Performance and Performance Engineering in Climate Modeling

Patrick H. Worley
Oak Ridge National Laboratory

Petascale Computation for the Geosciences Workshop
April 4-5, 2006
San Diego Supercomputer Center
University of California, San Diego
San Diego, California
Acknowledgements

- This research is sponsored by the Climate Change Research Division, Office of Biological and Environmental Research and by the Mathematical, Information, and Computational Sciences Division, Office of Advanced Scientific Computing Research, both in the Office of Science, U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725.

- This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

- This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

- These slides have been authored by a contractor of the U.S. Government under contract No. DE-AC05-00OR22725. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.
Acknowledgements

- This talk describes the work of many people, at many institutions. In particular, I want to recognize the following groups and projects:
  - CCSM Software Engineering Group (CSEG) at the National Center for Atmospheric Research (NCAR)
    - http://www.ccsm.ucar.edu/cseg/
  - CCSM Software Engineering Working Group
    - http://www.ccsm.ucar.edu/working_groups/Software/
  - SciDAC Project “Collaborative Design and Development of the Community Climate System Model for Terascale Computers”
    - http://www.scidac.org/BER/BER_CCSM.html
  - SciDAC Project “Performance Evaluation Research Center”
    - http://perc.nersc.gov/
Outline

- General Comments
- Community Climate System Model (CCSM)
- Community Atmosphere Model (CAM)
- Parallel Ocean Program (POP)
General Comments to the Performance Community

- As should be clear from the talks at this workshop, performance engineering is an integral part of the development process in climate and weather models, and has been since the inception of numerical modeling in these fields.
- In general, the developers and software engineers are quite sophisticated with regards to performance engineering. They are also limited in manpower and other resources, and are very supportive of people entering the community to help.
- The community is less interested in experimental or niche “solutions”, especially those that require resources from within the community to make work.
- Tools and techniques must be suitable for production codes, not degrading correctness, maintainability, extensibility, or performance portability of codes.
General Code Characteristics

- Community codes
  - Used on many different architectures (by many different people)
  - Used for many different types of investigations (and problem sizes) with widely varying performance characteristics
- Long-lived, with more than one (released) version used in production
- Often subject to rapid evolution, with contributions from many developers, but with centralized control of released versions and strict validation requirements
Performance Idiosyncracies

- Climate science runs consist of ensembles (5-20) typically.
- Simulations vary from a few simulated years, to tens of years, to hundreds of years, to thousands of years, depending on the science issue.
- Cost per horizontal grid point is relatively high due to parameterized physics, in contrast to weather and many CFD applications.
- Throughput requirements constrain problem size, and adding more processors does not eliminate this problem (though it will change the crossover point).
- Changing problem size must be done carefully, to make sure that parameterized physics is still accurate. Scaled speed-up studies are not meaningful. Fixed size analyses for a (small) assortment of problem sizes serve similar purpose.
Community Climate System Model (CCSM)

- Coupled model made up of 5 components
  - Atmosphere model, currently Community Atmosphere Model (CAM3)
  - Ocean model, currently Parallel Ocean Program (modified version of POP1.4.3 in released version, moving to POP2.1 “any day now”)
  - Land Model, currently Community Land Model (CLM3)
  - Sea Ice Model, currently Community Sea Ice Model (CSIM5.0)
  - CCSM Coupler, currently version 6.0
- All components are currently undergoing rapid (internal) development, adding, for example, atmospheric chemistry (atm), biogeochemistry (all components), and dynamic vegetation (Ind). What will be accepted for future released versions will not be known for a few years.
CCSM Hub and Spoke

Ind \(\rightarrow\) cpl \(\rightarrow\) atm \(\rightarrow\) ocn \(\rightarrow\) ice

(slides courtesy of CSEG/NCAR)
CCSM Architecture

- Multi-binary
  - Adding single executable option in near future
- Concurrent component execution
  - Can use more processors than serial component execution design, but may not achieve better performance for a given total processor count.
  - Dependencies between components can lead to significant load imbalances, partially addressed by “appropriate” assignment of processors to components (configuration problem)
  - Plans to introduce serial execution option
- Performance tuning options:
  - number and allocation of processors to components
  - component model tuning options
  - (replacement of components by better performing alternatives)
CCSM Process Flow

