

Latest Performance Results from ORNL: Cray X1 and SGI Altix

Patrick H. Worley

Oak Ridge National Laboratory

2003 LACSI Symposium
System and Application Performance Workshop
October 27, 2003
Eldorado Hotel
Santa Fe, New Mexico

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY



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.

Evaluation of Early Systems

A project that attempts to evaluate *quickly* the promise of “early” (possibly immature) systems:

- Verifying advertised functionality and performance
- Quantifying performance impact of unique system characteristics
- Providing guidance to (early) users
 - What performance to expect
 - Performance quirks and bottlenecks
 - Performance optimization tips

Early Systems

ORNL is currently “blessed” with a number of early systems:

- Cray X1
 - 64 processors installed in March 2003; upgraded to final 256 processor configuration on 11/14/03.
- SGI Altix
 - Initial system installed in August 2003; upgraded to 1.5 GHz processors on 11/15/03.
- IBM Federation switch (linking 32-way p690 nodes)
 - Part of Early Ship Program; pre-GA hardware delivered in October 2003.

Evaluation Methodology

“Measure early, measure often, analyze just in time”

- Hierarchical evaluation
 - Microbenchmarks
 - Application-relevant kernels
 - Compact or full parallel application codes
- Open evaluation
 - Rapid posting of evaluation results
 - Systems available to external performance researchers
- Fair evaluation
 - Determining appropriate way of using system, evaluating *both* traditional and alternative programming paradigms
 - Collecting data with *both* standard and custom benchmarks

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.



OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY


UT-BATTELLE

Ram

SGI Altix 3700 with 256 processors, configured as two 128-processor SMPs (i.e., 2 OS images)

- Itanium 2 processors running at 1.5 GHz
 - 16K L1 instruction cache, 16K L1 data cache
 - 256K L2 cache
 - 6MB L3 cache.
- 8 GB of memory per processor, for a total of 2 TB of shared memory
- 17 TB of disk space
- Four processor “compute brick”, made up of two 2-processor + memory nodes
- Fat-tree NUMAflex network, with support for MPI and SHMEM communication both within and between SMPs.
- Linux OS, Intel compilers

Other Platforms

- 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.
- HP/Compaq AlphaServer SC at Pittsburgh Supercomputing Center: 750 ES45 4-way SMP nodes (1GHz Alpha EV68) and a Quadrics QsNet interconnect with two network adapters per node.
- IBM p690 cluster at ORNL: 27 32-way p690 SMP nodes (1.3 GHz POWER4) and an SP Switch2 with two to eight network adapters per node.
- IBM SP at the National Energy Research Supercomputer Center (NERSC): 184 Nighthawk II 16-way SMP nodes (375MHz POWER3-II) and an SP Switch2 with two network adapters per node.
- SGI Origin 3000 at Los Alamos National Laboratory (LANL): 512-way SMP node. Each processor is a 500 MHz MIPS R14000.

Outline

Quick sampling of current results, using

- POP parallel application
- COMMTEST microbenchmark
- PSTSWM serial kernel

For more performance data, visit

<http://www.csm.ornl.gov/evaluation>

Caveats

- These are EARLY results (even on the Cray after 6 months), resulting from sporadic benchmarking on evolving system software and hardware configurations.
- Performance characteristics are still changing, due to continued evolution of OS and compilers and libraries.

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.
- Optimization on the X1 ongoing.

