Experiences with the Cray X1 at Oak Ridge National Laboratory

James B. White III (Trey)
trey@ornl.gov

July 14, 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.

X1 experiences

- X1(E) at ORNL
- Why X1E
- Using X1E
- Future

National Center for Computational Sciences

- Was 512-MSP X1 (2 TB memory)
- Upgrading to 1024-MSP X1E (2 TB memory)
- Also installed 5294-Opteron XT3

Why X1(E)?

- Breakthrough computational science
- Computation rate limits science progress
- Most-challenging computational problems

Most-challenging problems

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

Examples

- Gyrokinetic microturbulence in fusion plasmas
- Dynamic Cluster Approximation Quantum Monte Carlo simulation of superconductivity
- Explosion of core-collapse supernovae
- *Ab initio* calculations of double photo-ionization
- Molecular dynamics
- Coupled Climate System Model (see MS69)

Your Example Here!

- DOE Innovative and Novel Computational Impact on Theory and Experiment (INCITE)
- DOE Leadership Computing Facility (LCF)

INCITE

- 10% of NCCS X1E and XT3 for FY2006
- “open to all scientific researchers and research organizations, including industry”
- Call closes tomorrow
- http://hpc.science.doe.gov/

Leadership Computing Facility

- 80% of NCCS X1E and XT3 for FY2006
- Must be relevant to the mission of the DOE Office of Science
- Around 10 major projects (think “Grand Challenge”)
- Call closes August 12
- http://hpc.science.doe.gov/proposalCallFY06.do

Applications for X1E

- Limited by computation rate
- Runs at low efficiency on general-purpose processors
- Communication overhead limits scalability

Applications not for X1E

- Limited by memory size
- Limited by I/O bandwidth (out of core)
- Runs efficiently on general-purpose processors
- Easily scales to zillions of processors

Running on X1E

- 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

Running on X1E

- 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

Running on X1E

- 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

Running on X1E

- 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!

Future

- X1 a relic of the past?
- Or road to the future!

Future

- Gate counts keep increasing
  - Floating-point units get cheaper
  - More fine-grained parallelism
- Clock-speed increases are stalling (Heat!)
- Bandwidth may be catching up
  - Wire signal rates continue to increase
  - Optical communication will get cheaper

Future

- How do we use fine-grained parallelism?
- How do we hide latency?

Answer

- Vectors
- Globally addressable memory

Vectors

- Provide fine-grained parallelism
- Hide latency
- Work today
- Natural progression to more gates
- Systematic tuning strategy

Globally addressable memory

- Minimize latency
- Extend benefits of vector to remote access
- Hide latency (as with local memory)
- Local/remote hierarchy allows scalable architectures

Example

```
                      !Scatter
                      call sync_all()
                      if (this_image() == master) then
MVw----------<           do i = 1, num_images()
MVw                         x[i] = y(i)
MVw---------->           end do
                      end if
```

Example

```
                          !Gather
                          call sync_all()
                          if (this_image() == master) then
MVw----------<               do i = 1, num_images()
MVw                             y(i) = x[i]
MVw---------->               end do
                          end if
```

Example

```
!Gather
call sync_all()
y(this_image())[master] = x
```

Programming for the future

- Clearly present fine-grained parallelism
- Allow latency hiding (local and remote)

Programming for the future

- Operate on adjustable sub-aggregates
  - Not scalars (to allow vectorization and piplining)
  - Not the whole domain (to allow caching)
- Avoid false dependencies
  - Pointers!
  - I/O statements inside loops (for debugging)
- Also faster on general-purpose processors!

Programming for the future

- Use modules* instead of passing arguments (if you always pass the same object)
  - Easier promotion of scalar procedures
  - Easier promotion of variables to co-arrays**
  - Compilers can "see" the variables better
  - Adding "arguments" is a local modification (not throughout call stack)

*Fortran, the language of the future!
**Co-arrays will be in the Fortran 2008 standard.

Programming for the future

- Use modules instead of user-defined types (if there are one/few instances of that type)
  - Easy promotion of variables to co-arrays
  - Avoids artificial dependencies
  - Encourages operations on aggregates
  - Simpler for others to understand
  - Simpler for compilers to understand

Summary

- Cray X1(E) is for challenging Grand Challenges
- Hiding latency and enabling fine-grained parallelism will be critical for progress
- X1 series is designed accordingly
- Grand Challenge applications must be too

Try it yourself!

- INCITE
  - Closes tomorrow
  - http://hpc.science.doe.gov/
- NCCS LCF
  - Closes August 12
  - http://hpc.science.doe.gov/proposalCallFY06.do
  - http://nccs.gov/LCF/review.html