CPL sending data to component (state 1) [receive]
CPL receiving data from component (state 3) [send]
Component processing data (state 2) [rec to send]
Component processing (state 4) [send to rec]

(slide courtesy of CSEG/NCAR)
CCSM Performance Characteristics

- Each component has different performance characteristics:
  - Atm: 3D computational grid, currently latitude/longitude/vertical tensor product grid
    - 1D horizontal domain decomposition for spectral dynamics or 2D tensor product horizontal and 2D latitude/vertical domain decomposition for finite volume dynamics
    - 2D arbitrary horizontal domain decomposition for physics
  - Ocn: 3D grid using displaced pole, locally orthogonal horizontal grid; 2D horizontal decomposition using Cartesian blocks.
  - Lnd: 2D horizontal grid (same as Atm); 2D arbitrary horizontal domain decomposition
  - Ice: 2D horizontal grid (same as Ocn); 2D horizontal decomposition using Cartesian blocks
- At current resolutions, either the atmosphere or ocean component model (depending on the science investigation) is the performance bottleneck.
CCSM Performance and Optimization

- Not in this talk, and not by this speaker
  - Interpreting performance data, as currently collected, is complicated
    - CCSM performance exhibits higher variability than its component models (on all systems, though nature of variability is system specific) due to intermodel dependencies? I/O? ???
  - Configuration optimization is likewise complicated
    - Performance models could be a big help here, but would require validated models of all component models as well as intermodel dependencies and I/O.
    - Contact CSEG for more information
  - Stated throughput requirements for Intergovernmental Panel of Climate Change (IPCC) runs is 5 simulated years/day. This was achieved on previous generation platforms for a 1 degree resolution ocean and a 1.4 degree resolution atmosphere.
# Projected Computational Requirements

<table>
<thead>
<tr>
<th>Issue</th>
<th>Motivation</th>
<th>Compute Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatial resolution</td>
<td>Provide regional details</td>
<td>$10^3$-$10^5$</td>
</tr>
<tr>
<td>Model completeness</td>
<td>Add “new” science</td>
<td>$10^2$</td>
</tr>
<tr>
<td>New parameterizations</td>
<td>Upgrade to “better” science</td>
<td>$10^2$</td>
</tr>
<tr>
<td>Run length</td>
<td>Long-term implications</td>
<td>$10^2$</td>
</tr>
<tr>
<td>Ensembles, scenarios</td>
<td>Range of model variability</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total Compute Factor</strong></td>
<td></td>
<td>$10^{10}$-$10^{12}$</td>
</tr>
</tbody>
</table>

---

A Science Based Case for Large-Scale Simulation (SCaLeS), SIAM News, 36(7), 2003 - David Keyes

Establishing a PetaScale Collaboratory for the Geosciences
UCAR/JOSS, May 2005

*(Slide courtesy of John Drake, ORNL)*
Experimental Platforms

- **Cray X1** at ORNL: 128 4-way vector SMP nodes and a 4-D hypercube interconnect. Each processor has 8 64-bit floating point vector units running at 800 MHz.
- **Cray X1E** at ORNL: 256 4-way vector SMP nodes and a 4-D hypercube interconnect. Each processor has 8 64-bit floating point vector units running at 1.13 GHz.
- **Cray XT3** at ORNL: 5294 single processor nodes (2.4 GHz AMD Opteron) and a 3-D torus interconnect.
- **Earth Simulator**: 640 8-way vector SMP nodes and a 640x640 single-stage crossbar interconnect. Each processor has 8 64-bit floating point vector units running at 500 MHz.
- **IBM p575 cluster** at the National Energy Research Supercomputer Center (NERSC): 122 8-way p575 SMP nodes (1.9 GHz POWER5) and an HPS interconnect with 1 two-link network adapter per node.
- **IBM p690 cluster** at ORNL: 27 32-way p690 SMP nodes (1.3 GHz POWER4) and an HPS interconnect with 2 two-link network adapters per node.
- **IBM SP** at NERSC: 184 Nighthawk II 16-way SMP nodes (375MHz POWER3-II) and an SP Switch2 with two network adapters per node.
- **Itanium2 cluster** at Lawrence Livermore National Laboratory (LLNL): 1024 4-way Tiger4 nodes (1.4 GHz Intel Itanium 2) and a Quadrics QsNetII Elan4 interconnect.
- **SGI Altix 3700** at ORNL: 128 2-way SMP nodes and NUMAflex fat-tree interconnect. Each processor is a 1.5 GHz Itanium 2 with a 6 MB L3 cache.
- **SGI Altix 3700 Bx2** at NASA: 1024 2-way SMP nodes and NUMAflex fat-tree interconnect. Each processor is a 1.6 GHz Itanium 2 with a 9 MB L3 cache.
Community Atmosphere Model (CAM)

