Performance and Performance Portability of the Community Atmospheric Model

Patrick H. Worley
Oak Ridge National Laboratory

2005 LACSI Symposium

Workshop on Performance and Productivity of

Extreme-Scale Parallel Systems

October 11, 2005

Eldorado Hotel

Santa Fe, New Mexico

Acknowledgements

- Research sponsored by the Atmospheric and Climate Research Division and the Office of Mathematical, Information, and Computational Sciences, Office of Science, U.S. Department of Energy under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.
- 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
- Oak Ridge National Laboratory is managed by UT-Battelle, LLC for the United States Department of Energy under Contract No. DE-AC05-00OR22725.

Source Material

P. Worley and J. Drake, *Performance Portability in the Physical Parameterizations of the Community Atmospheric Model*, International Journal for High Performance Computer Applications, 19(3), August 2005, pp. 187-201.

A. Mirin and W. Sawyer, *A Scalable Implementation of a Finite-Volume Dynamical Core in the Community Atmosphere Model*, International Journal for High Performance Computer Applications, 19(3), August 2005, pp. 203-212.

W. Putman, S-J. Lin, and B-W. Shen, *Cross-Platform Performance of a Portable Communication Module and the NASA Finite Volume General Circulation Model*, International Journal for High Performance Computer Applications, 19(3), August 2005, pp. 213-223.

L. Oliker, J. Carter, M. Wehner, A. Canning, S. Ethier, A. Mirin, G. Bala, D. Parks, P. Worley, S. Kitawaki, and Y. Tsuda, *Leading Computational Methods on Scalar and Vector HEC Platforms*, in Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking and Storage (SC05), Seattle, WA, November 12-18, 2005.

Community Atmospheric 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)
  - Spectral semi-Lagrangian (SLD)
  - Finite-Volume semi-Lagrangian (FV)

  all using tensor product *longitude* x *latitude* 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
- http://www.ccsm.ucar.edu/models/atm-cam/

Performance 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.

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 two-sided protocols
  - MPI-2 one-sided protocols
  - Co-Array Fortran
  - SHMEM protocols

  and point-to-point implementations or MPI collective communication operators.
- OpenMP parallelism
  - Instead of some MPI parallelism
  - In addition to MPI parallelism
- Aspect ratio of dynamics 2D domain decomposition (FV-only)

Original Physics Data Structures

- Define
  - plon: the number of longitudes
  - pver: the number of vertical levels
  - plat: the number of latitudes
  - beglat:endlat: latitude indices assigned to a given process
  - plond = plon + 1 + 2*nexpt, where nexpt defines the size of a *halo* used in the semi-Lagrangian advection algorithm

  All but beglat and endlat specified at compile time.
- Arrays declared as (plond, pver, beglat:endlat).

  (Same as in spectral dynamics, predating physics/dynamics decoupling.)

Define a *vertical column* to be set of grid points of coordinates (i,*,j). In current physics, computation is independent between vertical columns, and tightly coupled within a vertical column.

Original Physics Data Structures

- Loops structured as

```
do j=beglat, endlat
    do k=1,pver
        do i=1,plon
            (physical parameterizations)
        enddo
    enddo
enddo
```

- Inner loop over longitude is vectorizable.
- Coarser grain parallelism exploited over outer loop over latitude. OpenMP can be used to parallelize j loop even further.

Define a *latitude slice* to be set of grid points of coordinates (*,*,j). In the original formulation, a latitude slice was the basic computational unit.

New Physics Data Structures

- Replace *latitude slice* as basic computational unit by an arbitrary collection of columns called a *chunk*.
- Define

  `ncols(j)`: number of columns allocated to chunk `j`

  `pver`: number of vertical levels

  `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`).

New Physics Data Structures

- Loops structured as

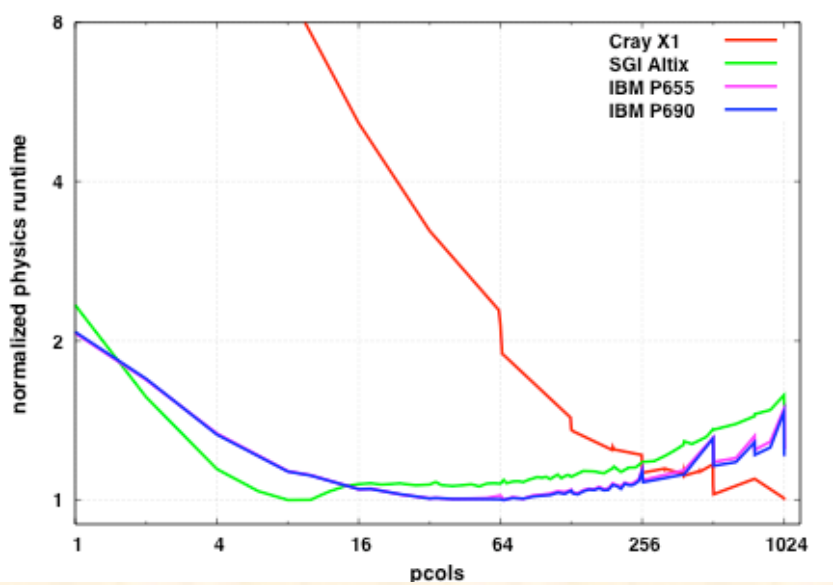
```
do j=begchunk,endchunk
    do k=1,pver
        do i=1,ncols(j)
            (physical parameterizations)
        enddo
    enddo
enddo
```

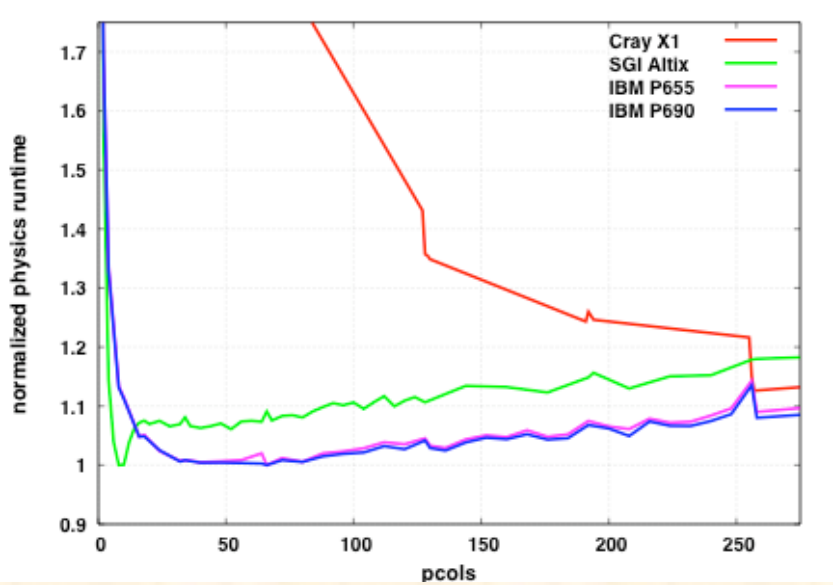
- Inner loop over columns is (still) vectorizable.
- Coarser grain parallelism is (still) exploited over outer loop over chunks, OpenMP can (again) be used to parallize j loop more.
- Length of inner loop can be adjusted for size of cache or for vector length.
- Columns can be assigned to chunk in order to balance load between chunks or to minimize communication cost of coupling between dynamics and physics.