POP Experiment Particulars

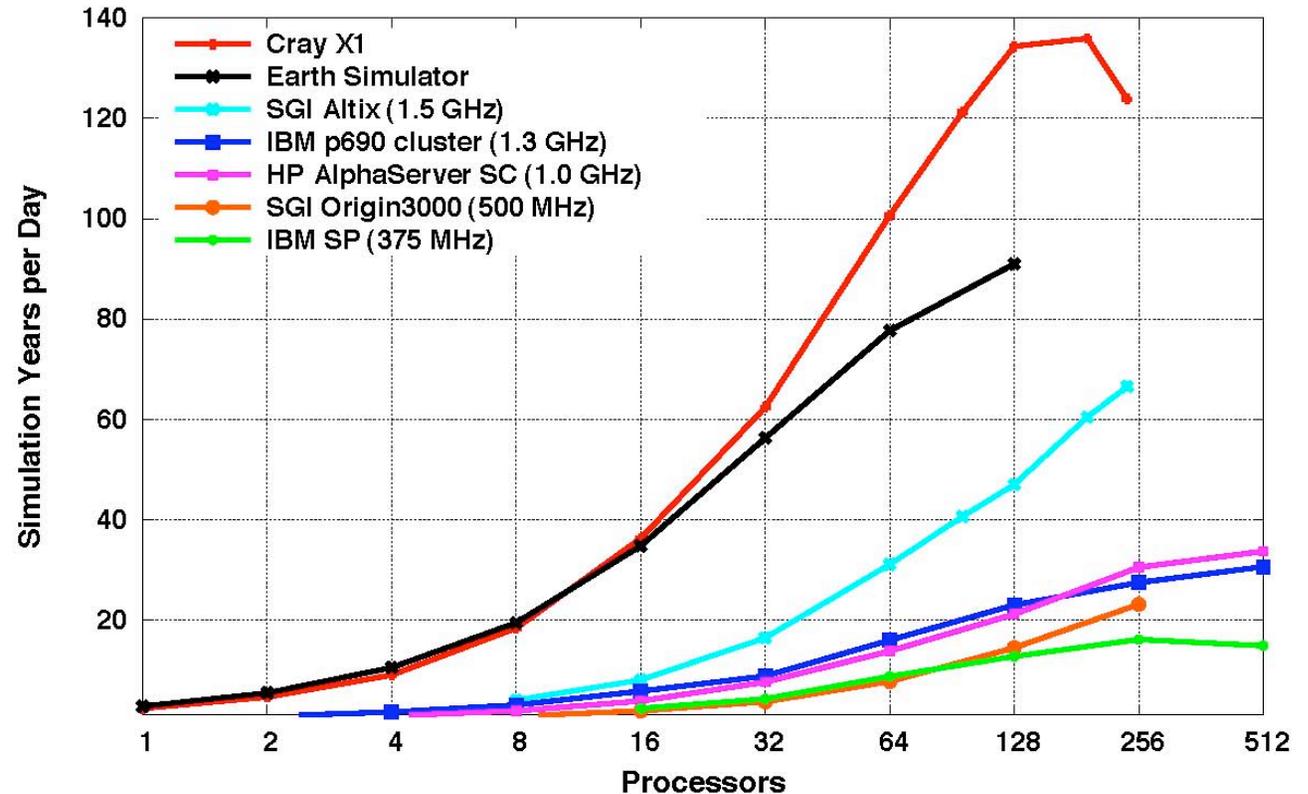
- Two primary computational phases
 - Baroclinic: 3D with limited nearest-neighbor communication; scales well.
 - Barotropic: dominated by solution of 2D implicit system using conjugate gradient solves; scales poorly
- One benchmark problem size
 - One degree horizontal grid (“by one” or “x1”) of size 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

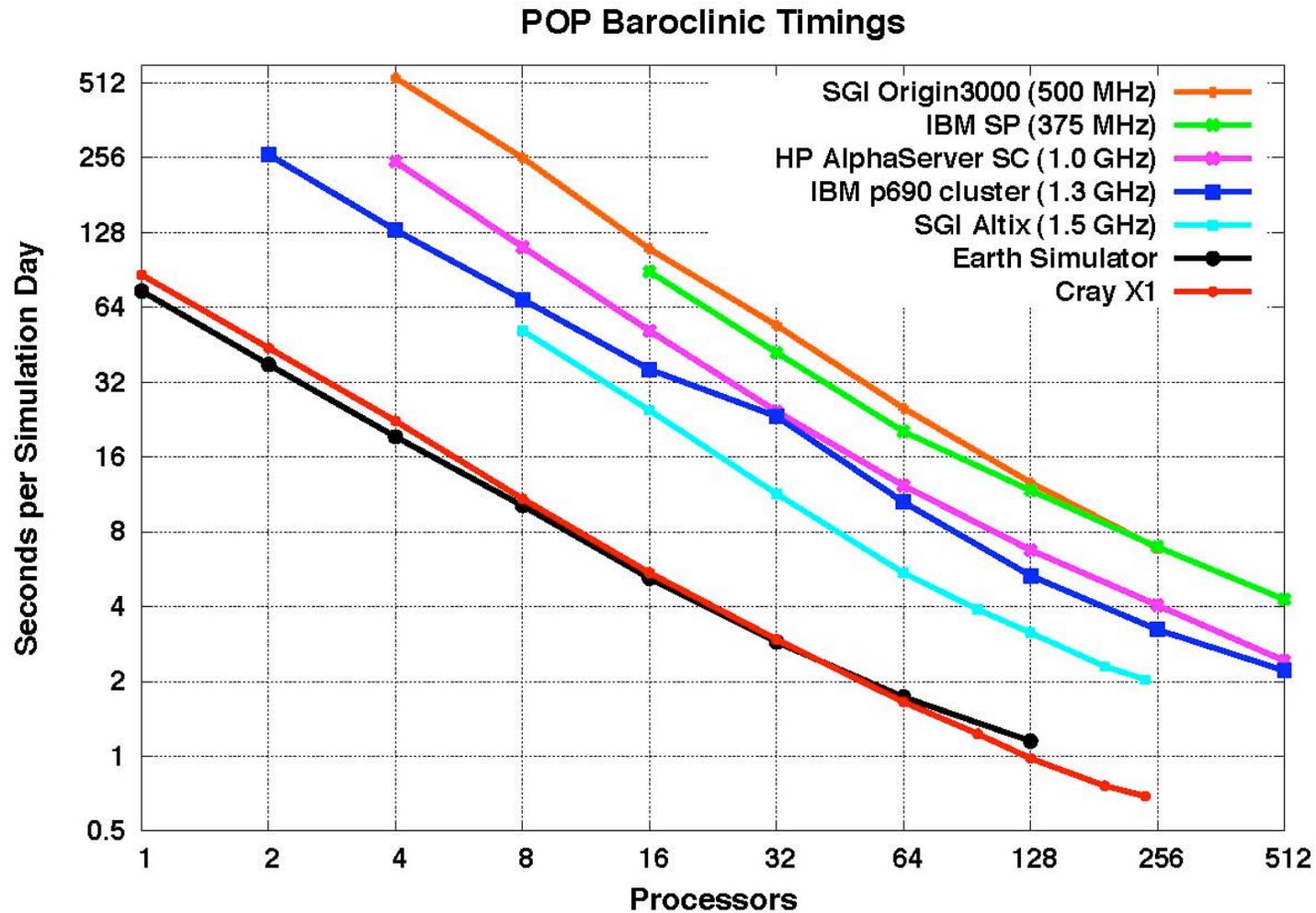
Comparing performance and scaling across platforms.

