tricky either way
- Submit batch jobs and access scratch files from cross-compiler system

Using Phoenix

- Initial port may* be slow
  * will probably be
- Build and generate loopmarks
  - Shows vectorization & multistreaming, or why not
- Instrument executable with "*pat_build*"
- Run and generate line profile
  - Shows which lines of which subroutines take the most time

Using Phoenix

- Use loopmarks to determine why time is spent where it is
- Vectorize and multistream
  - Add directives
  - Modify loops locally
  - Promote arguments to vectors
  - Globally modify data structures

Using Phoenix

- Repeat vector tuning until satisfied or communication bound
- Mitigate communication bottlenecks with targeted Co-Array Fortran or UPC
- Debug with TotalView and "*print*"
- Run regression tests for compiler upgrades!

Ram

- SGI Altix
- 1.5 GHz Intel Itanium
  - 6 GF peak
  - 6 MB L3 cache
  - 8 GB per processor
- 256 processors
  - 2 TB shared memory
- 32 TB of disk
  - XFS
  - 24 TB being installed

Using Ram

- Intel compilers
- Soon to be upgraded to SGI ProPak 4
- Currently used for computation
  - Batch systems
  - Many users
- Will be “support” system for LCF
  - Pre- and post-processing, data analysis
  - Not explicitly allocated
  - Some computing capacity?

LCF Proposals

- Download proposal form (MS Word)
  - http://nccs.gov/LCF/review.html
  - Due next Friday, August 12
- 6,000,000 MSP-hours on Phoenix
- 31,000,000 processor-hours on Jaguar
- 90% for ~10 big projects
- 10% for pilot projects (process TBD)

Technical reviews

- NCCS will perform technical reviews
  - Scientific Computing Group
  - Cray Supercomputer Center of Excellence
- Currently performing “Computational Readiness” reviews for INCITE
  - 600,000 MSP-hours on Phoenix
  - 3,000,000 processor-hours on Jaguar
- LCF similar to INCITE?

Computational readiness

- Comment on the reasonableness and appropriateness of the proposed request for computational resources.
  - If the request is better suited for one of the other INCITE computational resources, please identify the appropriate system.
- Comment on the appropriateness of the proposed computational approach.
- Discuss the technical readiness of the project, including:
  - Will the project make good use of the resources requested?
  - Are the codes ready to run?
  - Have they been optimized for the resources requested?
  - Does the code scale well?

Computational readiness

- ❑ Completely ready and scales well
- ❑ Completely ready but some scaling or other minor issues need to be resolved
- ❑ Partially ready but likely to port and run successfully
- ❑ **Partially ready with major issues to resolve***
- ❑ **Not ready***
- Provide a justification for the selection. Please identify any issues that need to be resolved.

** Proposal will not continue in FY2006 INCITE review process*

Questions?

- National Center for Computational Sciences
  http://nccs.gov/
- NCCS Technical Information
  http://info.nccs.gov/
  help@nccs.gov
- LCF Call for Proposals
  http://nccs.gov/LCF/review.html
  http://hpc.science.doe.gov/proposalCallFY06.do