pcols Tuning Experiments

- "Single" processor performance tuning, using two processors (chosen so that the two processors do not share cache).
- 2048 columns (26 vertical levels, 64 x 32 horizontal grid), so 1024 columns per processor
- 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 two processes summed, and results for each `pcols` value normalized with respect to minimum observed time over all experiments for a given platform.
- For SMPs, experiment repeated using all processors in SMP node (and larger total number of columns), looking for impact of memory contention.

pcols Tuning Experiments





Altix: minimum at pcols = 8

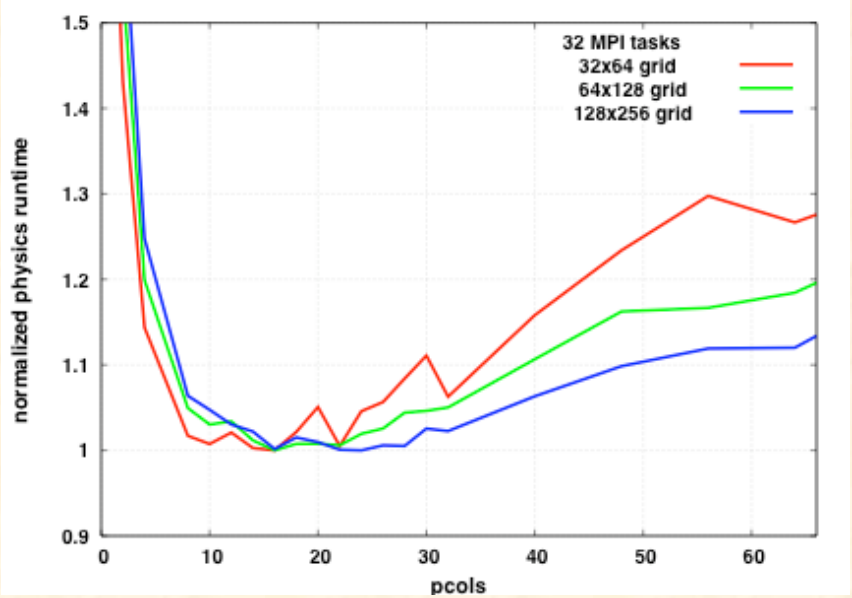
P690: minimum at pcols = 56

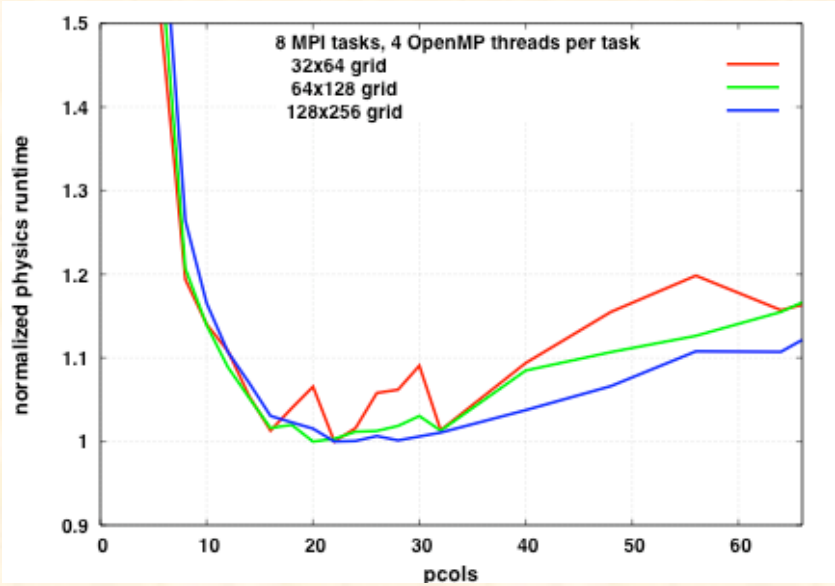
X1: minimum at pcols >= 1026

P655: minimum at pcols = 66

pcols <= 4 bad for all systems.

pcols Tuning Experiments





Altix, X1: no change in optimal `pcols`

P690: minimum between 16 and 32 for all experiments. (Memory contention does play a role in determining optimal `pcols`.)

Load Imbalance Characteristics

From computational scientists:

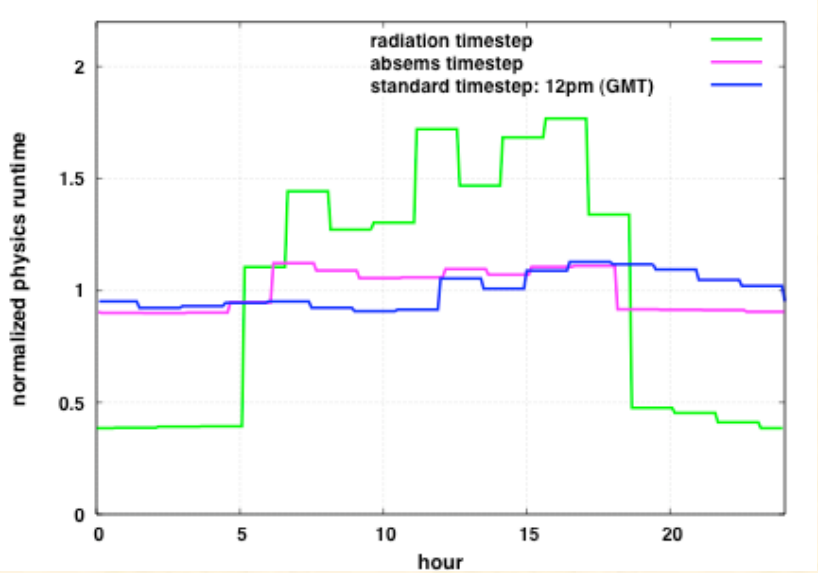
- Cost per column in physics is not uniform, but is only a function of location and simulation time (i.e., not a function of the solution).
- Most physical processes are calculated every timestep, but short-wave radiation, long-wave radiation, and water vapor absorptivity and emissivity are calculated less often. Default is to calculate both types of radiation once every simulation hour (independent of timestep length), and absorptivity and emissivity once every 12 simulation hours.
- Timesteps including radiation calculations are more expensive, but primarily where it is daylight.
- Timesteps including absorptivity and emissivity calculations are even more expensive, but the additional cost is uniform.