- Earth Simulator results courtesy of Dr. Y. Yoshida of the Central Research Institute of Electric Power Industry (CRIEPI).
- SGI Origin results courtesy of Dr. P. Jones of LANL.
- IBM SP results courtesy of Dr. T. Mohan of Lawrence Berkeley National Laboratory (LBNL)

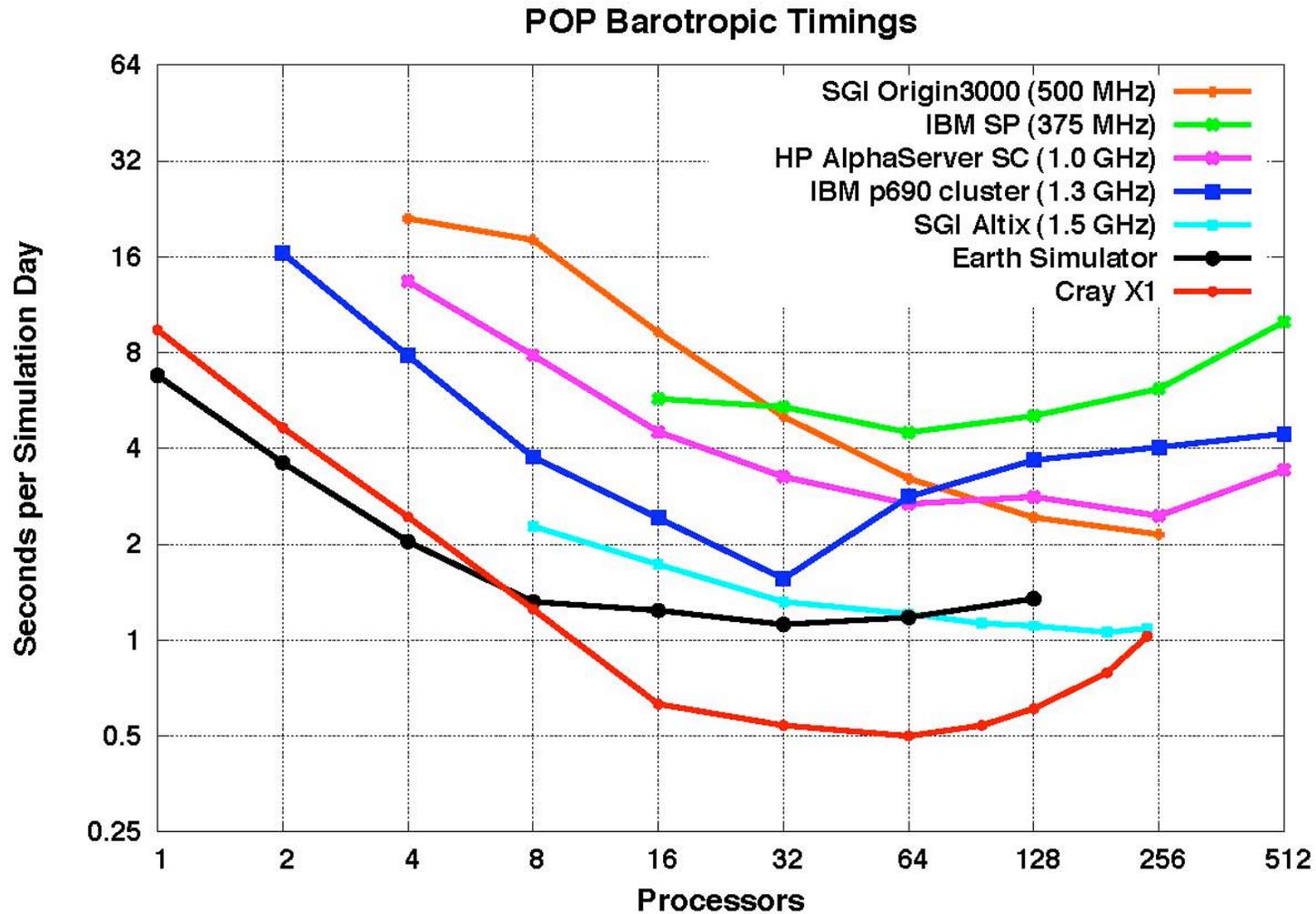
Parallel Application Performance
LANL Parallel Ocean Program, x1 benchmark



