The Performance Evolution of the Parallel Ocean Program on the Cray X1

Patrick H. Worley
Oak Ridge National Laboratory

John Levesque
Cray Inc.

46th Cray User Group Conference
May 18, 2003
Knoxville Marriott
Knoxville, TN

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.

Phoenix

Cray X1 with 64 SMP nodes

- 4 Multi-Streaming Processors (MSP) per node
- 4 Single Streaming Processors (SSP) per MSP
- Two 32-stage 64-bit wide vector units running at 800 MHz and one 2-way superscalar unit running at 400 MHz per SSP
- 2 MB Ecache per MSP
- 16 GB of memory per node

for a total of 256 processors (MSPs), 1024 GB of memory , and 3200 GF/s peak performance.

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)
- Ported to the Earth Simulator by Dr. Yoshikatsu Yoshida of the Central Research Institute of Electric Power Industry (CRIEPI).
- Initial port to the Cray X1 by John Levesque of Cray, using Co-Array Fortran for conjugate gradient solver.
- X1 and Earth Simulator ports merged and modified by Pat Worley and Trey White of Oak Ridge National Laboratory.
- POP used (and further optimized) over the past year to test OS scalability and MPI performance and as a Co-Array Fortran algorithm testbed.

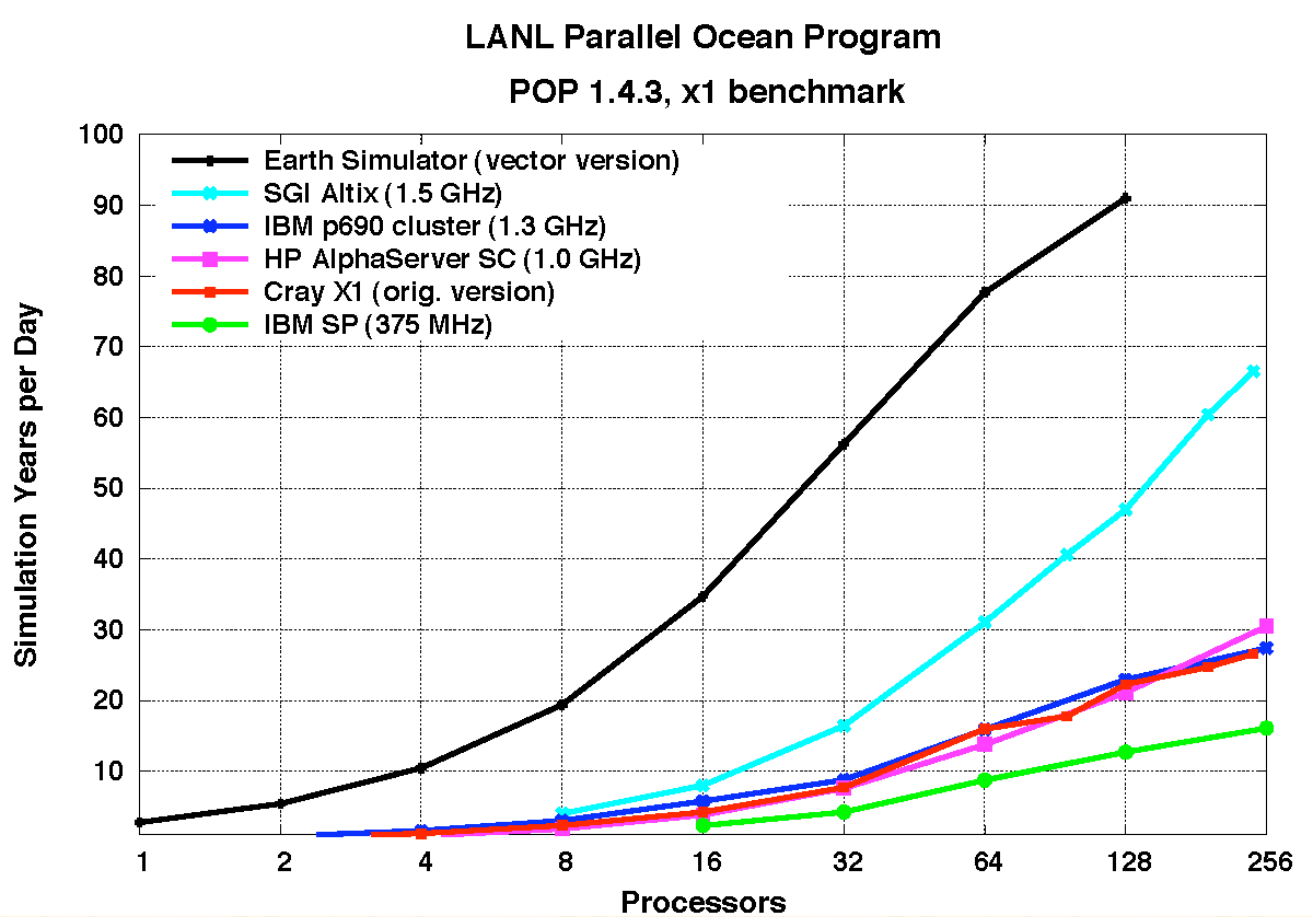
POP Experiment Particulars

- 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.
- One fixed size benchmark problem
  - One degree horizontal grid (“by one” or “x1”) of size 320x384x40.
- Domain decomposition determined by grid size and 2D virtual processor grid. Results for a given processor count are the best observed over all applicable processor grids.

POP Platform Comparison: Initial

Comparing performance and scaling across platforms.

- Earth Simulator results courtesy of Dr. Y. Yoshida of the Central Research Institute of Electric Power Industry
- IBM SP results courtesy of Dr. T. Mohan of Lawrence Berkeley National Laboratory
- X1 results (using standard distribution) are indistinguishable from those collected on IBM and HP systems.



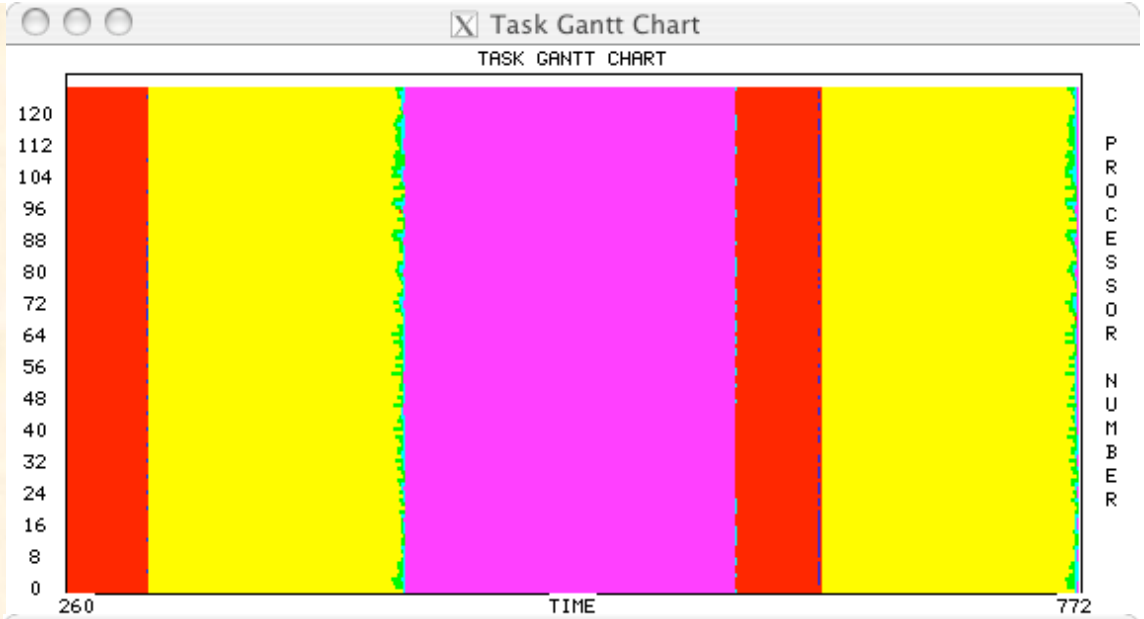
POP Vectorization

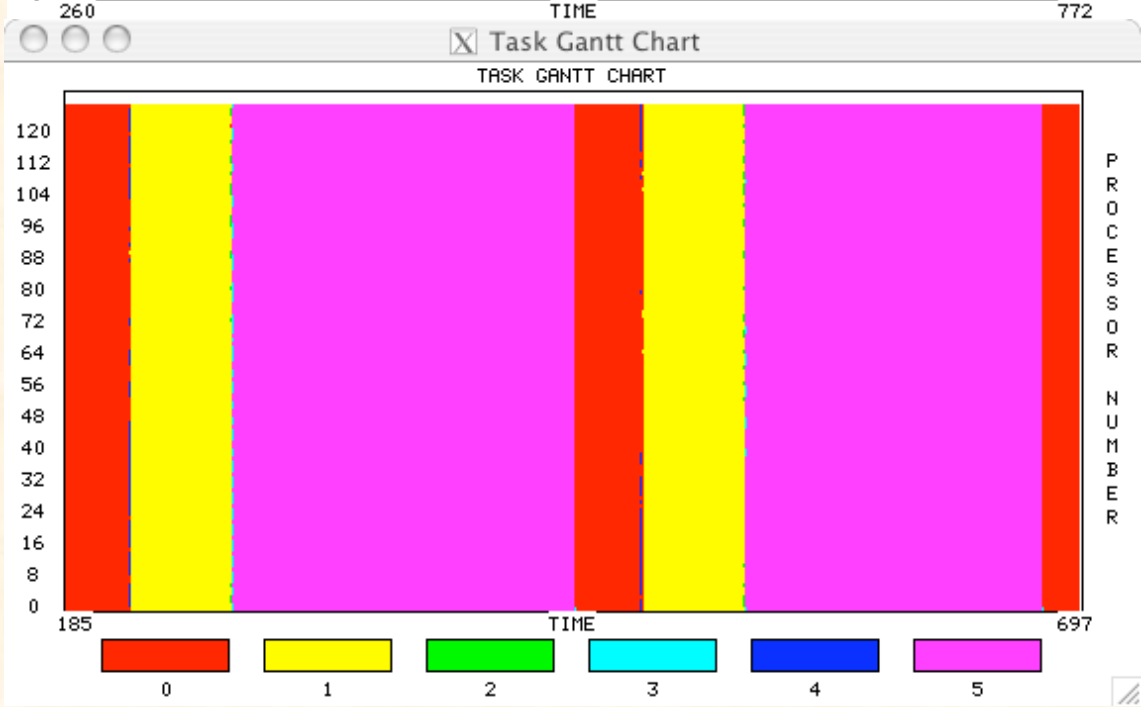
- Began with port to Earth Simulator
  - 701 lines replaced (by 1168 lines), out of 45000 lines
  - Over half of the ES modifications (~400 lines) do not change performance on the X1 significantly: e.g., replacement of F90 where, merge, eoshift, … by F77 equivalents.
- Modified two (previously modified) routines to improve performance of ES port on the X1
  - Number of lines replaced in original version approximately the same for the ES and X1 versions currently

Performance Impact of Vectorization

Task Gannt chart before and after code vectorization for 128 MSP run:

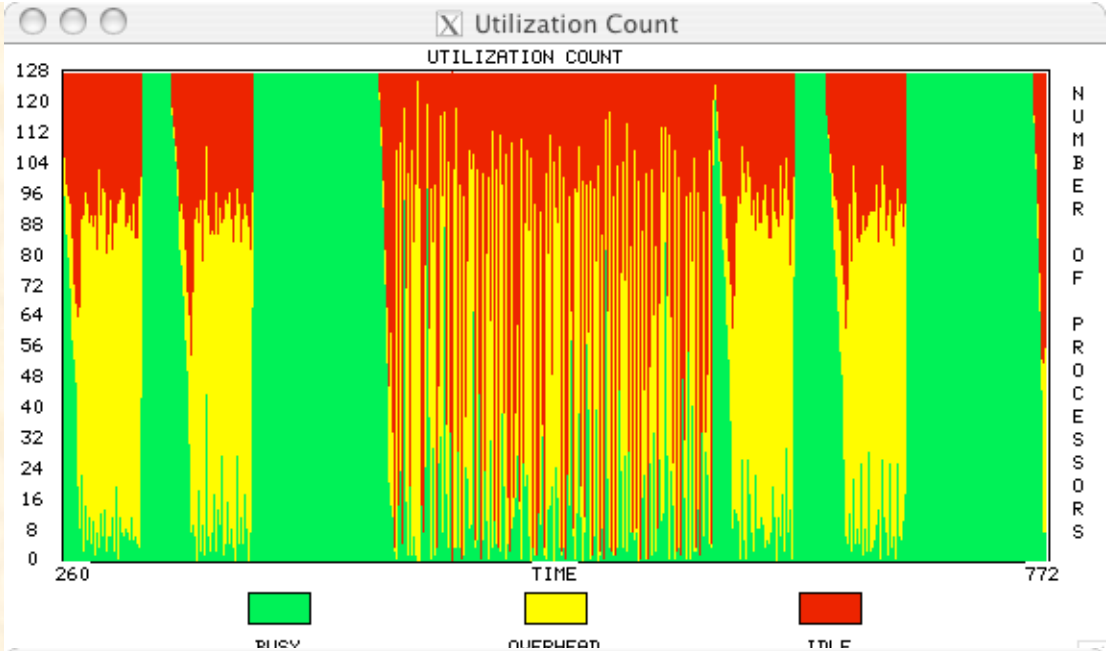
0: (primarily tracer updates)
1: baroclinic
2: baroclinic boundary update
3: barotropic (excl. solver)
4: barotropic boundary update
5: barotropic solver

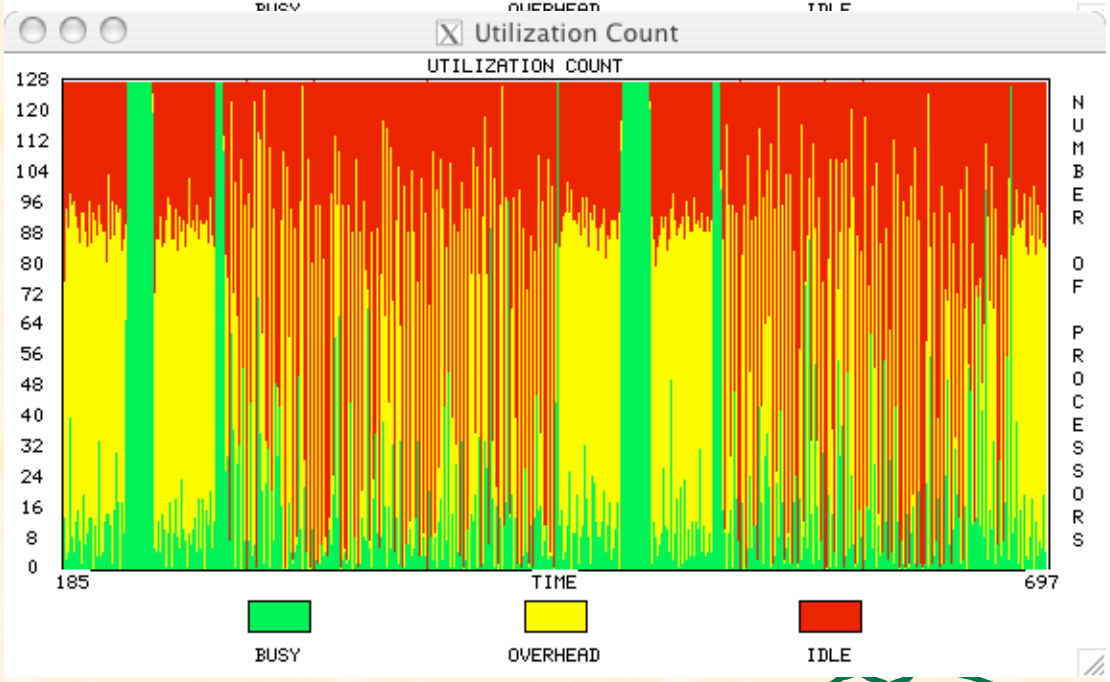




Performance Impact of Vectorization

Compute (Busy) / Communicate (Overhead and Idle) utilization graph before and after code vectorization for 128 MSP run.





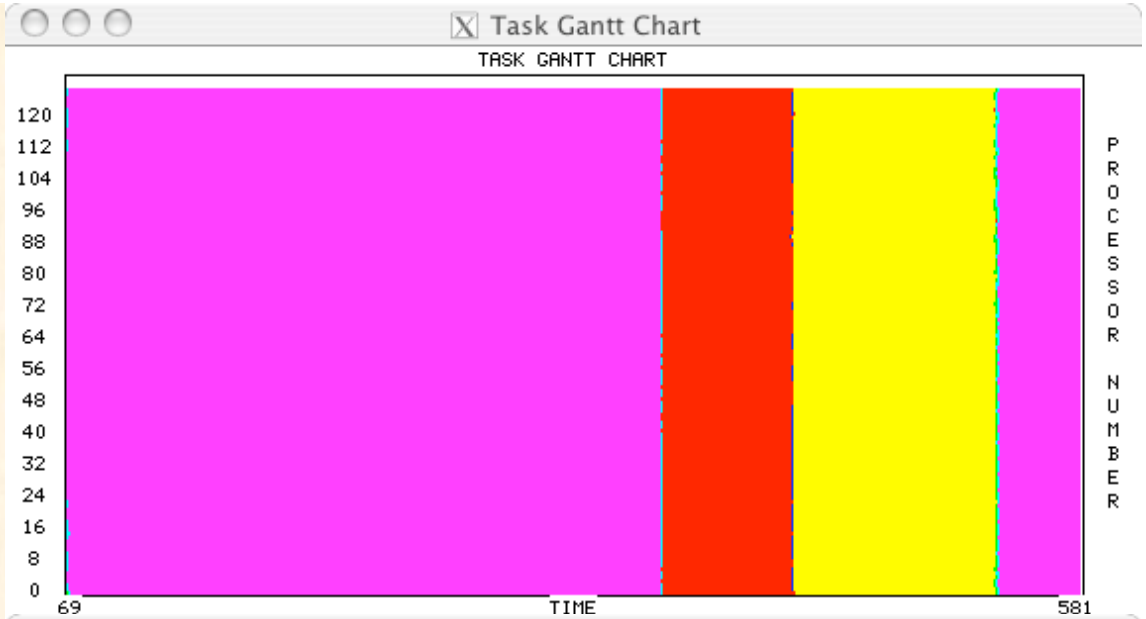
POP MPI Optimization

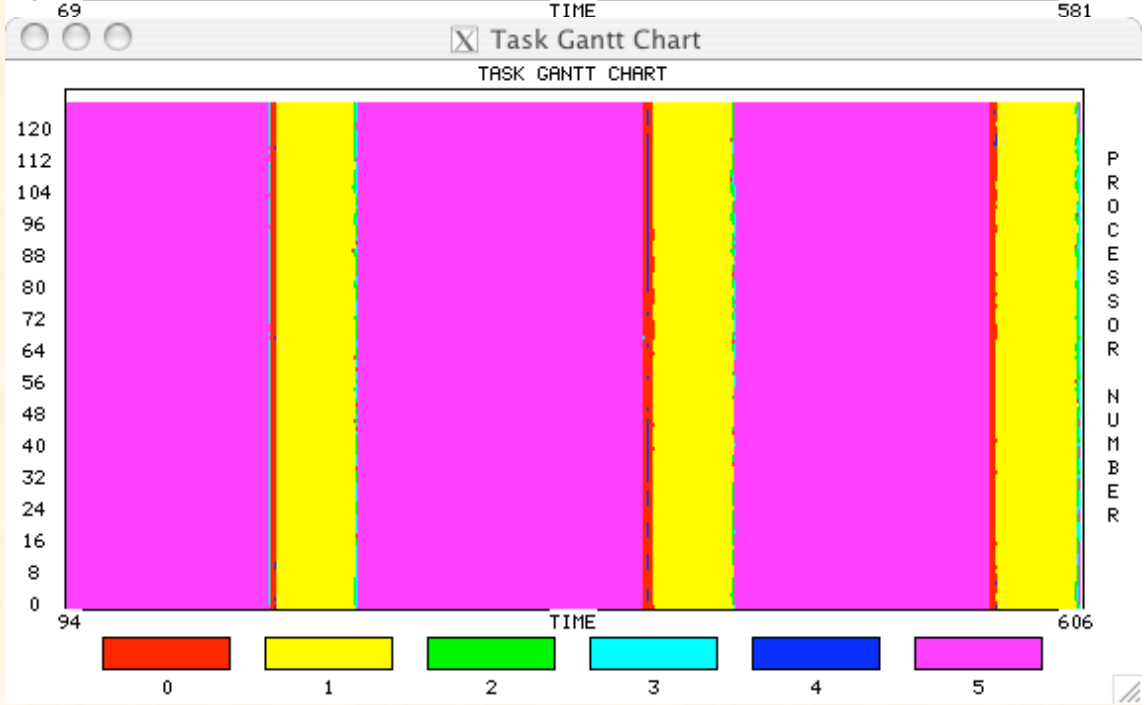
- Used Earth Simulator port unchanged
  - 125 lines replaced (by 233 lines); one new (140 line) routine added.
  - Five routines modified to replace irecv/isend logic with isend/recv logic. (The performance of POP on the X1 is not sensitive to this. Neither approach degrades overall performance.)
  - Three routines modified to replace communication of halo regions using derived datatypes with packing/unpacking message buffers and using standard datatypes. Routines called in barotropic phase.
  - New routine added to block communication, replacing communication of many small messages by a few large messages. Routine called in baroclinic phase and in baroclinic_correct_adjust (updating tracers).