Atmospheric global circulation model

- Timestepping code with two primary phases per timestep
  - *Dynamics*: advances evolution equations for atmospheric flow
  - *Physics*: approximates subgrid phenomena, such as precipitation, clouds, radiation, turbulent mixing, …

- Multiple options for dynamics:
  - Spectral Eulerian (EUL) dynamical core (*dycore*)
  - Spectral semi-Lagrangian (SLD) dycore
  - Finite-Volume semi-Lagrangian (FV) dycore
  
  all using tensor product *latitude* x *longitude* x *vertical level* grid over the sphere, but not same grid, same placement of variables on grid, or same domain decomposition in parallel implementation

- Separate data structures for dynamics and physics and explicit data movement between them each timestep (in a “coupler”)

- Developed at NCAR, with contributions from DOE and NASA
CAM Performance Portability Goals

1) Maximize single processor performance, e.g.
   a) Optimize memory access patterns
   b) Maximize vectorization or other fine-grain parallelism

2) Minimize parallel overhead, e.g.
   a) Minimize communication costs
   b) Minimize load imbalance
   c) Minimize redundant computation

for
· a range of target systems,
· a range of problem specifications (grid size, physical processes, …)
· a range of processor counts

while preserving maintainability and extensibility.

No optimal solution for all desired (platform,problem,processor count) specifications. Approach: compile-time and runtime optimization options.
CAM Performance Optimization Options

- Physics data structures
  - Index range, dimension declaration
- Physics load balance
  - Variety of load balancing options, with different communication overheads
  - SMP-aware load balancing options
- Communication options
  - MPI protocols (two-sided and one-sided)
  - Co-Array Fortran
  - SHMEM protocols and pt-2-pt implementations or collective communication operators
- OpenMP parallelism
  - Instead of some MPI parallelism
  - In addition to MPI parallelism
- Aspect ratio of dynamics 2D domain decomposition (FV-only)
CAM Performance Experiments

- Spectral Eulerian dycore running on T85L26 computational grid
  - 128x256x26 (latitude by longitude by vertical) grid
  - Current production dynamical core and grid resolution in CCSM
- Finite Volume dynamical code running on 0.5x0.625 degree resolution computation grid (also called “D grid”)
  - 361x576x26 (latitude by longitude by vertical) grid
  - Finite volume dycore is the preferred (required among current options) for atmospheric chemistry due to its conservation properties.
  - Initial production grid size will be 1.9x2.5 degree resolution (96x144x26), so 15 times smaller than D grid.
Physics Data Structures

- All three dynamics use a tensor product longitude-vertical-latitude (plon x pver x plat) computational grid covering the sphere.
- A *vertical column* is a set of grid points of coordinates (i,*,j). In current physics, computation is independent between vertical columns, and tightly coupled within a vertical column.
- The basic data structure in the physics is the *chunk*, an arbitrary collection of vertical columns. Grid points in a chunk are referenced by (local column index, vertical index).
- Define
  
  ncols(j): number of columns allocated to chunk j  
  nchunks: number of chunks  
  begchunk:endchunk: chunk indices assigned to a given process  
  pcols: maximum number of columns allocated to any chunk  
  pcols and pver specified at compile time.
- Arrays declared as (pcols, pver, begchunk:endchunk).
Physics Computational Structure