POP Performance Diagnosis: Baroclinic



POP Performance Diagnosis: Barotropic



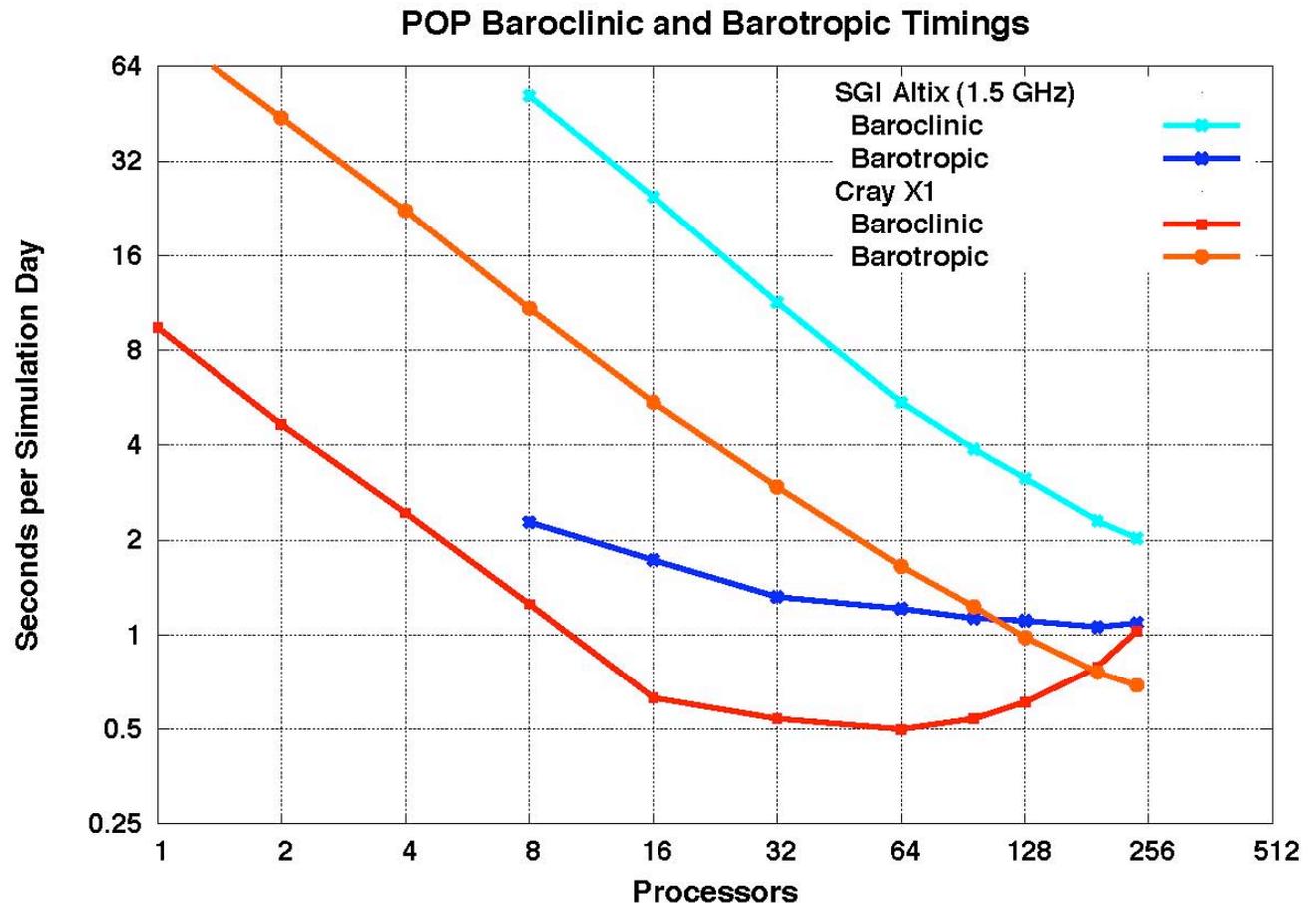
POP Performance Diagnosis

Cray X1

Communication-bound for more than 192 processors, with communication costs increasing. Communication algorithms