Performance Impact of MPI Tuning

Task Gannt chart before and after MPI optimization for 128 MSP run:

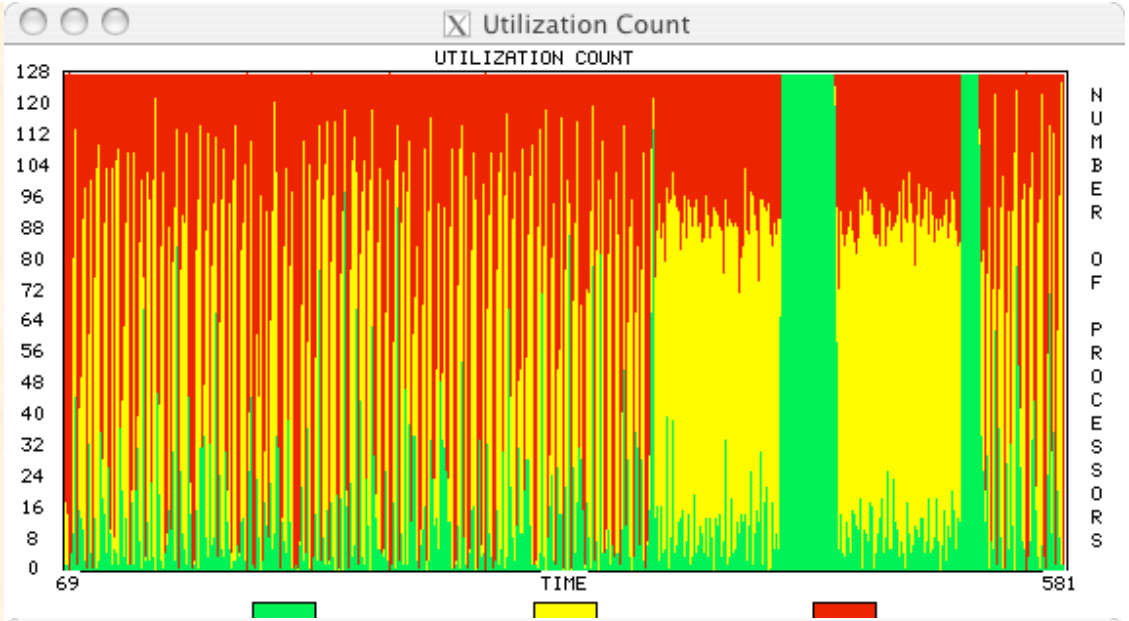
0: (primarily tracer updates)
1: baroclinic
2: baroclinic boundary update
3: barotropic (excl. solver)
4: barotropic boundary update
5: barotropic solver

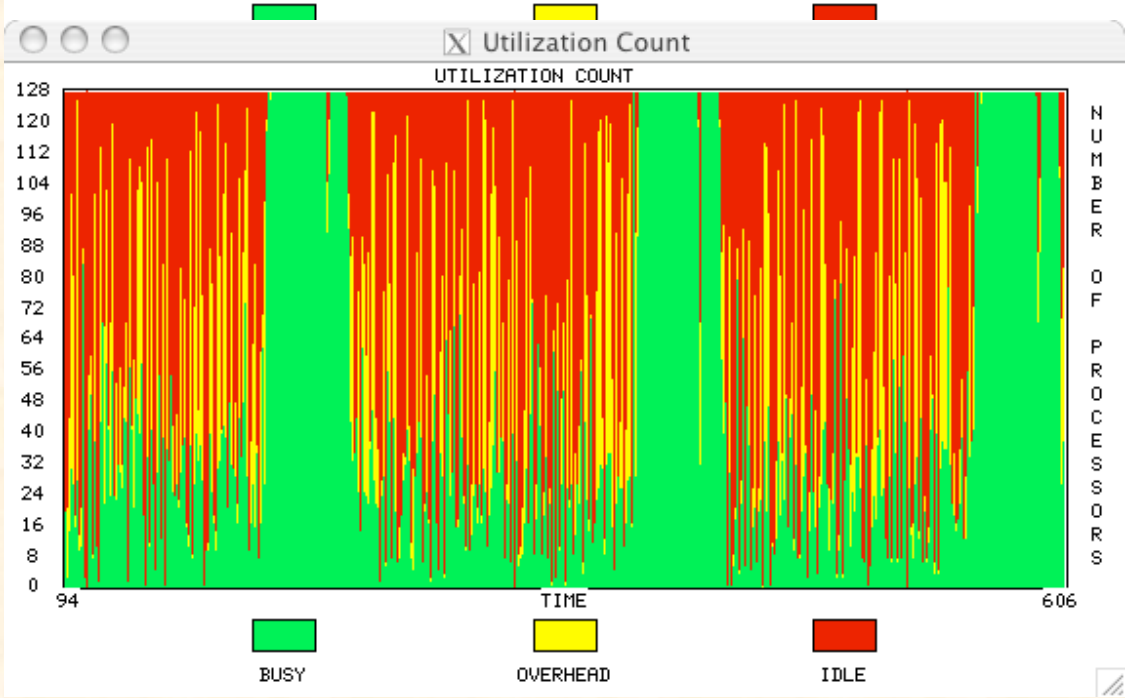




Performance Impact of MPI Tuning

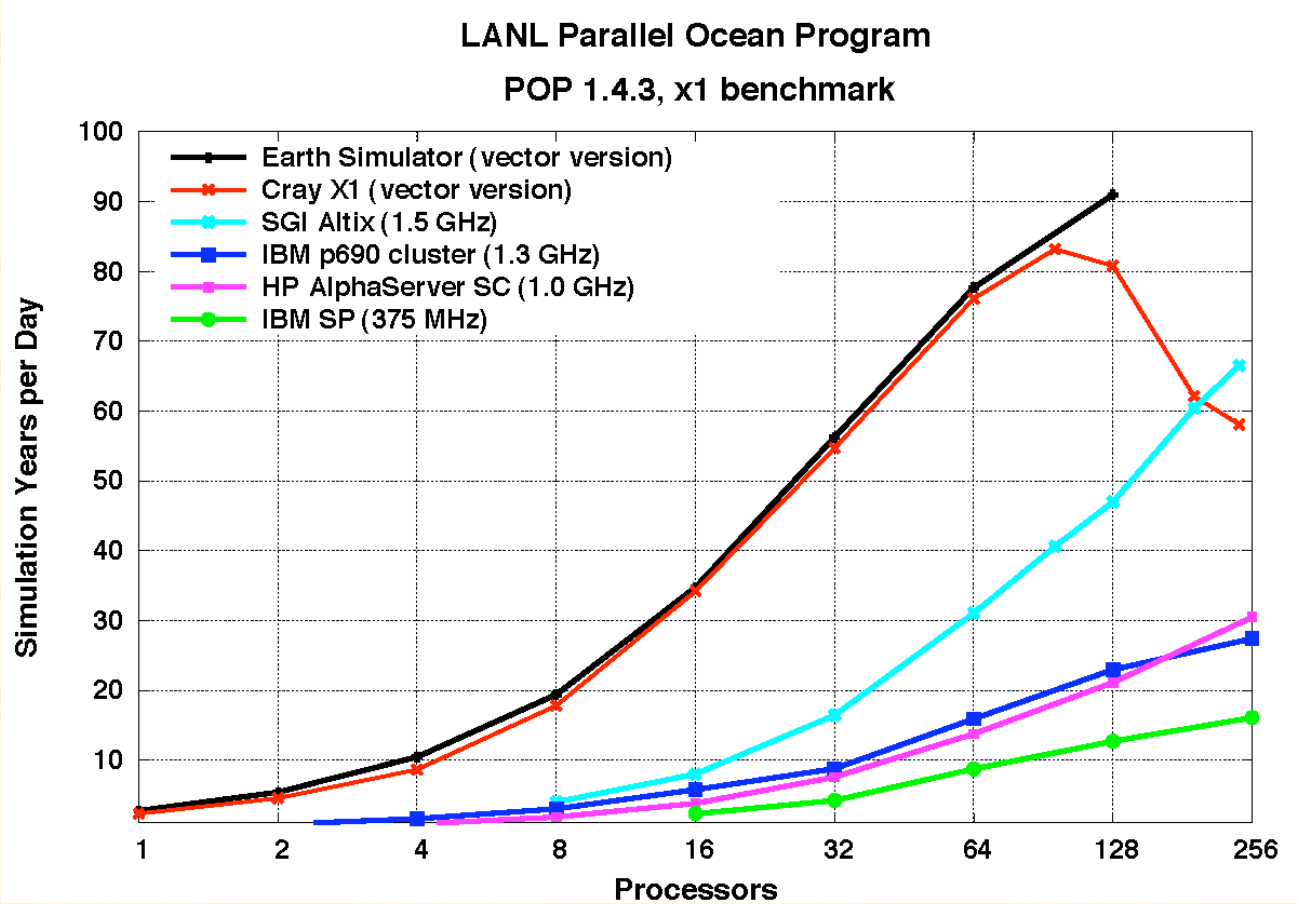
Compute (Busy) / Communicate (Overhead and Idle) Utilization Graph before and after MPI optimization for 128 MSP run.





POP Platform Comparison: MPI-Only

Comparing performance and scaling across platforms. Performance on ES40 and X1 are similar when using ES40 optimizations. Performance scales poorly on X1 when using more than 96 processors.



POP Co-Array Fortran Optimization

- Replaced MPI implementation with Co-array Fortran for two routines:

  NINEPT_4: Weighted nearest neighbor sum for 9 point stencil, requiring a halo update. Used to compute residuals in conjugate gradient solver in barotropic phase.

  GLOBAL_SUM: Global sum of the "physical domain" of a 2D array. Used to compute inner product in conjugate gradient solver in barotropic phase. MPI version used MPI_Allreduce.

- Spent 9 months trying different implementations of these two routines, attempting to improve POP scalability. OS changes over this period had significant impact on the performance.

POP Co-Array Fortran Optimization

May 7

- GLOBAL_SUM: Master reads partial sums from remote memory, completes sum, writes results back to other processes' memory. Global barrier used for synchronization.
- NINEPT_4: Each process reads remote memory to update local halo. East-West updates occur first, followed by North-South updates. Global barrier used for synchronization.

August 12

- GLOBAL_SUM: Processes write partial sums to master's memory. Flags used for pairwise synchronization, not global barrier.
- NINEPT_4: Each process writes remote memory to fill remote halos. Flags used for pairwise synchronization, not global barrier.

POP Co-Array Fortran Optimization

September 1

- GLOBAL_SUM: Add padding to co-arrays to eliminate contention.
- NINEPT_4: Add padding to co-arrays to eliminate contention.

October 25

- GLOBAL_SUM: Use special values in data co-arrays to implement pairwise synchronization (eliminating flags).
- NINEPT_4: Same as Sep. 1.