- Loops structured as
  ```
  do j=begchunk,endchunk
    do k=1,pver
      do i=1,ncols(j)
        (physical parameterizations)
        enddo
      enddo
    enddo
  enddo
  ```
- Inner loop over columns is vectorizable.
- Coarser grain parallelism is exploited over outer loop over chunks, OpenMP can be used to parallelize j loop more.
- Length of inner loop can be adjusted for size of cache or for vector length.
- Columns can be assigned to chunk in order to balance load between chunks or to minimize communication cost of coupling between dynamics and physics.
**pcols Tuning Experiments**

- “Single” processor performance tuning, using MPI only and
  - all processors in SMP node, to include impact of memory contention, or
  - two processors for systems with single processor nodes solving problems with
    - 2048 columns (26 vertical levels, 64 x 32 horizontal grid), so 1024 columns per processor in two processor experiments, or
    - 32768 columns (26 vertical levels, 256 x 128 horizontal grid)
- Columns assigned to chunks both “in order” and to balance load between chunks. (Results similar in both cases.)
- CAM executed for one simulation day and for two simulation days. Difference examined to check for atypical start-up costs.
- Varied pcols (which necessarily varied ncols(j)).
- Physics-only execution times for all processes summed, and results for each pcols value normalized with respect to minimum observed time over all experiments for a given platform.
**pcols Tuning Experiments**

Altix: minimum at pcols = 8
p690: minimum at pcols = 24
XT3: minimum at pcols = 34

pcols <= 4  bad for all systems.

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY
1D vs. 2D Decompositions

- Parallel performance as a function of processor count for different domain decompositions for finite volume dycore:
  - 1D over latitude
  - 2D, defined by a (P/4)x4 virtual processor grid
  - 2D, defined by a (P/7)x7 virtual processor grid
    where first dimension in the processor grid decomposes latitude dimension

- Total runtime for a typical simulation day for 361x576x26 (latitude by longitude by vertical) computational grid, measured on
  - Cray X1E
  - Cray XT3
  - IBM p690 cluster
    and used to calculate computation rates.

- Performance results optimized over communication protocol, pcols, load balancing, and number of OpenMP threads (separately for each data point).
Communication: Finite Volume Dycore

1D Decomposition
64x1

2D Decomposition
16x4

(from L. Oliker, et al, Leading Computational Methods on Scalar and Vector HEC Platforms, in SC05 Proceedings.)
1D vs. 2D: Finite Volume Dycore

- 2D decompositions become superior to 1D when the number of MPI processes used to decompose latitude dimension in 1D exceeds some limit, ~70 on two Cray systems (not using OpenMP). Similarly, (P/7)x7 decomposition begins to outperform (P/4)x4 when the number of MPI processes decomposing latitude in (P/4)x4 exceeds ~70.

- On IBM p690, using OpenMP and the 1D decomposition is superior to using the 2D decomposition up to 672 processors, even when the optimum for 1D uses more threads per process than the optimum for 2D. Note 1D never uses more than 84 MPI processes.
Load Balancing Comparisons

- Parallel performance as a function of processor count for different load balancing schemes for finite volume dycore:
  - Load balancing only within MPI process (no interprocess communication)
  - Load balancing only between pairs of processes (single step, pairwise interprocess communication)
  - Best load balancing (requiring MPI_Alltoallv functionality)
- Total runtime for a typical simulation day, measured on
  - Cray XT3
  - IBM p690 cluster
  and divided by the runtime for the minimum over all load balancing schemes for a given processor.
- Performance results optimized over communication protocol, pcols, domain decomposition, and number of OpenMP threads (separately for each data point).
On the IBM p690, full load balancing is always best, but no load balancing is, at worst, only 7% slower. (For spectral dycore, full load balancing not always optimal.)

On the Cray XT3, full load balancing is usually best, but pairwise exchange load balancing is competitive. No load balancing is usually more than 4% slower, and as much as 12% slower.