known to have scaling problems, and alternatives being developed.

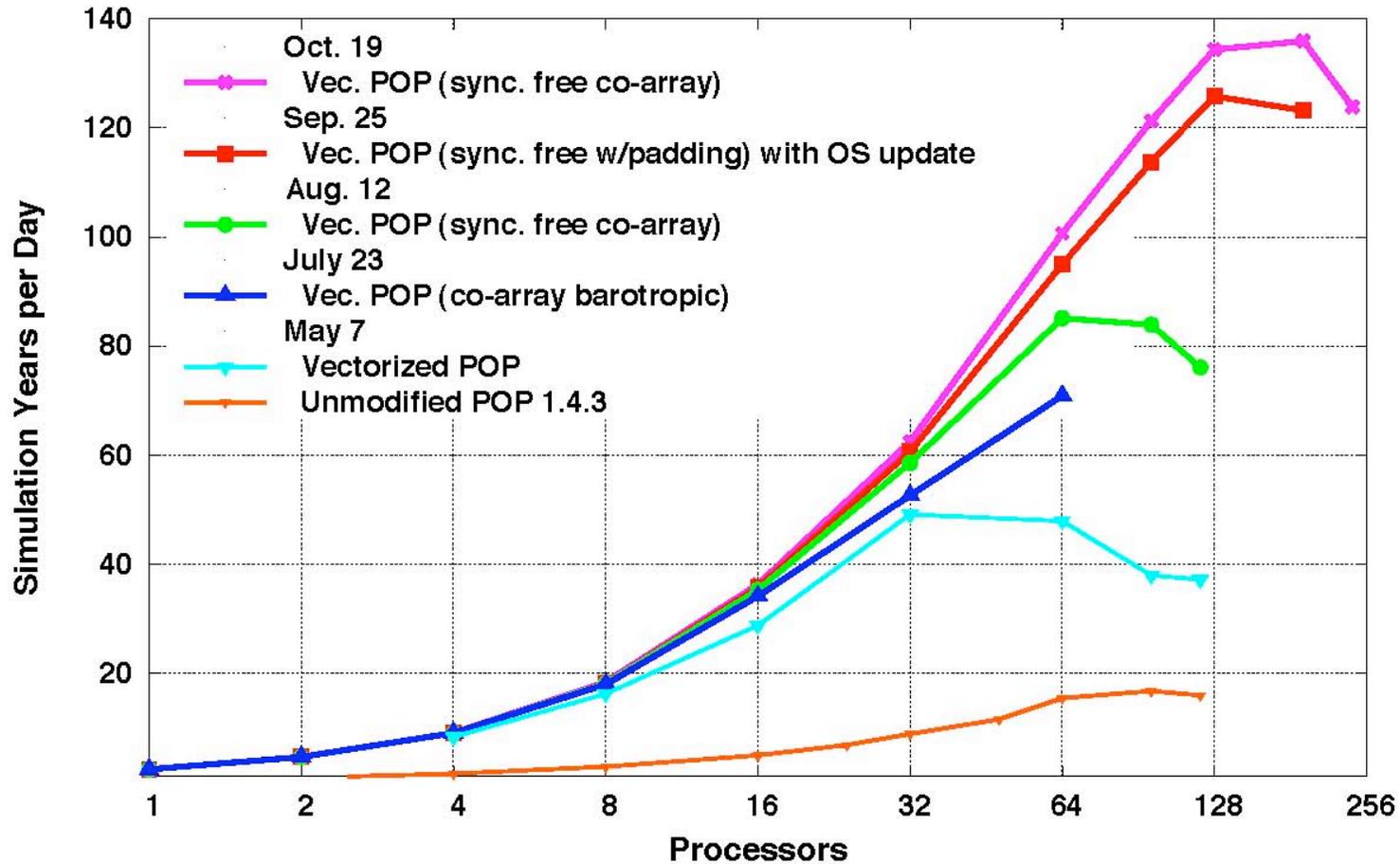
SGI Altix

Not yet communication bound. Using MPI point-to-point and collectives for barotropic. Initial experiments with do not show significant improvement.