POP Co-Array Fortran Optimization

December 13a

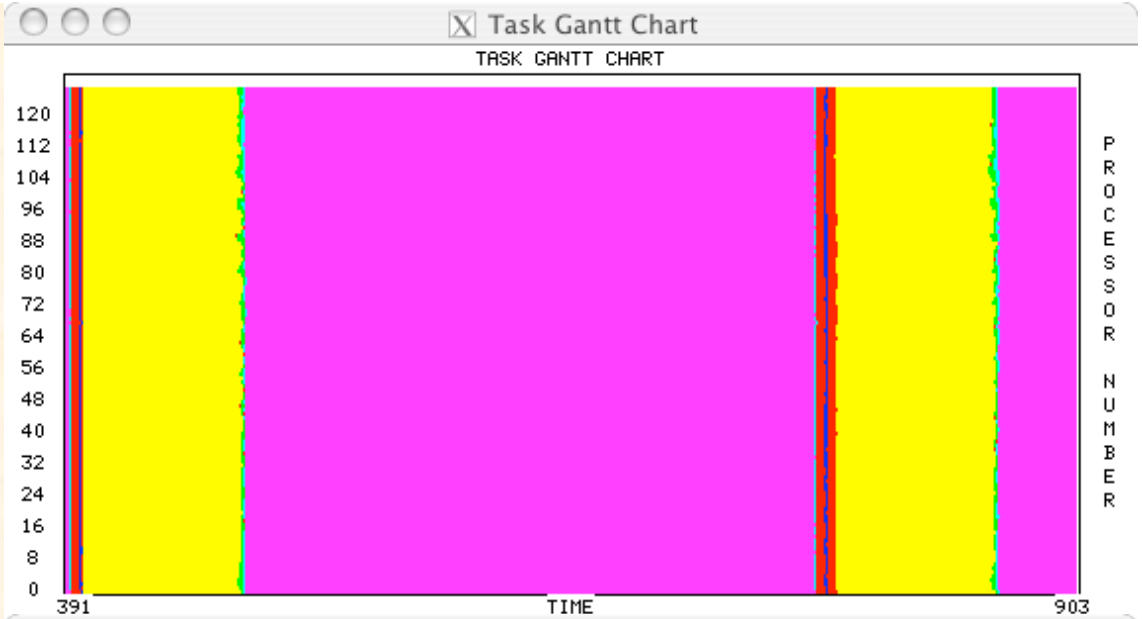
- GLOBAL_SUM: Variant of May 7 algorithm in which partial sums are written to the master's memory. Global barrier is used for synchronization.
- NINEPT_4: Same as Aug. 12 algorithm.

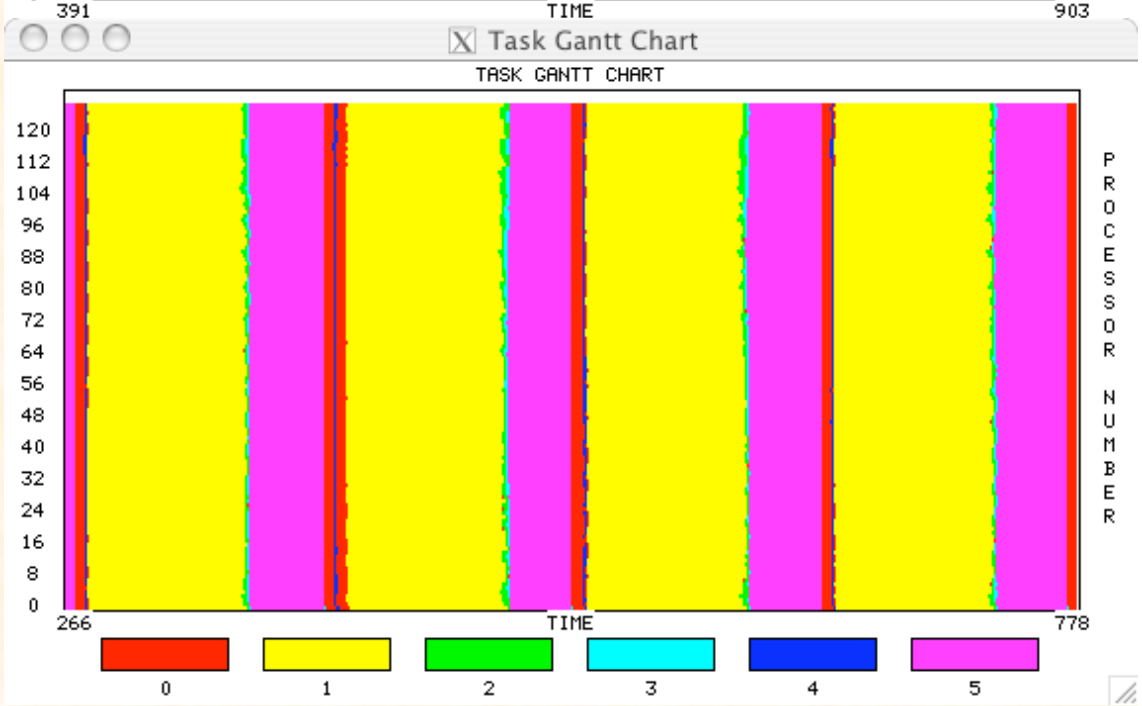
December 13b

- GLOBAL_SUM: Tree sum and tree broadcast variant of Sep. 1 algorithm.
- NINEPT_4: Same as Aug. 12 algorithm.

Performance Impact of Co-Array Fortran

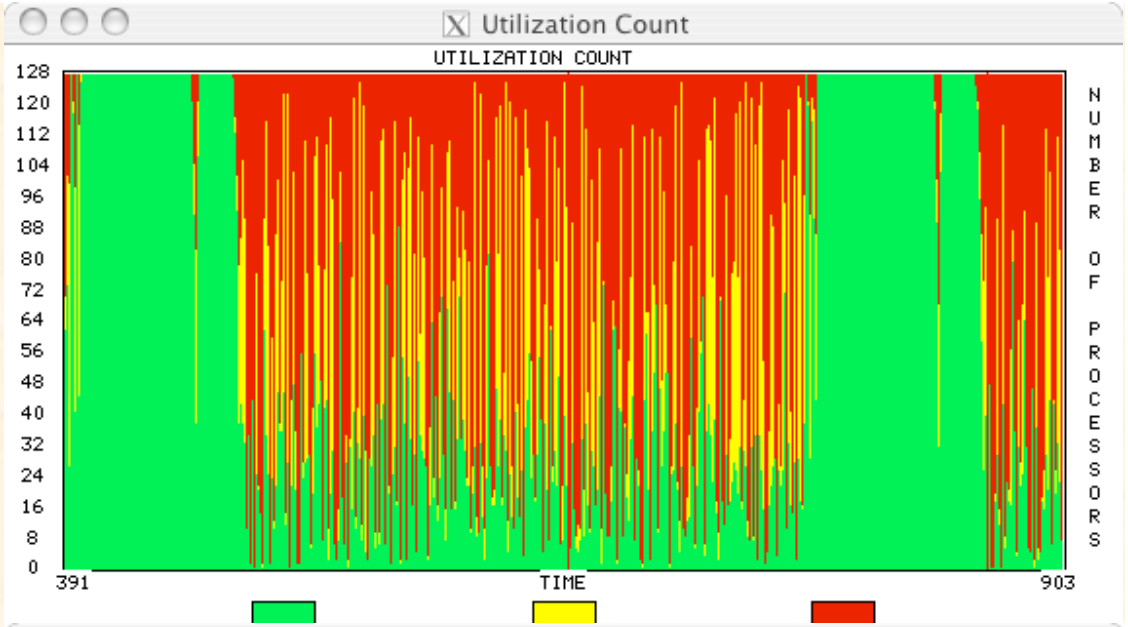
Task Gannt chart before and after replacement of MPI allreduce and halo update in barotropic solver (task #5) with Co-Array Fortran (Dec13b version) for 128 MSP run.

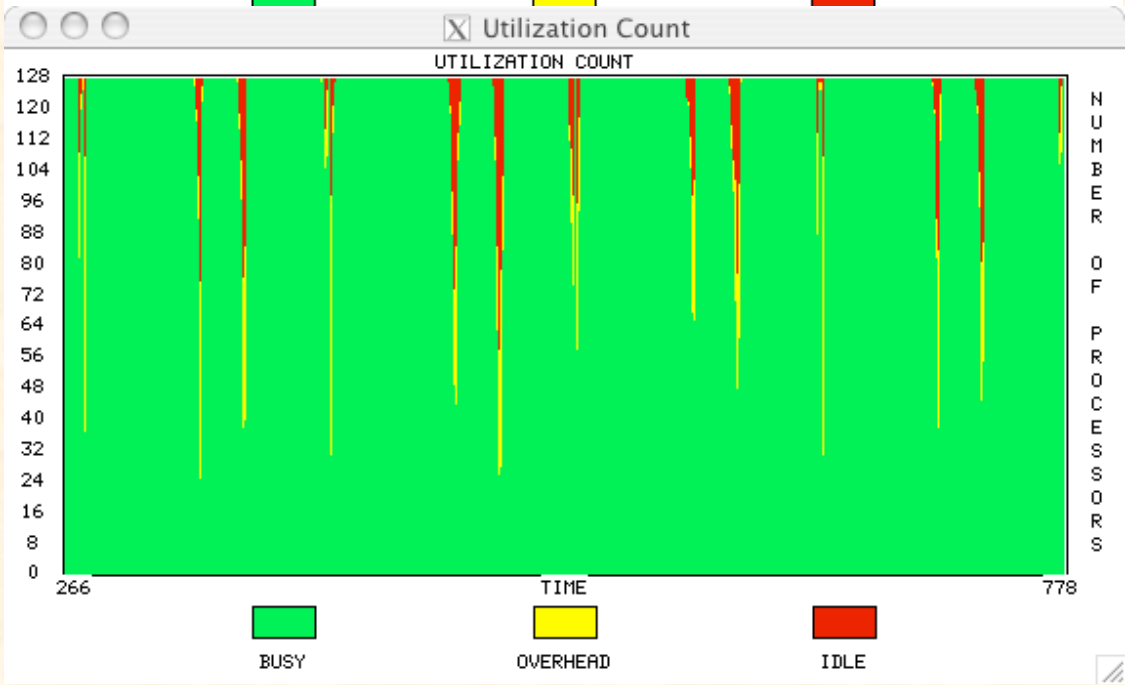




Performance Impact of Co-Array Fortran

Compute (Busy) / Communicate (Overhead and Idle) utilization Graph before and after replacement of MPI allreduce and halo update in barotropic solver for 128 MSP run. Time spent in Co-Array Fortran is not marked as communication overhead. Data indicate that (remaining) MPI overhead is not limiting performance.