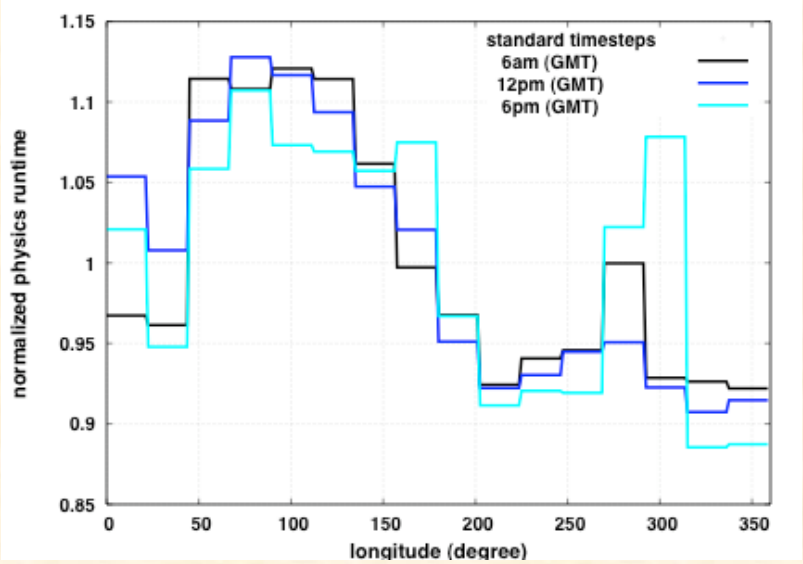
Load Imbalance Characteristics

Validating characterization on IBM p690 cluster

- 256 x 128 x 26 grid on 32 processors using `pcols = 16`
- Columns assigned to chunks consecutively, by longitude, and no chunk contained columns from multiple latitudes.
- Runtime for individual column assumed to be 1/16 of runtime for entire chunk.
- Runtime measured for each chunk from a latitude line near the equator for a single radiation timestep, for a single absorptivity and emissivity timestep (*absems*), and for three different standard timesteps. Runtimes normalized by average runtime for these chunks at the given timestep. (Note that all of these chunks in following figures were assigned to the same processor.)

Load Imbalance Characteristics





- Strong diurnal cycle signal in radiation timestep.
- Weak geographic signal in standard timestep.

	standard	radiation	absems
steps/sim. day	120	22	2
secs/step	.078	.270	1.63
secs/sim. day	9.36	5.94	3.26

Load Balancing

- Goal
  - Define chunks so that each has the same physics runtime.
  - Assign equal number of chunks to each thread.
- Approach
  - Assign same number of columns to each chunk.
  - Assign columns to chunks so that each chunk has the same number of “day” columns and “night” columns at all times.
  - Do not assign geographically neighboring columns to the same chunk.
- Problem: Dynamics domain decomposition
  - Spectral: latitude slices
  - FV: 2D longitude/latitude blocks

So opposite of proposed physics domain decomposition, requiring significant interprocess communication in physics/dynamics coupler.

Load Balancing Options

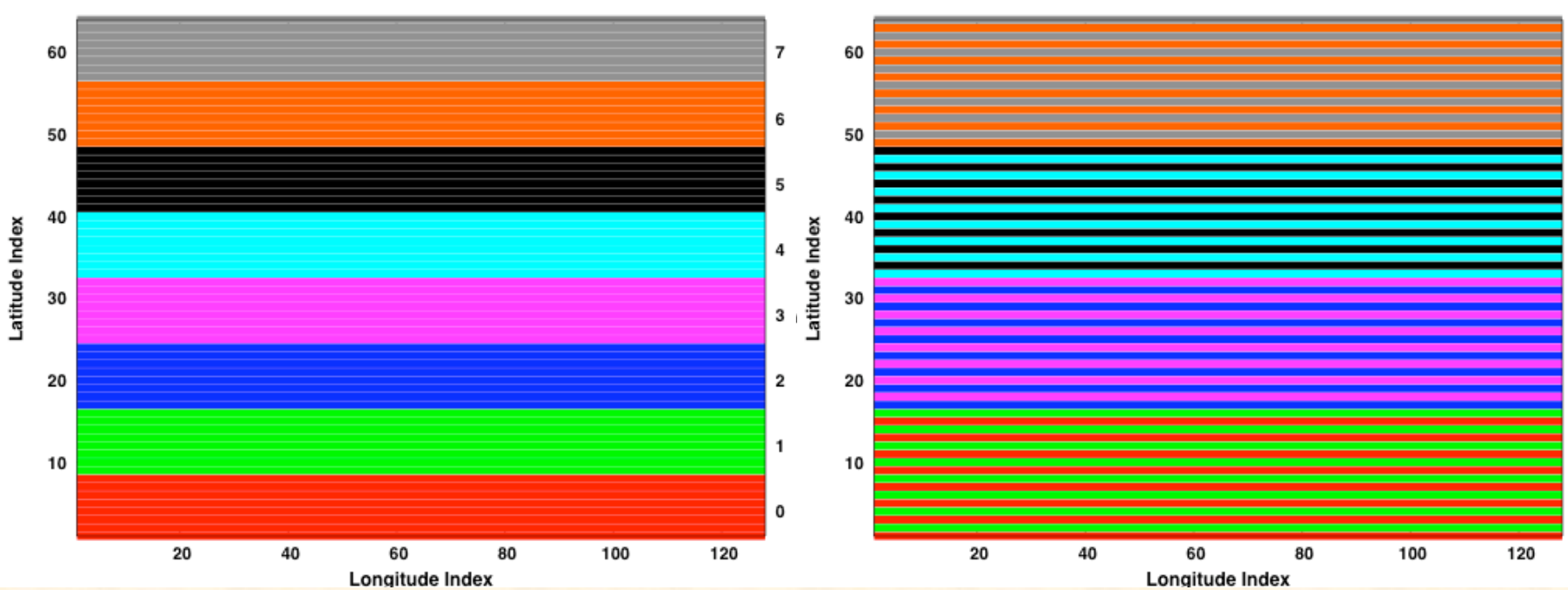
- +2:
  - Assign columns to chunks in pairs, where partner for a given column defined by reflecting latitude coordinate over the equator and adding 180 degrees to longitude coordinate.
  - Assign column pairs (ordered by longitude-latitude indices of first column in pair) to chunks in a round robin fashion.
  - Assign same number of chunks to each thread, attempting to minimize MPI overhead in physics/dynamics coupling.
- +1:
  - Same as option 2, except that pair creation and pair assignment restricted so that physics and dynamics domain decompositions assign columns to same SMP nodes.
  - If optimal partner not in same SMP, use column at same latitude, but 180 degrees different in longitude. If second choice also not available, assign column to chunk without a partner.
  - This approach does not require inter-SMP node communication.

Load Balancing Options

- +3:
  - Same as option 1 except that, instead of SMP nodes, pair creation and pair assignment is restricted to two process pairs.
  - Process pairs chosen to maximize possibility that can create optimal column pairs.
  - All interprocess communication is between processes in same pair.
- +0:
  - Same as option 1 except that pair creation and pair assignment restricted to a single process, i.e. the same domain decomposition is used for both physics and dynamics (at the process level). Column pairs are still assigned to chunks in a round robin fashion, and OpenMP parallelism over local chunks will be different than OpenMP parallelism over latitude slices or longitude-latitude blocks.
- -1:
  - Same as option 0 except that columns are assigned individually consecutively to chunks using same ordering as in dynamics blocks or slices.