POP Performance Evolution on the X1

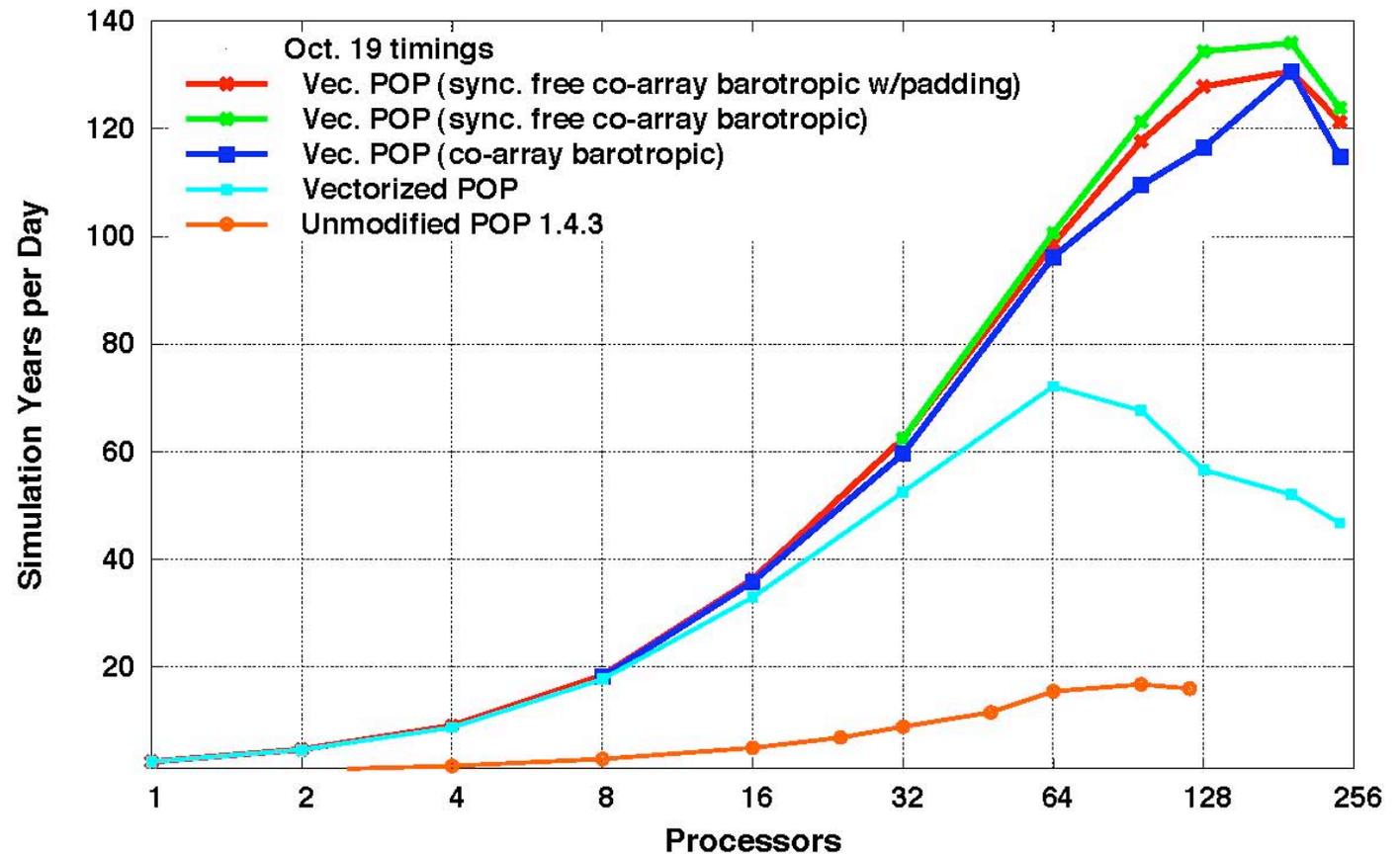
LANL Parallel Ocean Program
POP 1.4.3, x1 benchmark