The best example of the advantage of full load balancing is on the Cray X1E. It is so much better that it was clear early on in the tuning process and the other load balancing options were “pruned” from the search tree.
Tuning Impact: EUL Dycore / T85L26

- CCM mode - tuning option settings representative of what used in CAM’s predecessor.
- Big win on IBM, primarily due to ability to use OpenMP parallelism in physics when dynamics parallelism exhausted (at approximately 128 processors).
- As vector length for CCM mode is near optimal on the X1E, performance difference is primarily load balancing.
CAM Platform Comparisons: FV Dycore

Performance of the CAM3.1 Atmospheric Model

Finite Volume Dynamics, 361x576x26 benchmark

- Cray X1E (ORNL)
- Cray X1 (ORNL)
- Earth Simulator
- IBM p575 cluster (NERSC)
- Cray XT3 (ORNL)
- Itanium2 cluster (LLNL)
- IBM p690 cluster (ORNL)
- IBM SP (NERSC)

- Earth Simulator results courtesy of D. Parks. LLNL results courtesy of A. Mirin. SP results courtesy of M. Wehner. Maximum number of MPI processes is 960. IBM systems and Earth Simulator use OpenMP to increase scalability.

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY
CAM Performance Diagnosis

Performance of the CAM3.1 Atmospheric Model

Finite Volume Dynamics, 361x576x26 benchmark

- Dynamics and physics scaling very similarly on XT3.
- On X1E, physics scaling much better. (Dynamics best performance at 448 processors.)
At large scale, dynamics performance on p575 similar to performance on X1E. However, p575 uses OpenMP and X1E does not in these experiments. P575 using a coarser dynamics domain decomposition.
Known Limits to FV Scalability

1. Requirement that at least 3 latitudes and 3 vertical levels be present in each “block” of domain decomposition within FV dycore. For D grid with 26 vertical levels, limit is 120 x 8 processor grid, or 960 MPI processes.

2. Polar filter introduces load imbalances, especially on vector systems (because the small number of short FFTs do not vectorize well).

3. Physical parameterizations can use many more processors, but is currently limited to the same number of MPI processes as the dynamical core. OpenMP can be used to assign more processors to physics than to dynamics, mitigating this to some degree. There is also some OpenMP parallelism available within the dynamics.

4. On vector systems, additional parallelism in physics is of limited utility, as vector length drop below 220 for more than 960 processors (and drops below 110 in radiation routines).
Proposed Solutions and New Issues

1. Dynamics Scaling?
   · Using alternative (cubed sphere?) computational grid.

2. Physics Scaling?
   · Allow different number of MPI processes in dynamics and in physics (generalizing current OpenMP approach to pure MPI codes).

3. Introduction of atmospheric chemistry will add many new tracers, adding work to both dynamics and to physics.
   · Parallelize advection over species?
   · Parallelize chemistry over species?
   · Use 3D decomposition for chemistry?

4. Increasing model resolution will exacerbate the current I/O bottlenecks and memory impact of the (few) remaining global arrays.
   · Parallel I/O will allow both of these problems to be addressed, hopefully successfully.
Ongoing CAM Work

1. Scaling Studies with FV Dycore
   - 1 x 1.25 degree horizontal resolution (C grid)
   - 0.5 x 0.625 degree horizontal resolution (D grid)
   - 0.25 x 0.3125 degree horizontal resolution (E grid)

2. Scaling Studies with EUL Dycore
   - T42L26
   - T85L26
   - T170L26
   On Cray X1E, Cray XT3, Earth Simulator, IBM POWER5 cluster, NEC SX-8, SGI Altix, …(and a number of other Opteron and Itanium2 clusters)

3. Constructing a CAM “Performance Model”

4. Repeating / Updating 1-3 as add atmospheric chemistry and other new physical processes, and as add additional parallelism into physics.
Source Material


Parallel Ocean Program (POP)