Load Balancing Examples

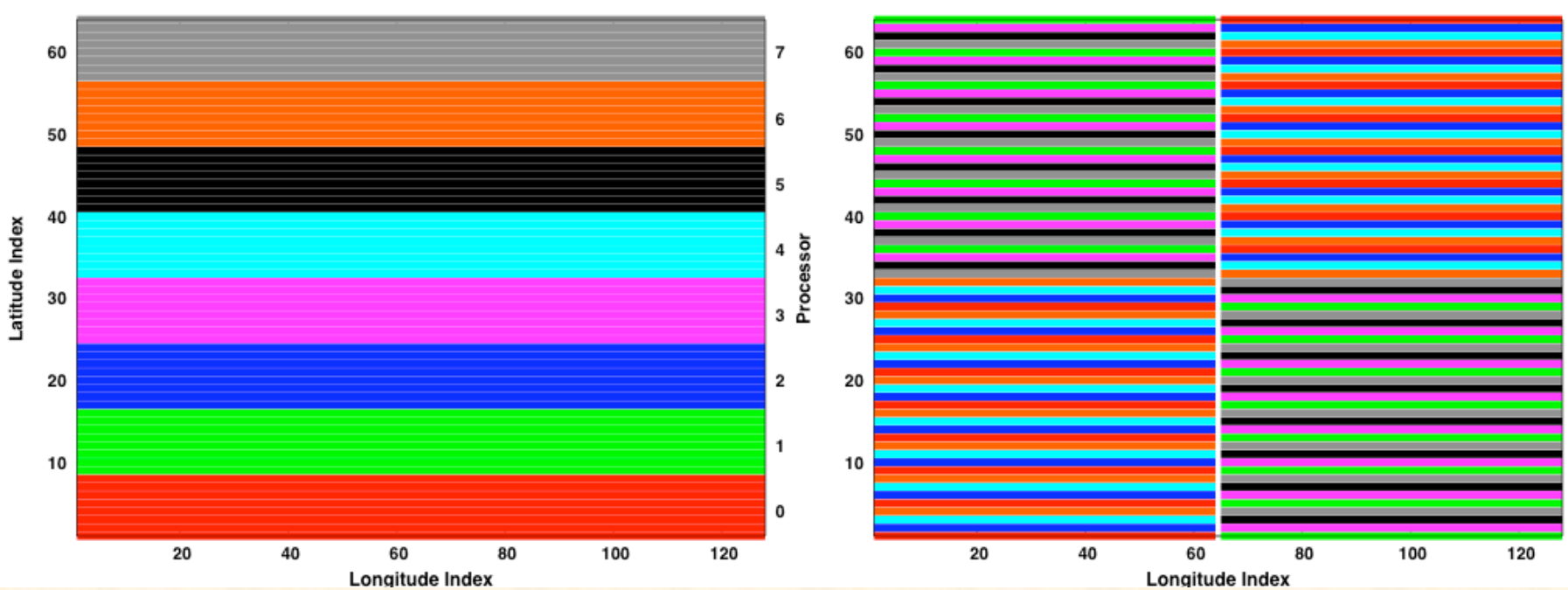
- 128x64 horizontal grid
- 4 nodes of 2-processor SMP nodes
- Latitude slice-based domain decomposition
- Color denotes processor that column assigned to.



Option 0 **Option 1**

Load Balancing Examples

- 128x64 horizontal grid
- 4 nodes of 2-processor SMP nodes
- Latitude slice-based domain decomposition
- Color denotes processor that column assigned to.