POP Implementation Comparison on the X1

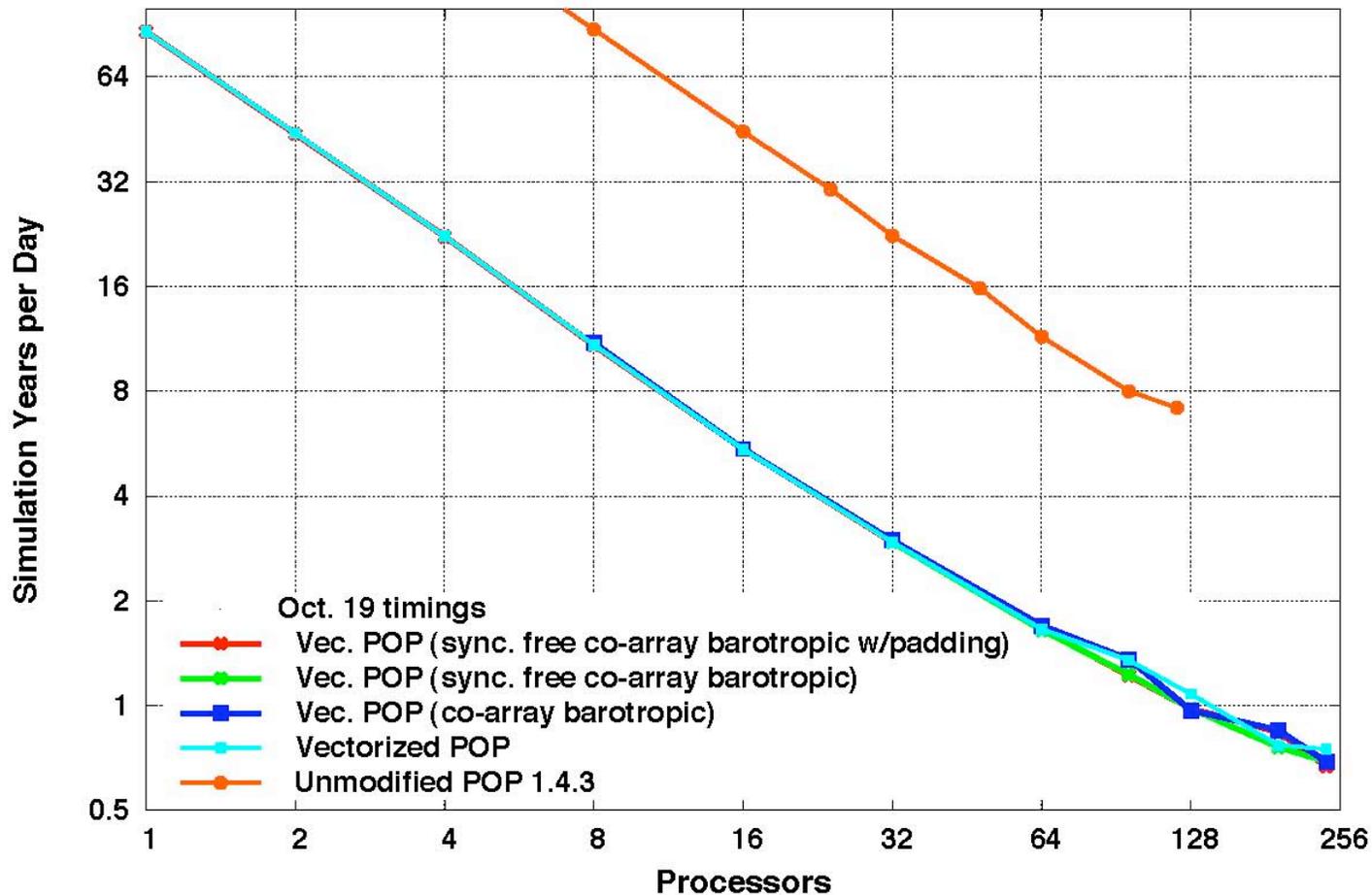
Much of recent algorithm development driven by OS performance problems. Once OS problems solved, algorithmic differences less significant. MPI performance still poor for latency sensitive Algorithms, and restructuring for vectorization still vital.