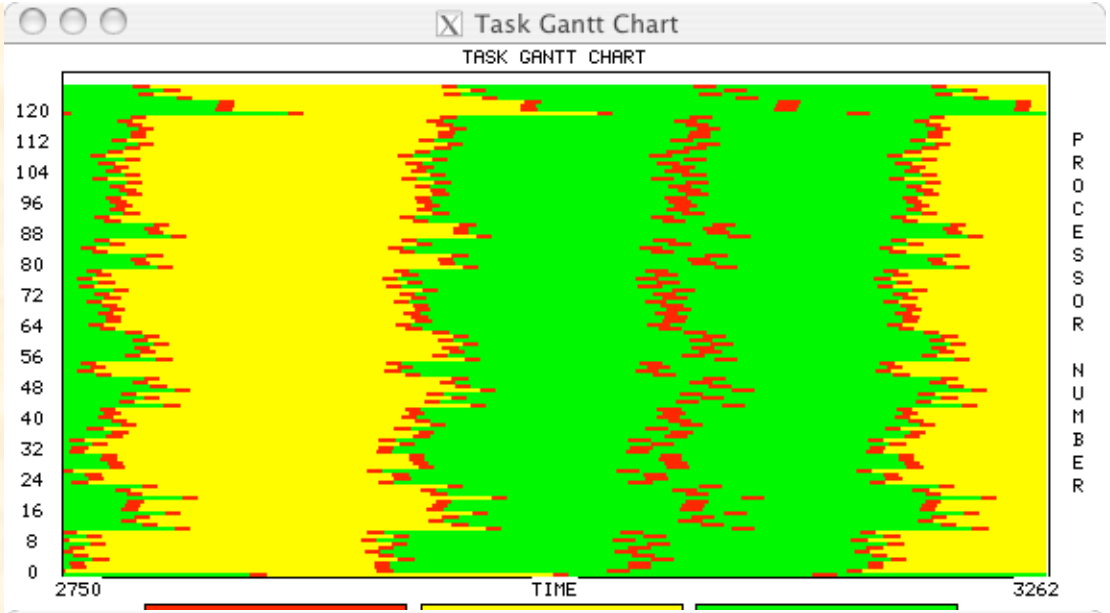


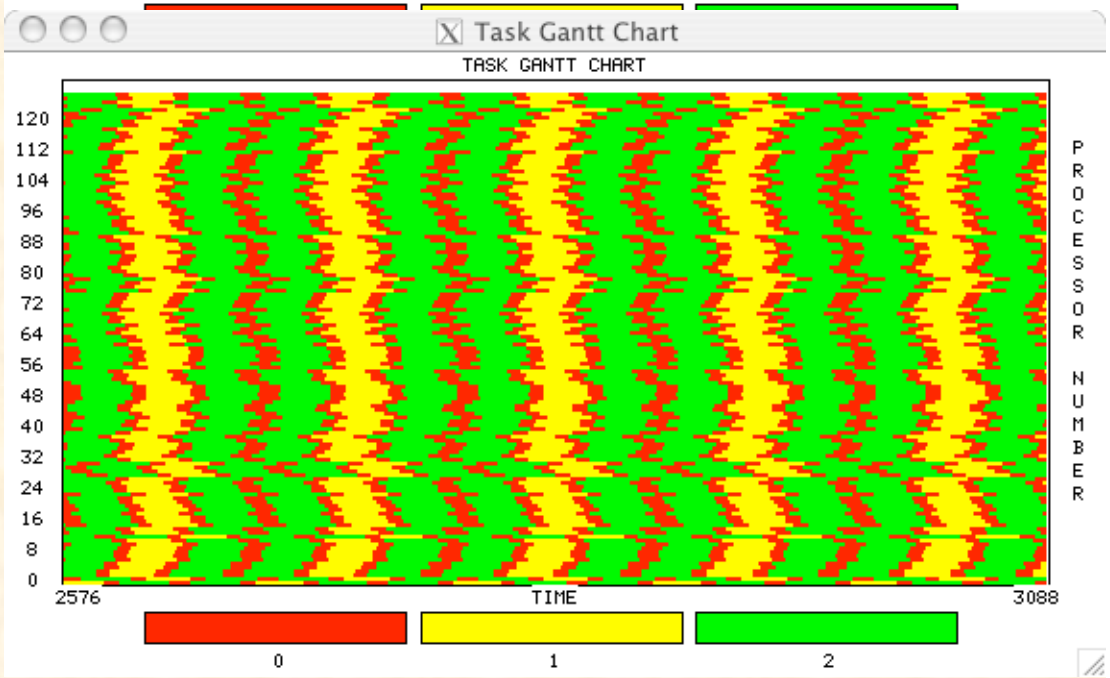
Performance Impact of Co-Array Fortran

Task Gannt chart before and after replacement of MPI allreduce and halo update in barotropic solver for 128 MSP run:

0: “other”
1: Allreduce
2: Halo Update

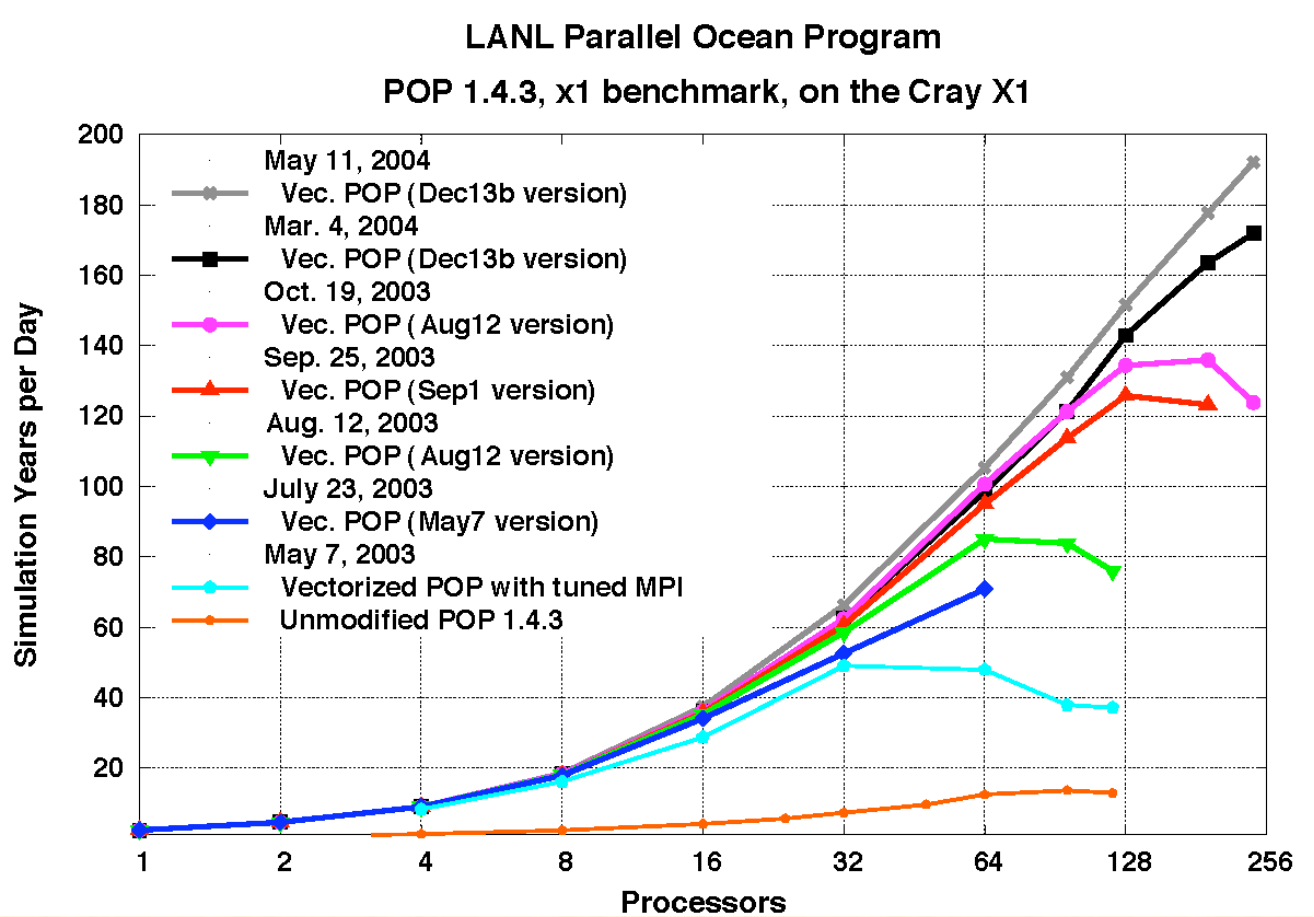
Data comes from within solver.





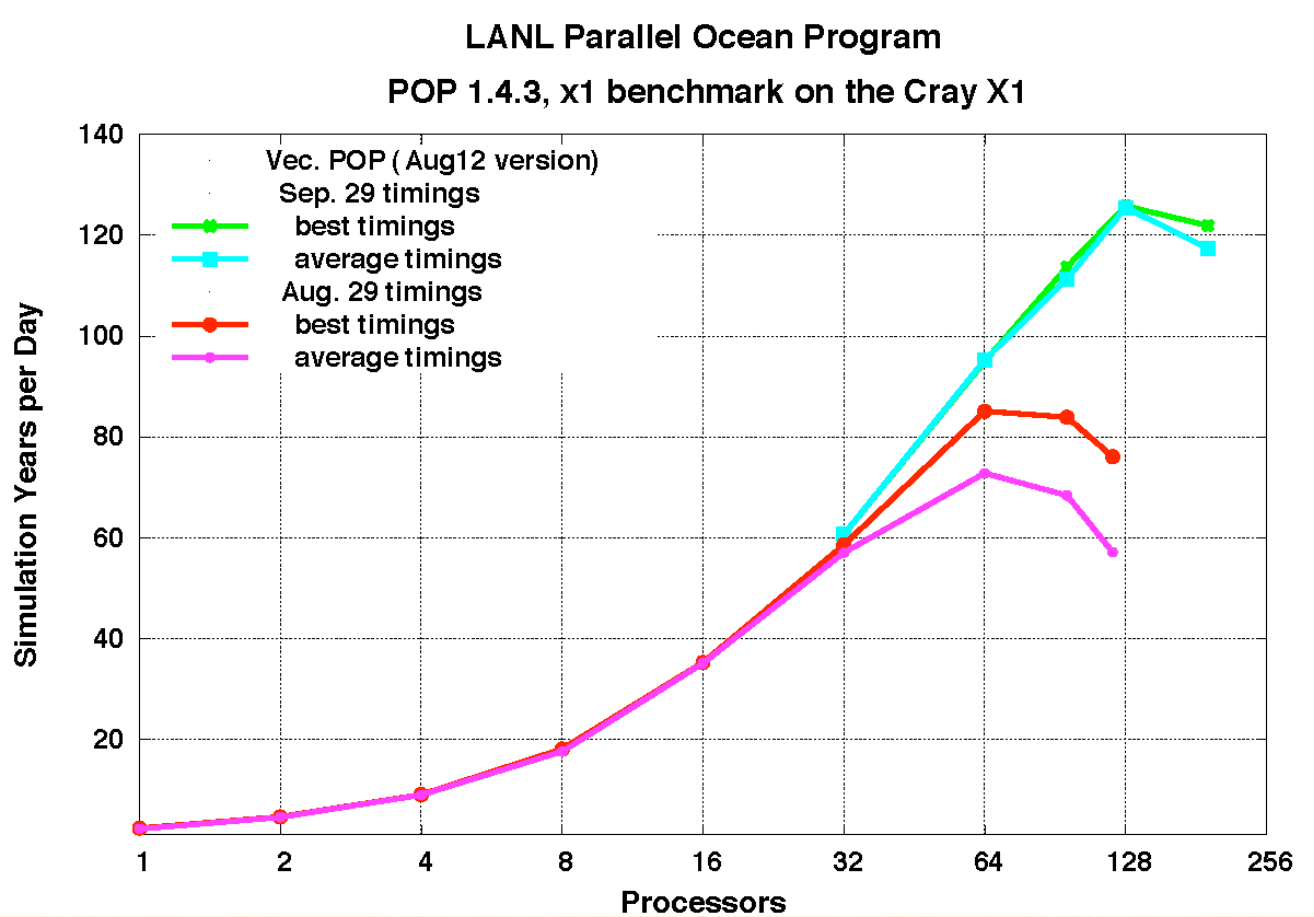
POP Performance Evolution on the X1

Comparing performance and scaling on the X1 over time. Many of the performance improvements are due to updates to the OS and other system software. Some of these updates were motivated by the parallel algorithm analysis and optimizations.