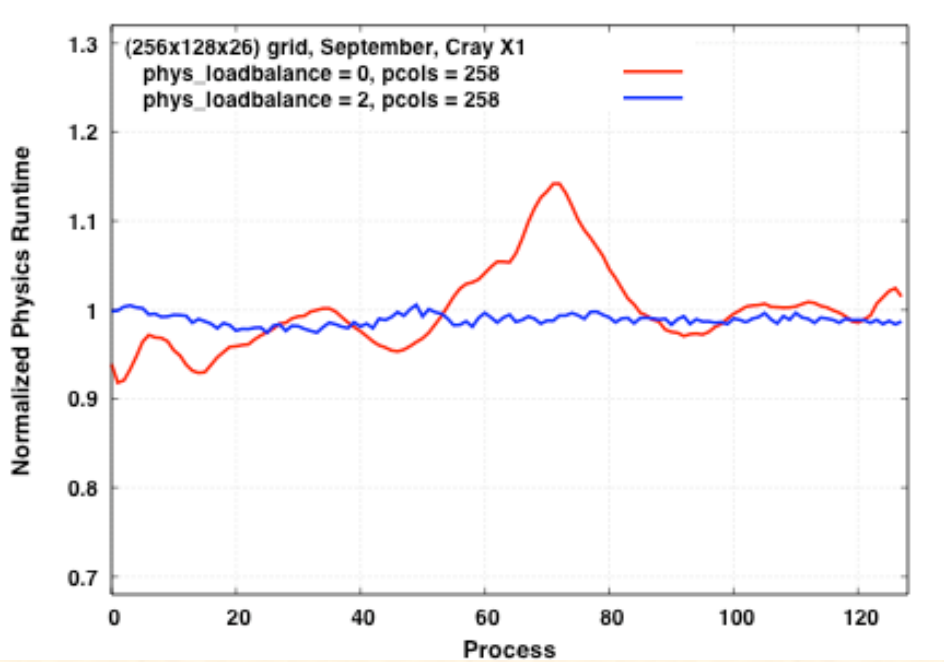


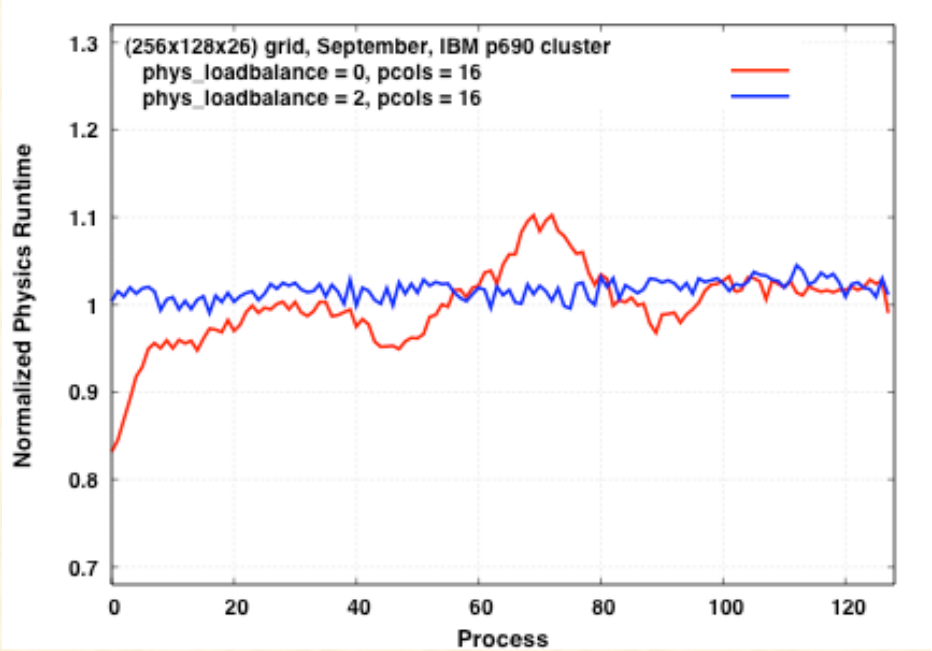
Option 0 **Option 2**

Load Balancing Experiments

- 256 x 128 x 26 problem
- Two runs each of 30 simulation days, one in September and one in January
- 128 processors
  - 4 32-way IBM p690 nodes with HPS interconnect
  - 32 4-way Cray X1 nodes
- Load balancing options 0 and 2
- `pcols` = `16` or 258 for the p690 cluster
- `pcols` = `258` for the X1
- Physics runtime for each process normalized by average over all processes for option 0 (and `pcols=16` for p690 cluster)
- NOT including communication overhead in implementing option 2.

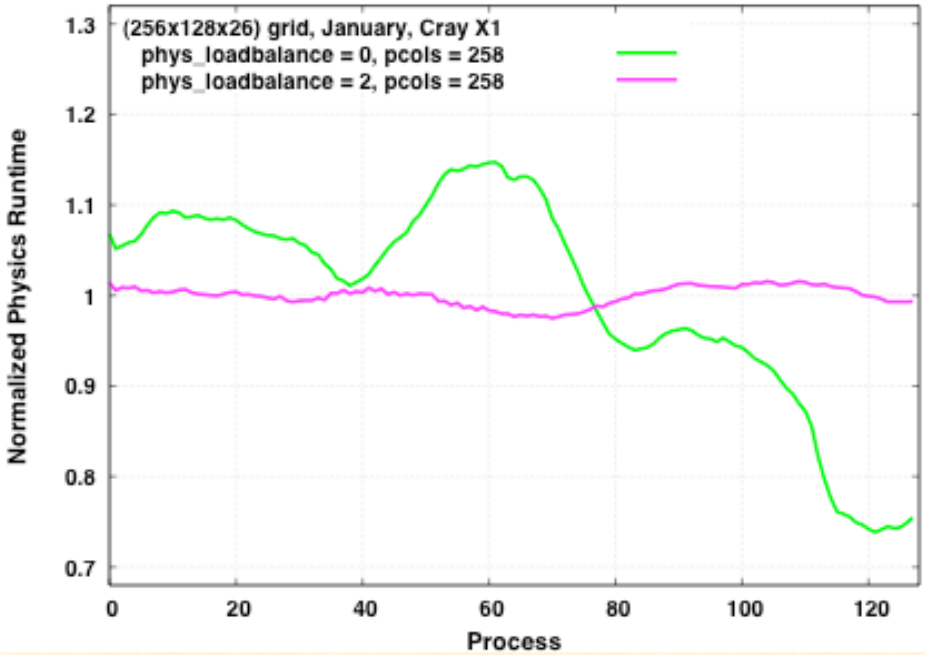
September Load Balancing Expts.

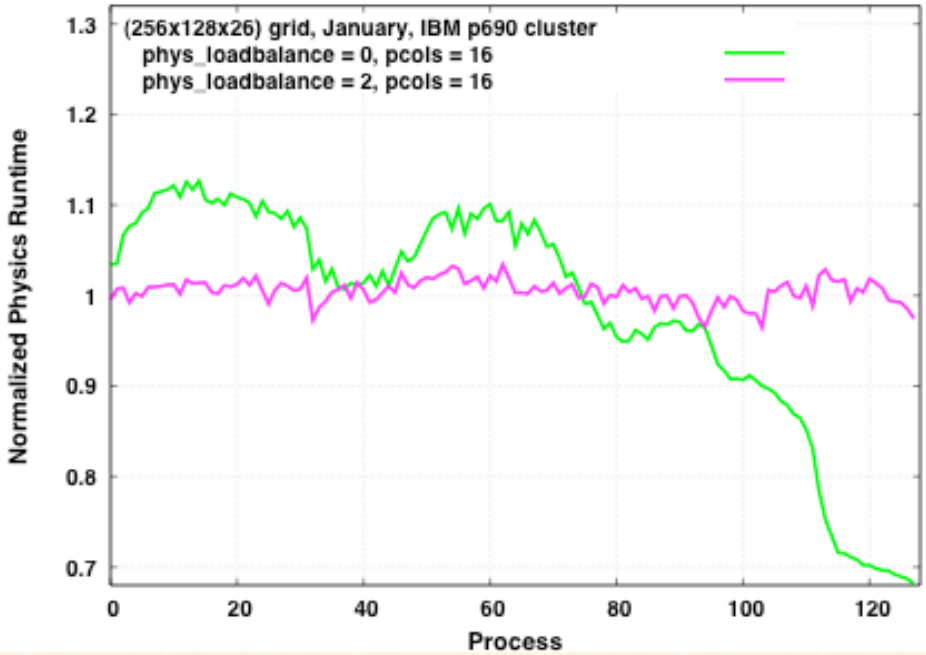




- Option 0 represents no load balancing since not using OpenMP.
- Option 2 does an excellent job balancing load.
- Despite very different processor and memory architectures, load balance signatures very similar.

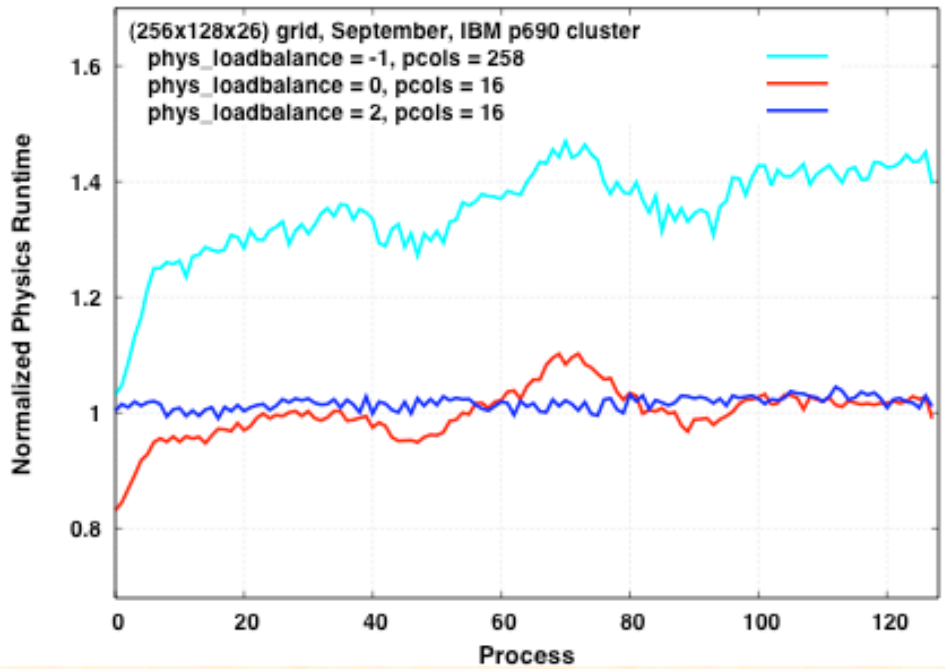
January Load Balancing Expts.

