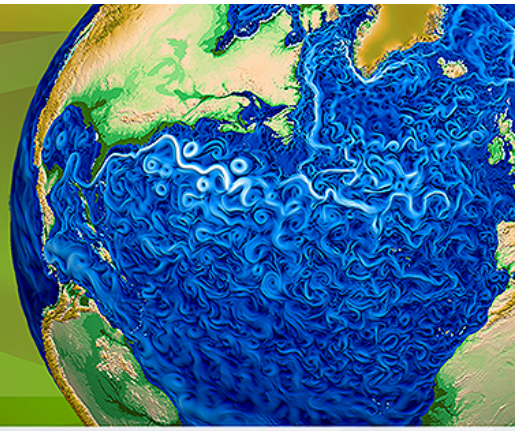
ACME
Accelerated Climate Modeling for Energy



Exascale Computing and Earth System Modeling

David C. Bader and Mark Taylor
ACME Project
June 27, 2017

U.S. DEPARTMENT OF ENERGY

Architectures Already Disruptive

- Ideal machine - powerful general purpose nodes and large amounts of high-bandwidth memory to support strong scaling applications
- DOE - two different pre-exascale architectures; neither of which fits above description (e.g. power considerations have lowered clock speed)
- Each will have different implementation requirements for achieving good computational performance
- A robust programming and tools environment for portability across architectures does not currently exist

Following slides courtesy of Jim Hack

- from a presentation made in January 2016
- slightly modified
- still relevant

Two Architecture Paths for Future Leadership Systems

Power concerns for large supercomputers are driving the largest systems to either Hybrid or Many-core architectures

Hybrid Multi-Core (e.g. Summit)

- CPU / GPU hybrid systems
- Likely to have multiple CPUs and GPUs per node
- Small number of very powerful nodes
- Expect data movement issues to be much easier than previous systems – coherent shared memory within a node
- Multiple levels of memory – on package, DDR, and non-volatile

Many Core (e.g Intel MIC - NERSC Cori-2)

- 10's of thousands of nodes with millions of cores
- Homogeneous cores
- Multiple levels of memory – on package, DDR, and non-volatile
- Self hosted

Architecture and Performance Portability

Application portability among NERSC, ALCF and OLCF architectures is critical concern of ASCR (and ACME)

- Application developers target wide range of architectures
- Maintaining multiple code version is difficult
- Porting to different architectures is time-consuming
- Many Principal Investigators have allocations on multiple resources
- Applications far outlive any computer system

Improve data locality and thread parallelism

- GPU or many-core optimizations improve performance on all architectures
- Exposed fine grain parallelism transitions more easily between architectures
- Data locality optimized code design also improves portability

Use portable libraries

- Library developers deal with portability challenges
- Many libraries are DOE supported

Need for a common programming model

- Significant work is still necessary for MPI + OpenMP
- All ASCR centers are on the OpenMP standards committee

Encourage portable and flexible software development

- Use open and portable programming models
- Avoid architecture specific models such as Intel TBB, NVIDIA CUDA
- OpenACC still has a ways to go and is just for accelerators
- Use good coding practices: parameterized threading, flexible data structure allocation, task load balancing, etc.

CAAR: Lessons Learned so far…

- **Significant code restructuring of applications to leverage across new architectures**
  - 70-80% of the time spent in code restructuring regardless of the parallel programming model
  - Level of effort is application specific and depends on different factors
    - Code execution profile (compute intensity), code size (LOC), structure of algorithm (parallelism, etc)
  - Some applications need new algorithms
    - Exploit and map to the parallelism available in the node (e.g. accelerator, etc
  - Performance portable libraries are extremely important
- **There is a trend in the increasing complexity of exposing, managing and mapping the parallelism**
  - Code transformations were needed to expose more parallelism
    - S3D: permuted loops across codes to expose more coarse grain parallelism
    - CAM/SE: fused element loops

W. Joubert et al, "Accelerated Application Development: The ORNL Titan Experience", Computers and Electrical Engineering, in press (2016)

CAAR: Lessons Learned so far…

- **Dealing with multiple programming models and languages**
  - Lack of well established standards makes the development difficult
    - E.g. OpenMP4 for accelerators is very new and tools are still in early stages of development
  - Hybrid programming adds more level of complexity for optimizations
    - Each programming model has their own optimization strategies that may be orthogonal.
      - (e.g. MPI optimizations, OpenMP multithreading, accelerator optimizations, etc)
- **Constant adaptation to new architectures**
  - Each new architecture is becoming more complex to program
    - Vector, Clusters, Multi-core, Heterogeneous, Data-Centric (heterogenous memories, burst buffers, etc). More problems thrown to the user.
- **New approaches are needed to produce performance portable codes**
    - Isolate data layouts from business logic of the code (e.g. templated approaches, Kokkos/Raja).
    - Codes software layers to separate what is architecture dependent
    - Codes may need to be adaptive in terms of available parallelism (e.g. # of threads, etc)

-----End of Jim Hack's slides----

DOE Exascale Computing Project and ACME

- MMF replaces deep convection parameterization. Cloud model runs on GPUs Using OpenACC
- GPUs for the ocean
  - Unstructured grid
  - Code needs to be refactored
  - Communication bound
- I/O - Parallel NetCDF

Additional considerations

- I/O impact on performance varies by installation
- Use additional capability for larger ensembles – weak scaling application better suited to many-core computers
- Refactoring core parts of current models will require several years of development and testing in the best of circumstances
- Drivers for hardware innovation are not similar to our problems - ”big data”, AI, etc – data centric applications