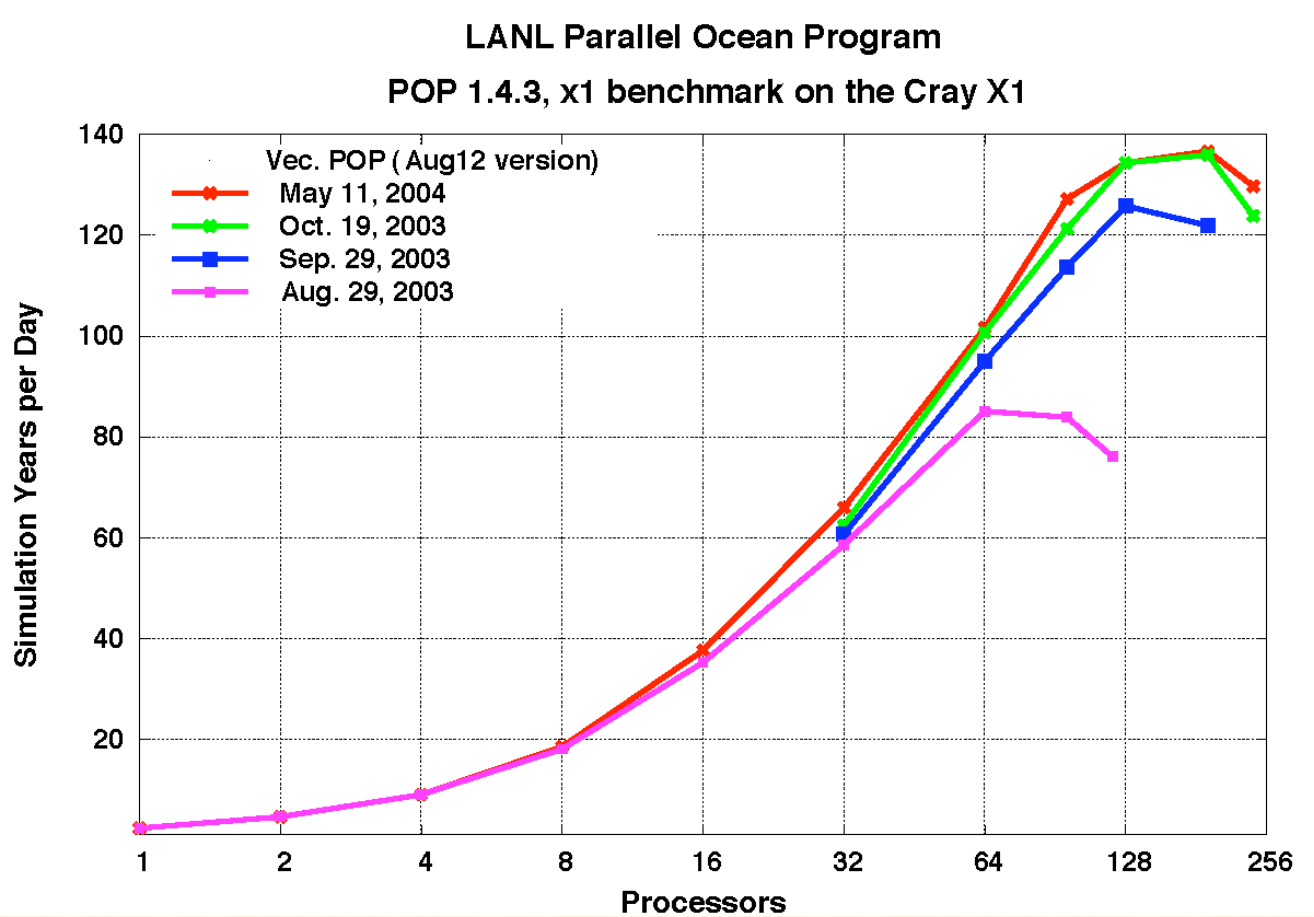
Impact of OS Interrupts on Scalability

Comparing average and best performance of the Aug. 12 version before and after the Sep. 1, 2003 OS update. The update used a global clock to schedule system interrupts, improving best performance and decreasing performance variability.



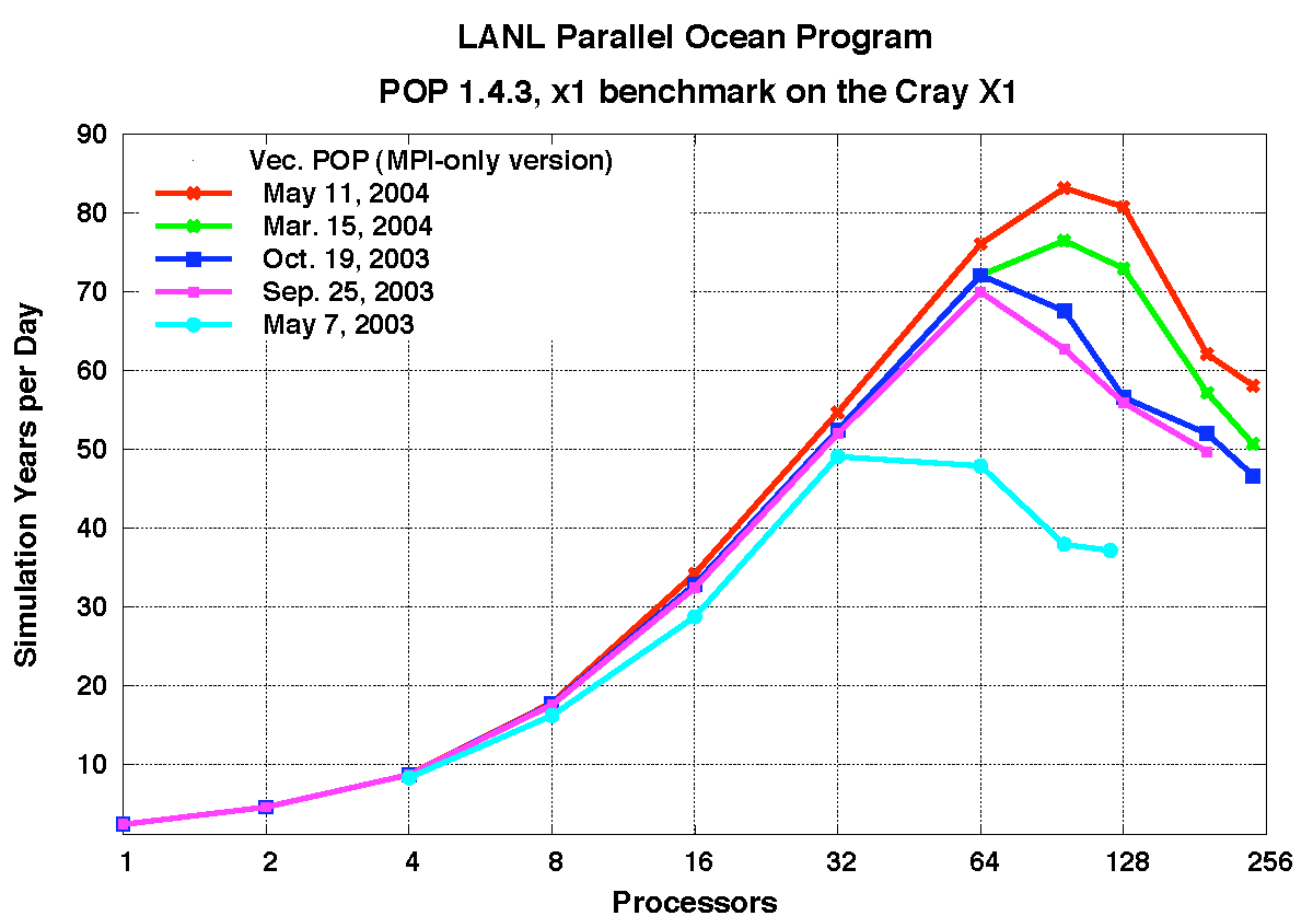
Impact of OS Updates on Performance

Comparing performance of the Aug. 12 version on the X1 over time. While the largest improvement was from the Sep. 1, 2003 update, performance continued to improve over time.



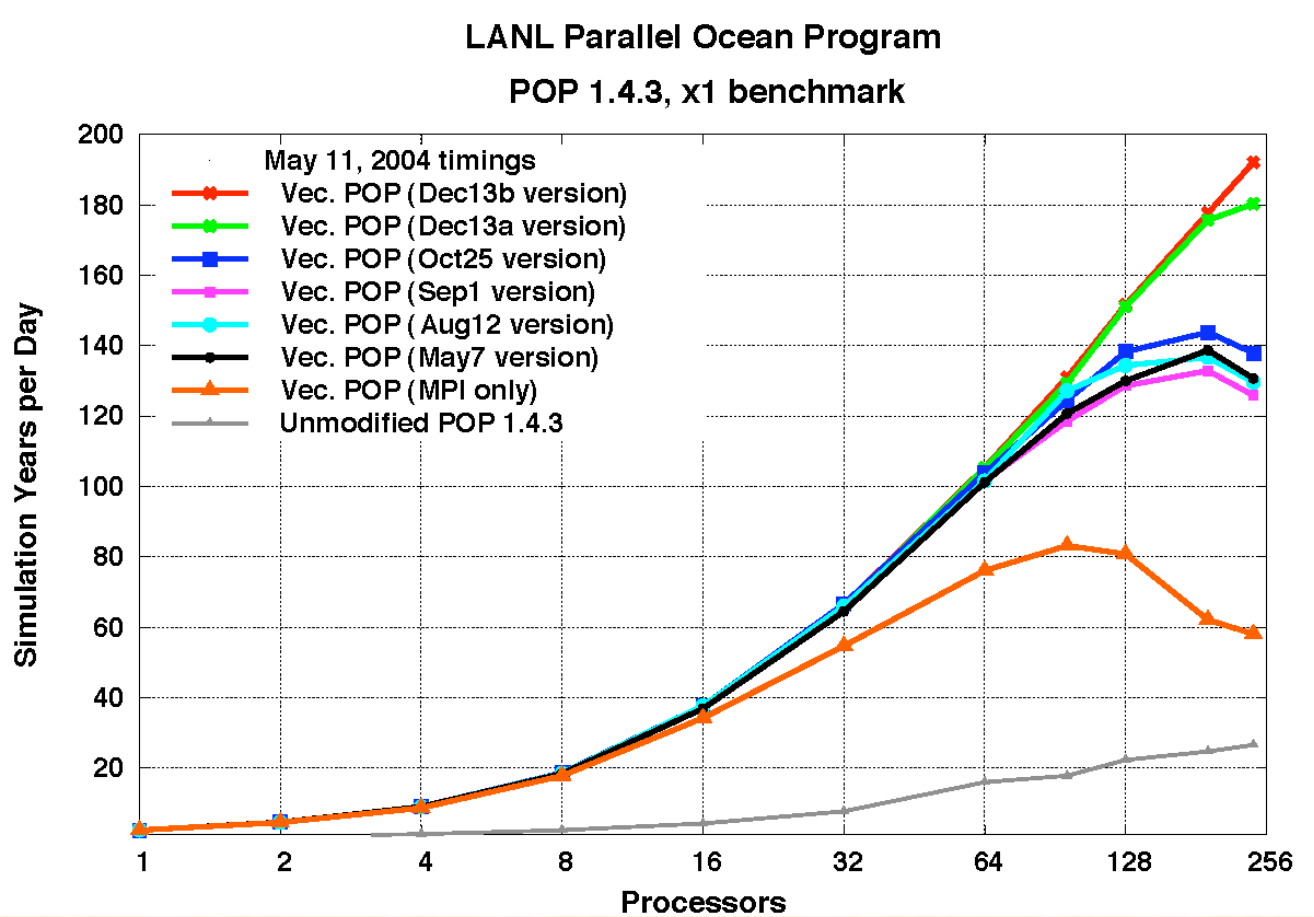
Impact of OS Updates on Performance

Comparing performance of the MPI-only vector version on the X1 over time. While still not competitive with the Co-Array Fortran implementations, MPI performance has improved.