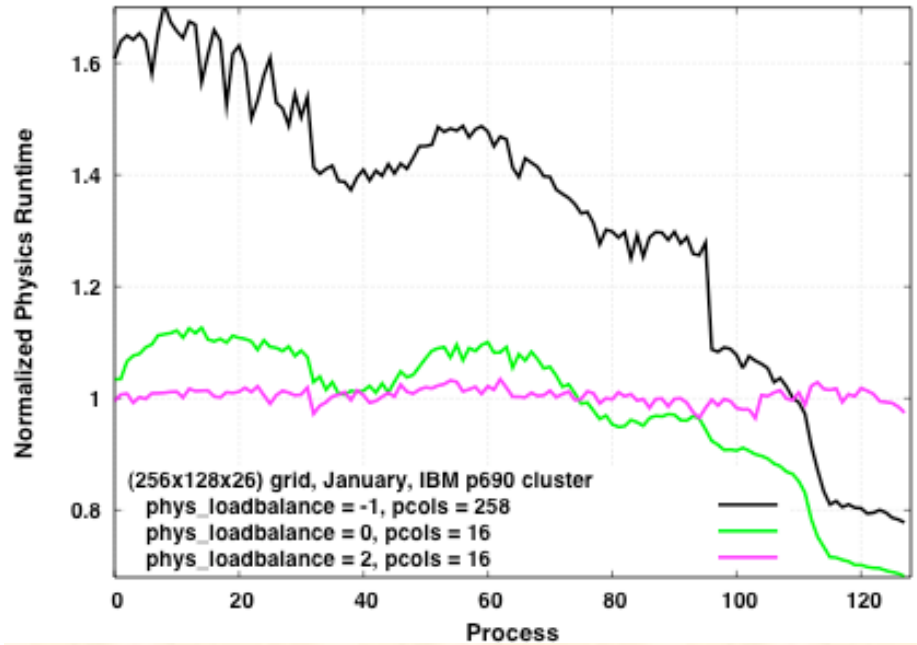




- Option 0 load imbalance worse for January (due to impact of season on upper latitudes)
- Option 2 still does an excellent job balancing load.
- Load balance signatures again similar on two systems.

IBM Load Balancing Expts.





- Adding data for option 0 when using original latitude slice-based domain decomposition, i.e. a single chunk per process with `pcols=258`. This indicates that both chunk size and load balance are important to performance on the IBM system.

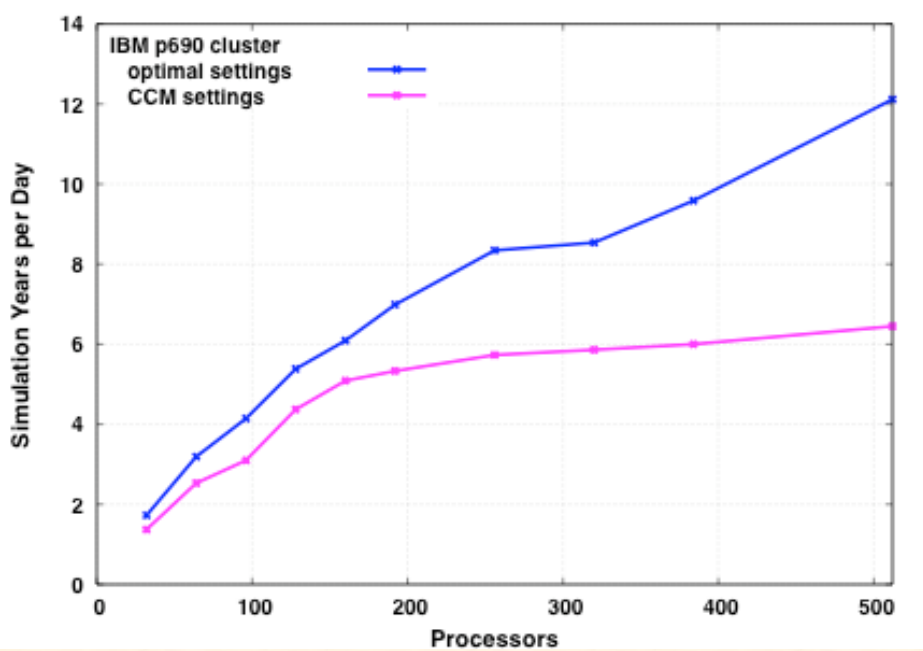
Summary So Far

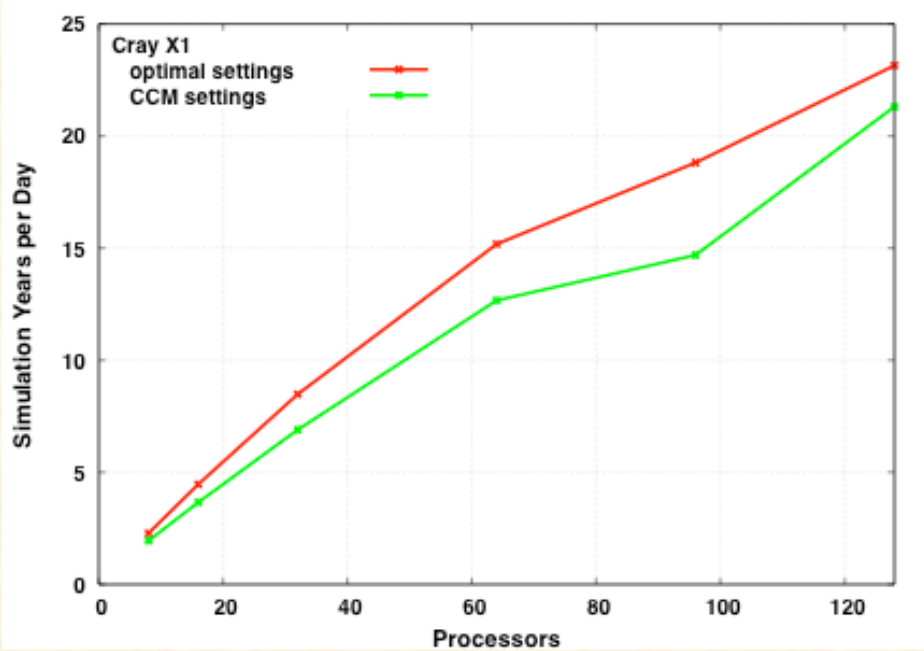
- Tunable "chunk'ed" physics data structure allows
  - Single processor optimization by improving memory access patterns and/or vectorization
  - Load balancing
- Other impacts
  - Setting `pcols` "small" increases number of chunks and exposes more parallelism in the physics, typically more than is available in the dynamics (or more than is worth exploiting due to communication overhead). Using OpenMP allows use of more processors in physics than in dynamics, increasing scalability when physics is more expensive than dynamics (which is typically the case)
- Other optimizations
  - Optimal load balancing option is a function of communication overhead. Tuning communication important first step before deciding on load balancing option.