- Developed at Los Alamos National Laboratory. Used for high resolution studies and as the ocean component in the Community Climate System Model (CCSM)
- Two primary computational phases
  - Baroclinic: 3D with limited nearest-neighbor communication; scales well.
  - Barotropic: dominated by solution of 2D implicit system using conjugate gradient solves; scales poorly.
- Domain decomposition determined by grid size and 2D virtual processor grid.
POP Performance Options

- Modified version of POP 1.4.3 used in CCSM. Will be replaced by POP2.1 in near future.
- POP1.4.3 is a pure MPI code (i.e., does not use SMP parallelism). Performance optimizations are limited to vector vs. nonvector version, shape of 2D virtual processor grid, choice of messaging layer (MPI, SHMEM, Co-Array Fortran), and two MPI protocol options.
- POP2.1 additionally supports subblocks (for improved cache locality and load balancing), OpenMP, using fewer processors in the barotropic phase, and an alternative formulation of the conjugate gradient solver requiring fewer inner products.
POP Experiment Particulars

- Los Alamos National Laboratory version of POP1.4.3 with vectorization and parallel algorithm tuning options.
- Two fixed size benchmark problems
  - One degree horizontal grid (“by one” or “x1”) of size 320x384x40 (current CCSM production resolution)
  - Tenth degree horizontal grid of size 3600x2400x40 both with very little I/O.
- Results for a given processor count are the best observed over all applicable processor grids.
Implementation Comparison: Cray X1

Comparing performance of nonvector and vector, and of MPI and hybrid MPI/Co-Array Fortran, implementations.
Vector version includes MPI optimizations such as NOT using MPI derived datatypes. Scalability is limited when using only MPI.
These are old data. Performance of MPI version is better now due to new implementation of MPI_Allreduce, as indicated by single 128 processor data point, but Co-Array Fortran is still useful in the halo update used in the code.
POP Platform Comparisons: 1 degree

- Earth Simulator results courtesy of Y. Yoshida. BG/L results courtesy of R. Loy. P575 results courtesy of J. White. X1E performance is excellent when using Co-Array Fortran.
Examining time spent in baroclinic and barotropic phases for the X1E, with and without Co-Array Fortran. Co-Array Fortran was used to reimplement halo update in the barotropic conjugate gradient solver.
POP Platform Comparisons: Barotropic

- Comparing time spent in barotropic phases. Even MPI-only version of POP on X1E has superior barotropic performance compared to MPI-only experiments on other platforms, due to high performance of MPI_Allreduce.
POP Platform Comparisons: Baroclinic

- Comparing time spent in baroclinic phases. Scaling excellent on all platforms, though slowing some on vector systems at large processor counts, due to … shorter vector lengths? communication overhead?

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY
POP Platform Comparisons: 0.1 degree

Earth Simulator results courtesy of Y. Yoshida.

X1, X1E, and XT3 experiments used different horizontal grid (but same number of grid points), increasing number of CG iterations in barotropic phase. Performance data normalized as if same number of iterations occurred as in IBM and Earth Simulator runs.

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY
POP Platform Comparisons: 0.1 degree

- Unnormalized results for X1, X1E, and XT3.
- More CG iterations degrades XT3 performance more than it degrades X1E performance.
POP Performance Analysis: 0.1 degree

- Unnormalized results for X1E and XT3.
- Barotropic performance will limit further scalability on the XT3.
- Note superlinear speedup in baroclinic phase.
POP Performance Summary

- Small communication latency was required to achieve good scalability for both POP benchmark problems. New POP2.1 conjugate gradient solver will decrease this sensitivity, but not eliminate it.
- Memory performance limiting overall performance on 0.1 degree benchmark, but continuing to scale well to very large processor counts. Cache blocking option in POP2.1 may be useful on X1E as well as on non-vector systems.
- Good performance on the X1 was originally achieved by using Co-Array Fortran to implement two collectives: allreduce and halo update. The Co-Array Fortran implementation of allreduce is no longer necessary as the MPI_Allreduce in the supplied MPI library is as efficient as our hand-coded version.
- Achieving good performance required vectorization, MPI optimization, and exploiting alternative programming paradigms, but it was worthwhile.