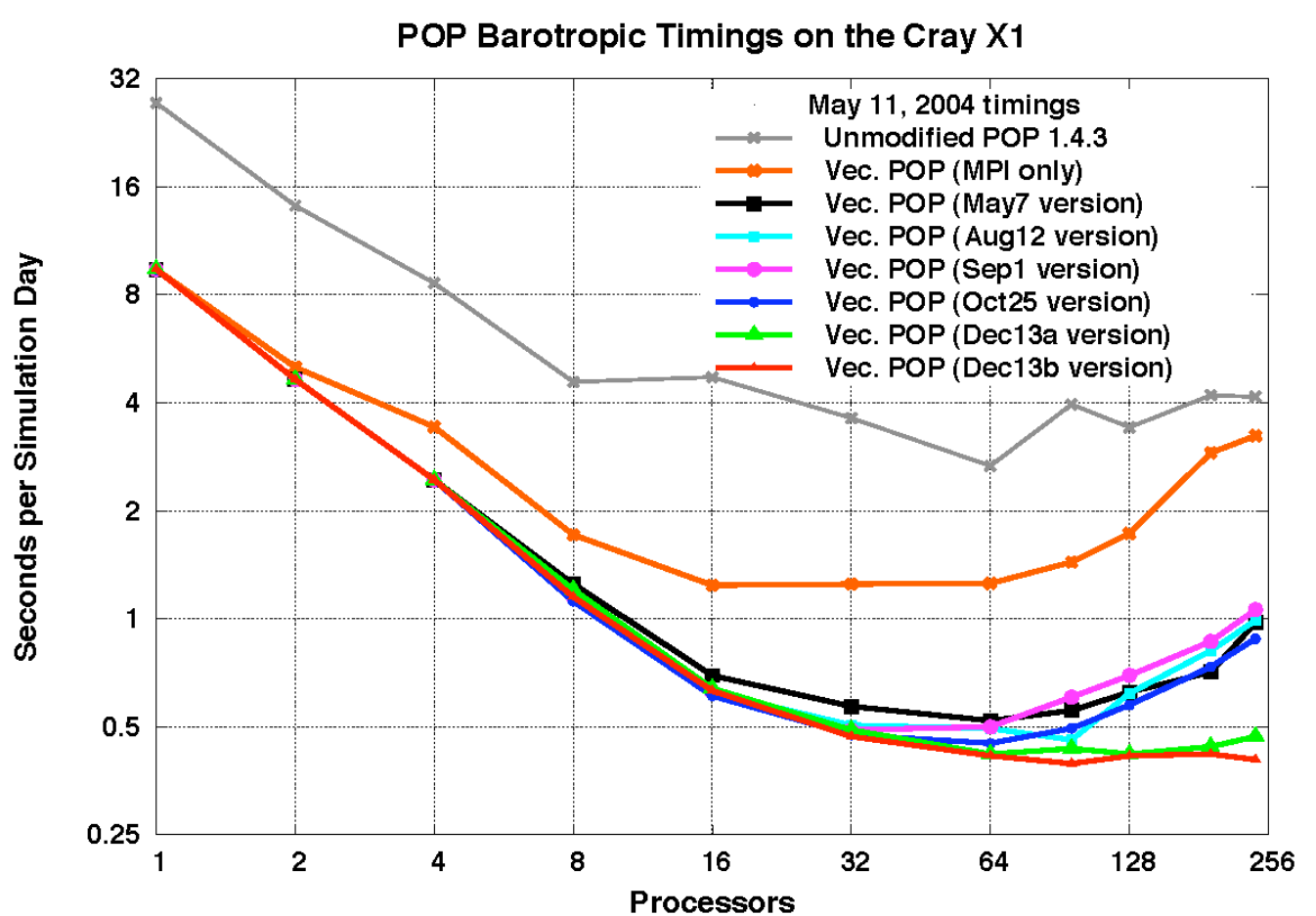
POP Implementation Comparison

Comparing performance and scaling on the X1 for the different versions of POP. Much of the algorithm development was motivated by OS performance problems. Once OS problems were solved, algorithmic development became more "effective".



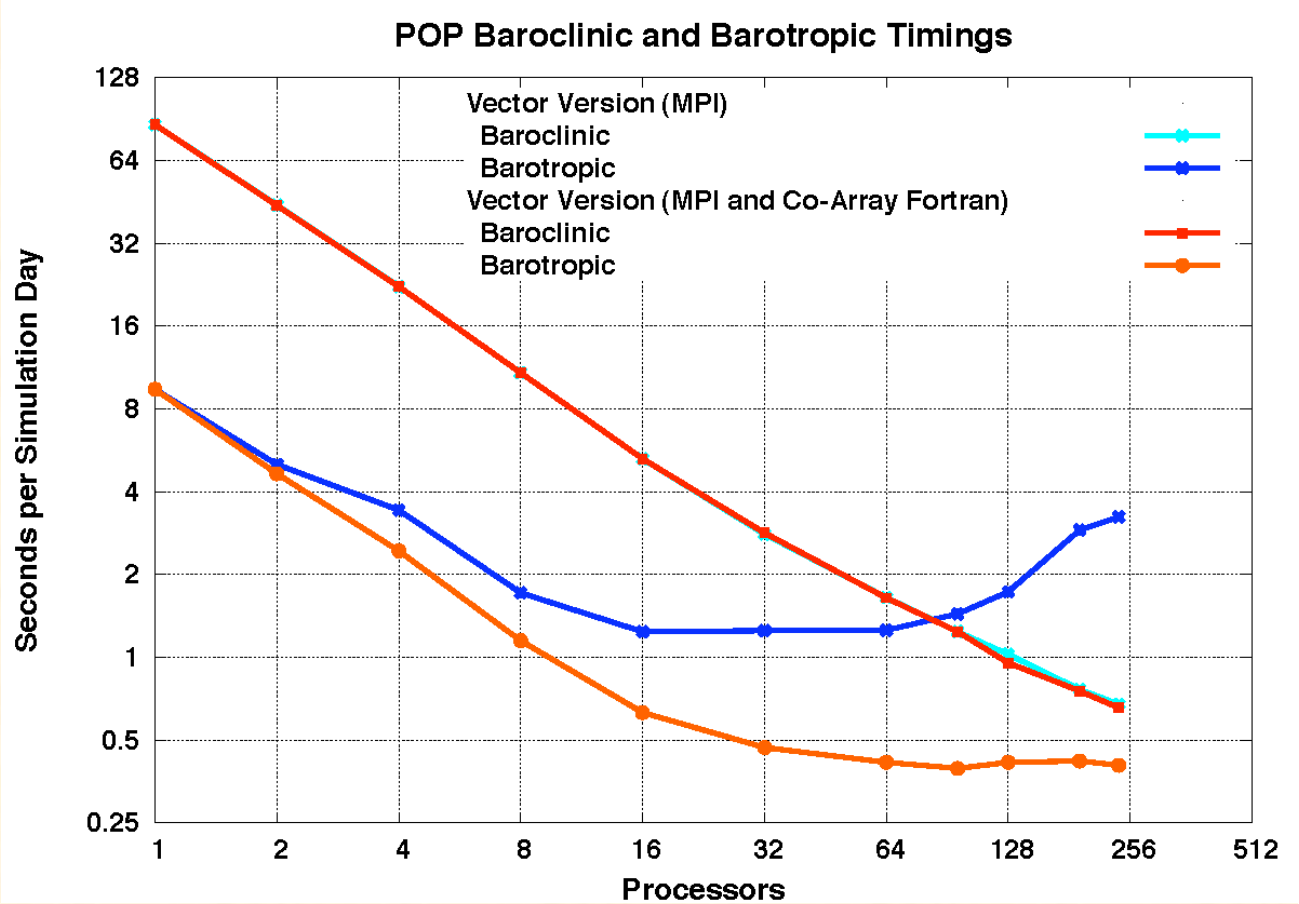
POP Implementation Comparison

The performance advantage of the best algorithms is in the scalability of the allreduce in the barotropic solver.



POP Implementation Comparison

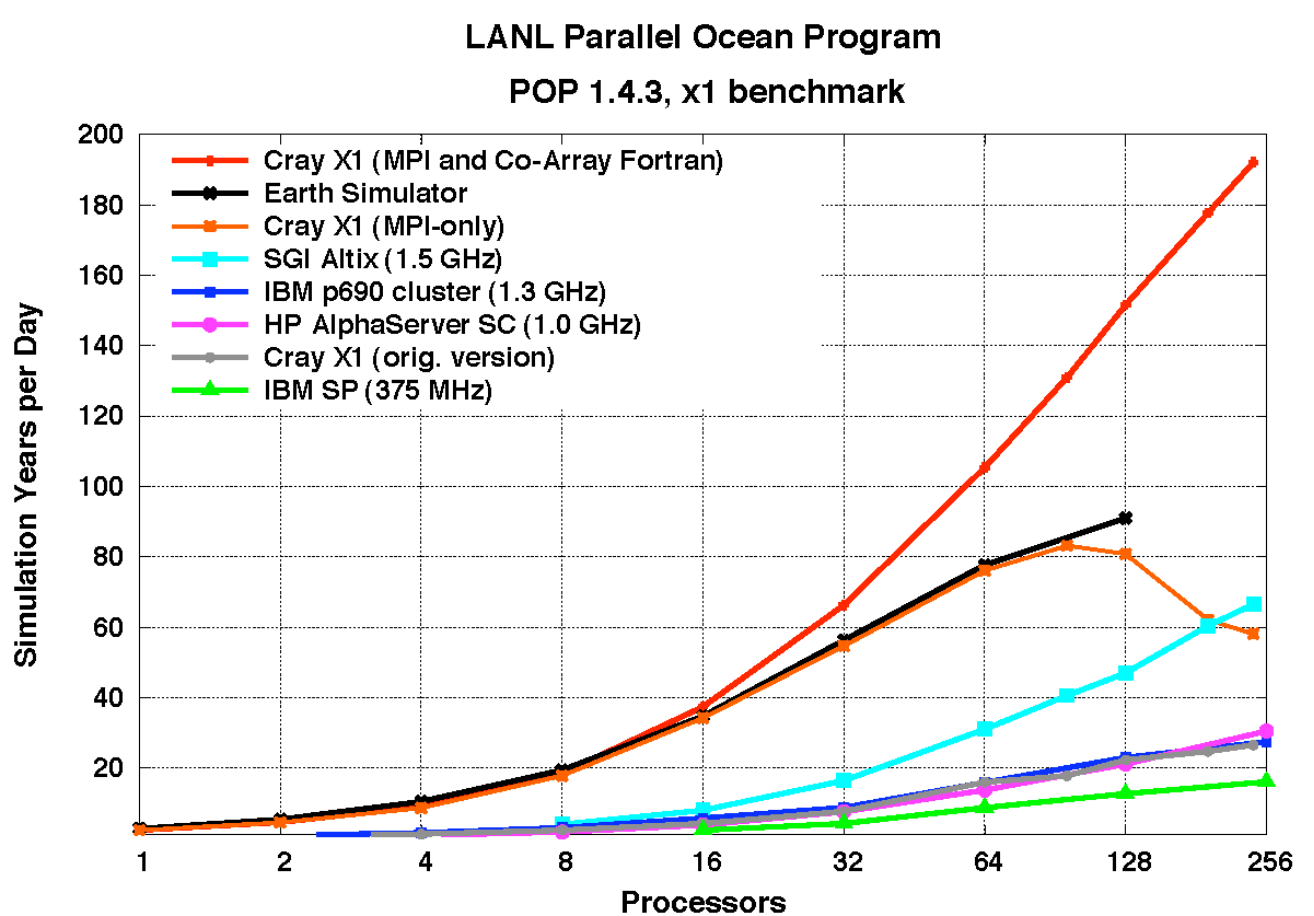
Comparing performance of MPI-only and Dec13b hybrid MPI/Co-Array Fortran vector implementations for both baroclinic and barotropic phases. The MPI-only implementation is “barotropic-bound” for more than 64 processors. In contrast, the hybrid implementation is still “baroclinic-bound”, and performance is continuing to scale to higher processor counts.