Benchmarking Experiments

- Spectral Eulerian dynamics
- 256 x 128 x 26 problem
  - CAM resolution when used in Community Climate System Model production runs.
- Optimized over `pcols`, load balancing options, number of OpenMP threads per process for a given total number of processors, and communication protocols for each system and processor count.
- Also run in "original" formulation, `pcols=256`, `258`, and `259` with load balancing option -1. Still optimized over number of OpenMP threads per process. Referred to as "CCM" settings, where CCM is the CAM predecessor.
- Optimal setting determined using one and two simulation day experiments.
- Benchmark runs are 30 simulation days in September .

Benchmark Results: p690 and X1





- Big win on IBM, primarily due to ability to use OpenMP parallelism in physics when dynamics parallelism exhausted (at approximately 128 processors).
- Ability to increase vector length longer than 258 for smaller processor counts helps on X1, as does load balancing. OpenMP does not help.

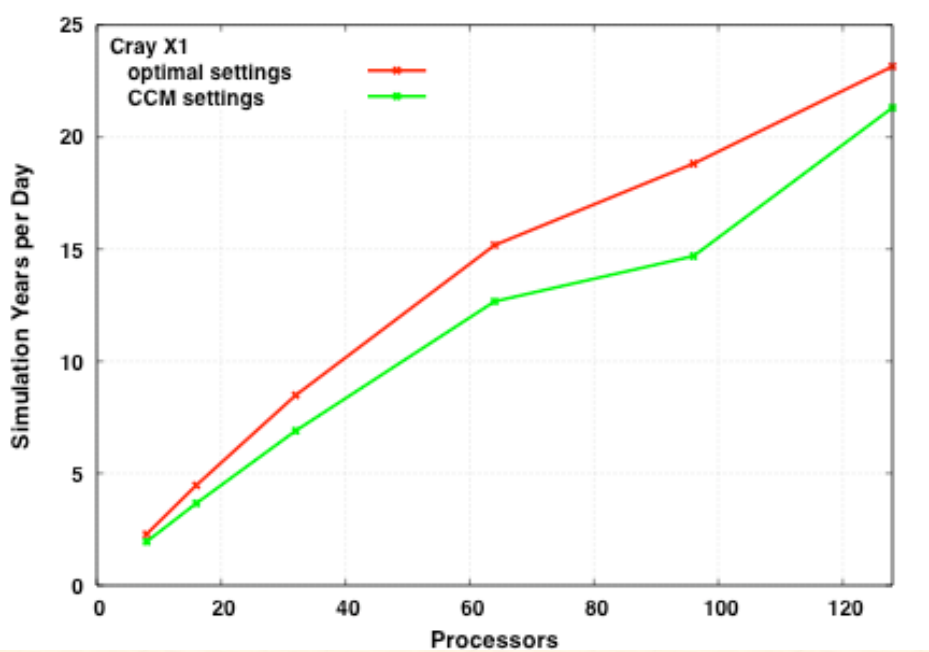
Benchmark Details: IBM

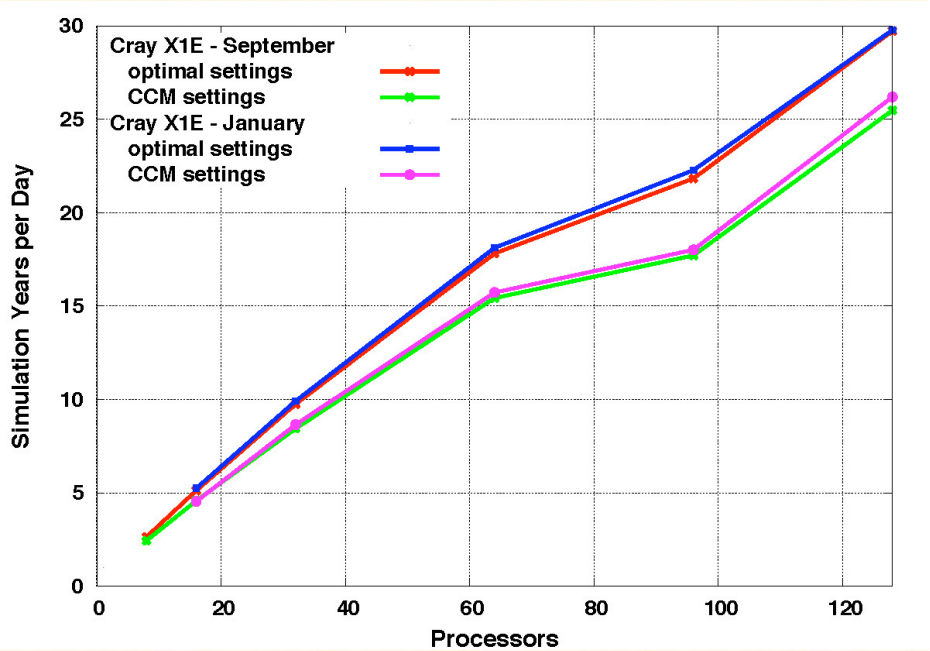
Proc.	MPI proc.s	OpenMP threads	pcols	load balance	MPI collective	improv. vs. CCM
32	32	1	16	2	yes	28%
64	64	1	16	3	no	28%
96	96	1	24	2	yes	36%
128	128	1	16	3	no	25%
160	40	4	24	2	yes	21%
192	48	4	24	3	no	33%
256	128	2	16	3	no	47%
320	40	8	32	0	-	48%
384	48	8	24	3	no	62%
512	128	4	16	3	no	91%

Benchmark Details: Cray

Proc.	MPI proc.s	OpenMP threads	pcols	load balance	MPI collective	improv. vs. CCM
8	8	1	1026	2	yes	19%
16	16	1	1026	2	yes	23%
32	32	1	1026	2	yes	24%
64	64	1	514	2	yes	21%
96	96	1	514	2	yes	30%
128	128	1	258	2	yes	10%

Benchmark Results: X1 and X1E





- At 128 processors, vector length is the same in both CCM and optimal settings, so load balancing is primary difference. Load balancing is even more useful on the X1E.
- Results for January almost identical to results for September.