POP Platform Comparisons: Current

Comparing performance and scaling across platforms.

- Earth Simulator results courtesy of Dr. Y. Yoshida of the Central Research Institute of Electric Power Industry
- IBM SP results courtesy of Dr. T. Mohan of Lawrence Berkeley National Laboratory



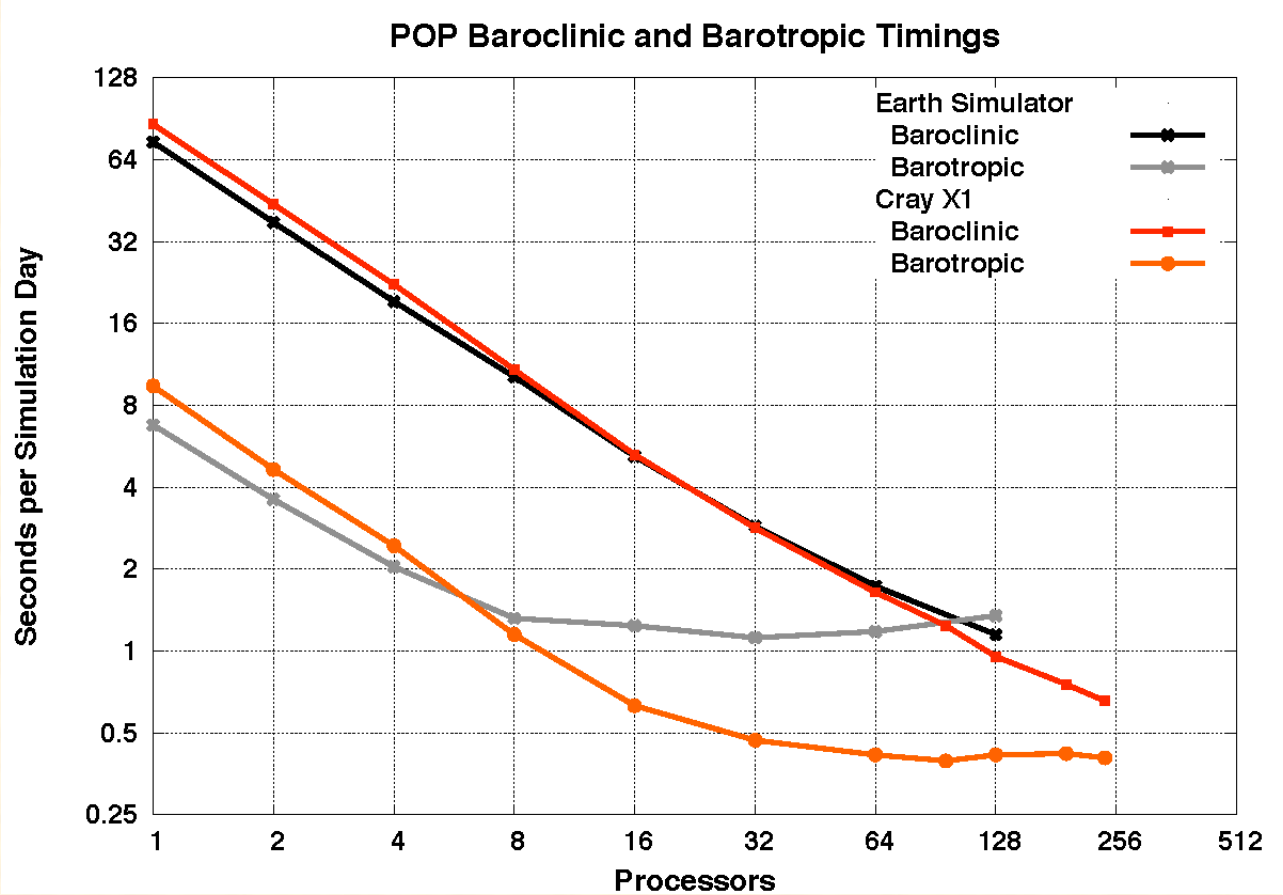
POP Performance Diagnosis vs. ES40

Cray X1

Not yet communication-bound. Communication costs not yet starting to increase.

Earth Simulator

Communication-bound for 128 processors, and communication costs beginning to increase. Better performance for large granularity, but worse performance compared to X1 for small granularity (just communication or also shorter vectors?).



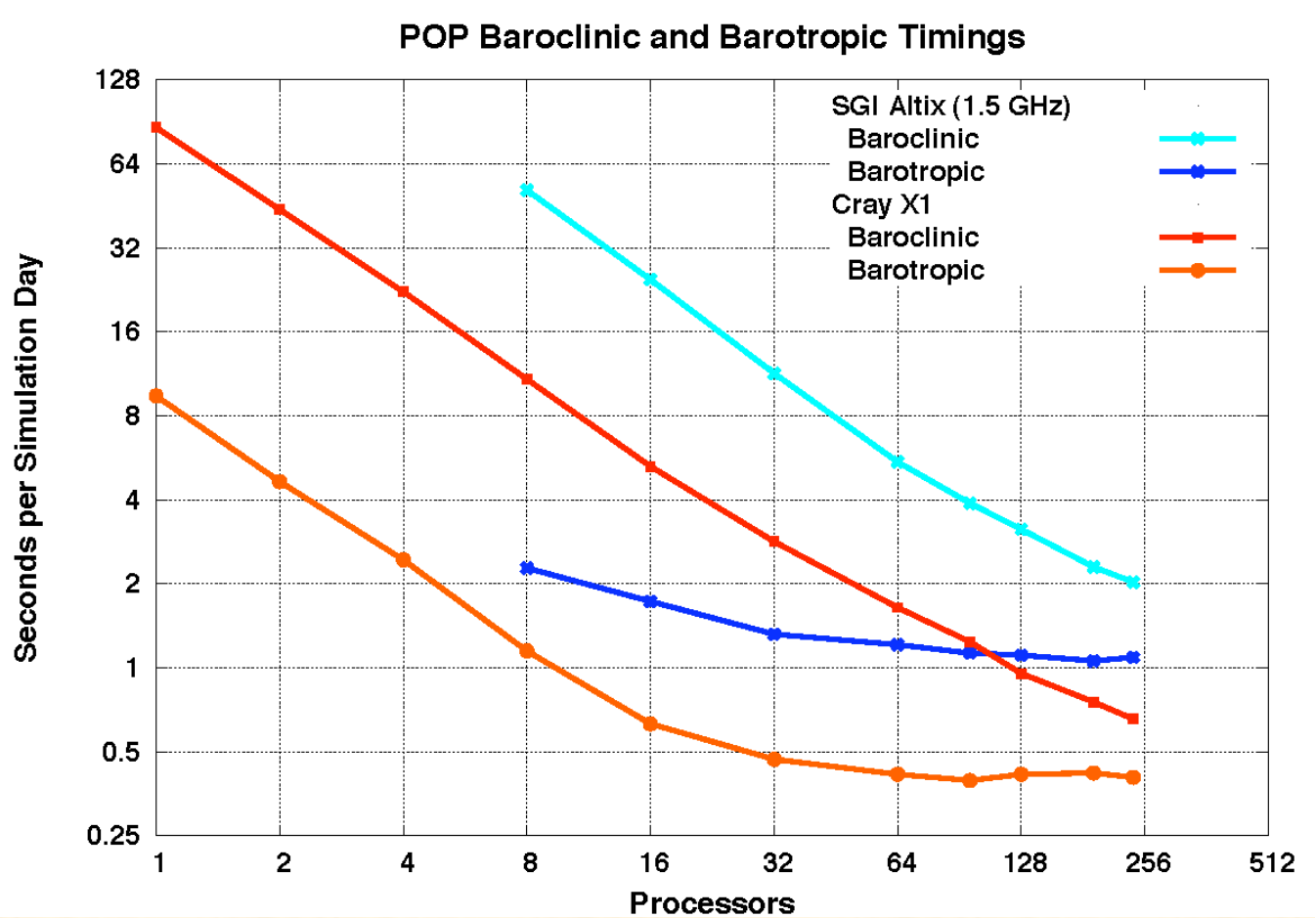
POP Performance Diagnosis vs. Altix

Cray X1

Not yet communication-bound. Communication costs not yet starting to increase.

SGI Altix

Not yet communication bound. Using MPI point-to-point and collectives (ES40 version). Initial experiments with SHMEM do not show significant improvement.



POP Performance Evolution Lessons

- NEC optimizations were reasonably efficient on the Cray X1.
  - Many of the modifications were unnecessary for the X1, but they did no harm.
  - Need to revisit NEC optimizations, to see whether additional Cray-specific optimizations might be worthwhile.
- Using MPI derived datatypes can hurt performance (currently).
- Performance of latency-sensitive MPI collective and point-to-point commands limit scalability (currently).
- Performance aspects of the OS (and other system software) have improved significantly over the past year.
- It is difficult to solve OS performance problems with parallel algorithm optimizations.

What's Next for POP?

- Optimization and benchmarking for high resolution problems
- Continued scaling studies:
  - Up to 500 processors this summer
  - Up to 1000 processors early 2005
- Continued Co-Array Fortran global_sum and halo update algorithm optimizations (esp. for high processor count and/or large problem sizes)
- Continued MPI evaluations, tracking improvement of latency-sensitivity communication operators
- Continued OS scalability evaluations
- Re-evaluating NEC-inspired vector optimizations
- POP in the Community Climate System Model (CCSM)
- POP2.0

Special Thanks to

- Dr. Phil Jones of LANL, one of the primary authors of POP, for help in obtaining and porting POP (to numerous platforms) and for defining benchmark problems.
- Dr. Yoshikatsu Yoshida of CRIEPI for his excellent port of POP to the Earth Simulator and for providing the performance data from the ES40. Note that the ES40 results were obtained over a year ago. While we are not aware of any further optimizations or more recent performance results, the ES40 version of the code may have continued to evolve as well.
- James B. White III ("Trey") of ORNL for his contributions to the development of Co-Array Fortran algorithms for use in POP.