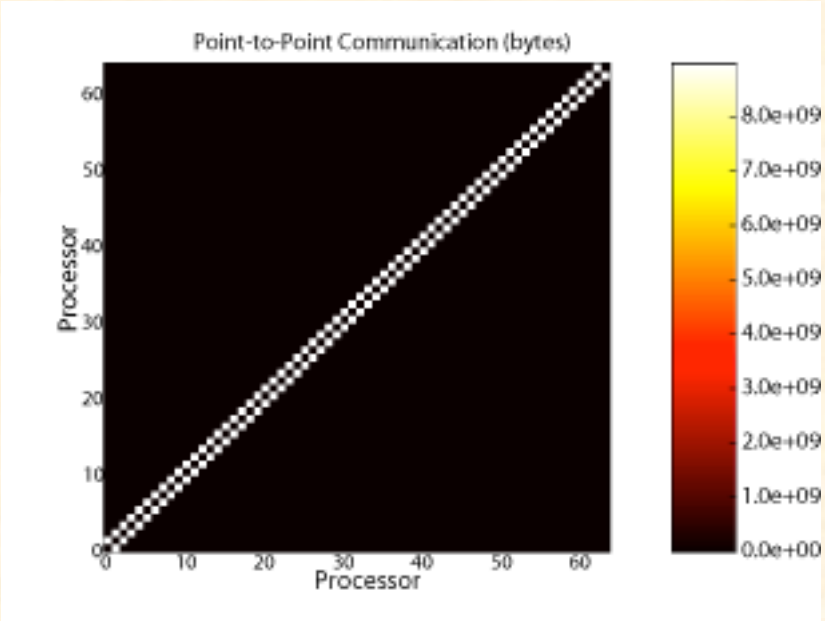
Summary and Future

- Compile-time and runtime options allowed not only good performance improvements on individual systems, but allowed the *same* code to be optimized on significantly different architectures and for a large range of processor counts.
- What's next:
  - More aggressive optimization of communication options in dynamics.
  - Significant increase in cost of physics coming (atmospheric chemistry), changing nature of load imbalances. New analysis will be required, and static load balancing schemes may no longer be sufficient.
  - New dynamical cores are being considered that do not use longitude-latitude grids.
  - Supporting mesh refinement.
  - Adoption of ESMF and/or CCA interfaces.
  - Dealing with increasingly large number of optimization options, especially in context of full CCSM. Exploiting models to prune optimization tree? How about testing?

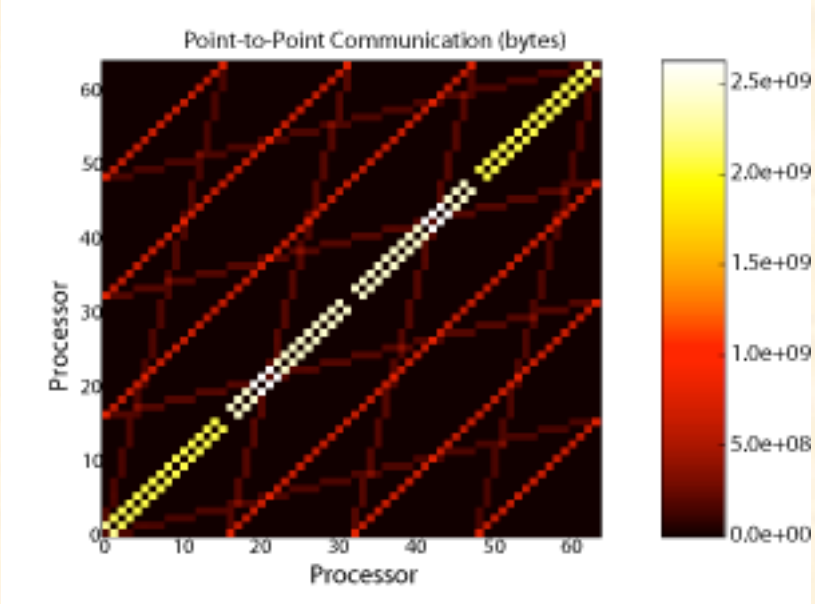
Addendum: FV Domain decomposition

- 1D domain decompositions seriously limit exploitable parallelism in the dynamics.
- FV supports 2D domain decompositions, both latitude-longitude and latitude-vertical (each in different phases of the code). 2D decomposition comes at cost of higher communication overhead, primarily due to transpose between latitude-vertical and latitude-longitude decompositions, so is often not used until parallelism in 1D decomposition is exhausted.
- Experimental results for 576 x 361 x 26 problem. FV requires at least 3 latitudes per process, so, in a 1D decomposition, can not use more than 120 MPI processes in the dynamics for this problem. 2D decomposition decomposes over vertical, so can't use more than 26, and practical limit appears to be 7 on current systems, but this still increases maximum to 840. OpenMP parallelism can be used with remaining vertical parallelism.

FV Communication Patterns

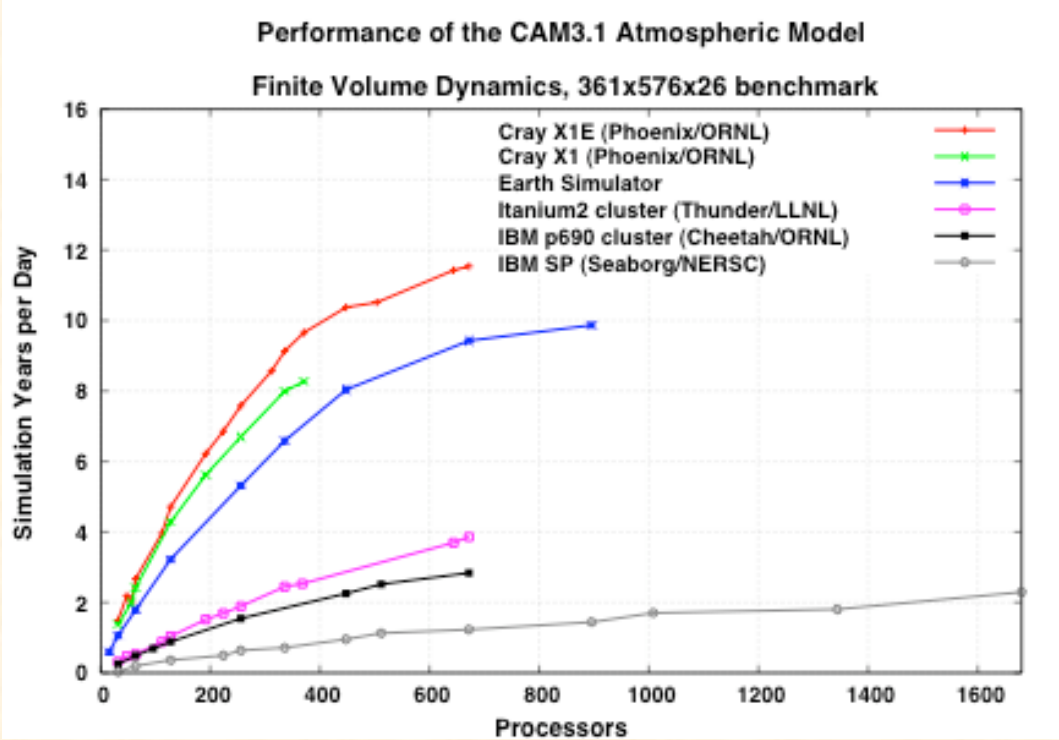


1D Decomposition
64x1



2D Decomposition
16x4

FV Benchmark



- 2D decomposition allowed much larger processor counts, even on systems like the X1 and X1E where OpenMP was not used.
- Also included all physics optimizations described earlier.