LANL Parallel Ocean Program
POP 1.4.3, x1 benchmark

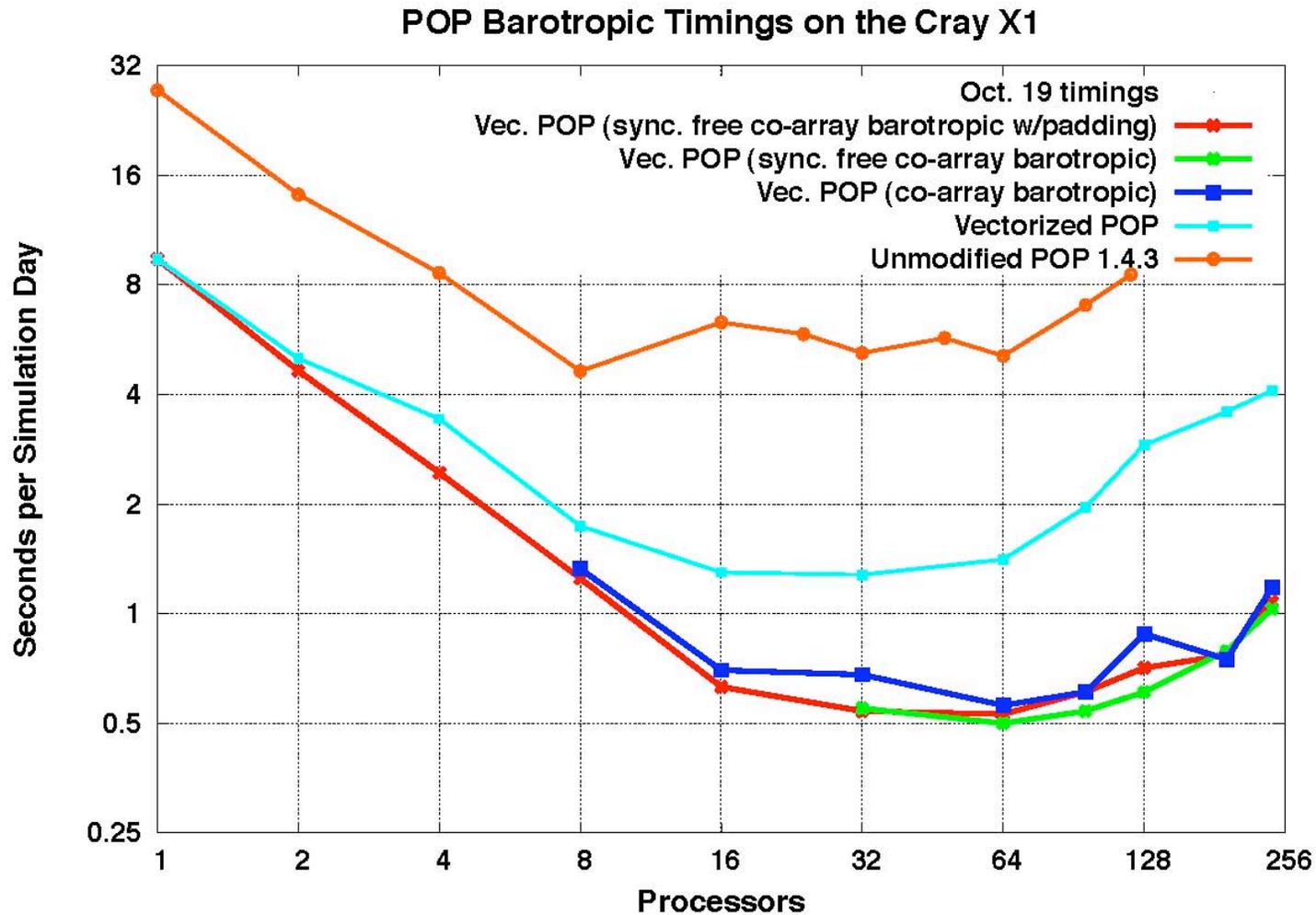


POP Implementation Comparison on the X1

POP Baroclinic Timings on the Cray X1



POP Implementation Comparison on the X1



What's next for POP?

- Additional Cray X1 optimizations
 - Scalable (tree-based) allreduce
 - Portable Co-array Fortran
 - Cray-specific vectorization
- Additional SGI Altix optimizations
 - Careful evaluation of SHMEM-based barotropic
- Increased resolution
 - 0.1 degree resolution
- More recent versions of the model
 - CCSM version of POP 1.4.3
 - POP 2.0
 - HYPOP

COMMTEST Benchmark

COMMTEST is a suite of codes that measure the performance of MPI interprocessor communication. In particular, COMMTEST evaluates the impact of communication protocol, packet size, and total message length in a number of “common usage” scenarios. (However, it does not include persistent MPI point-to-point commands among the protocols examined.) It also includes simplified implementations of the SWAP and SENDRECV operators using SHMEM.

COMMTEST Experiments

i-j

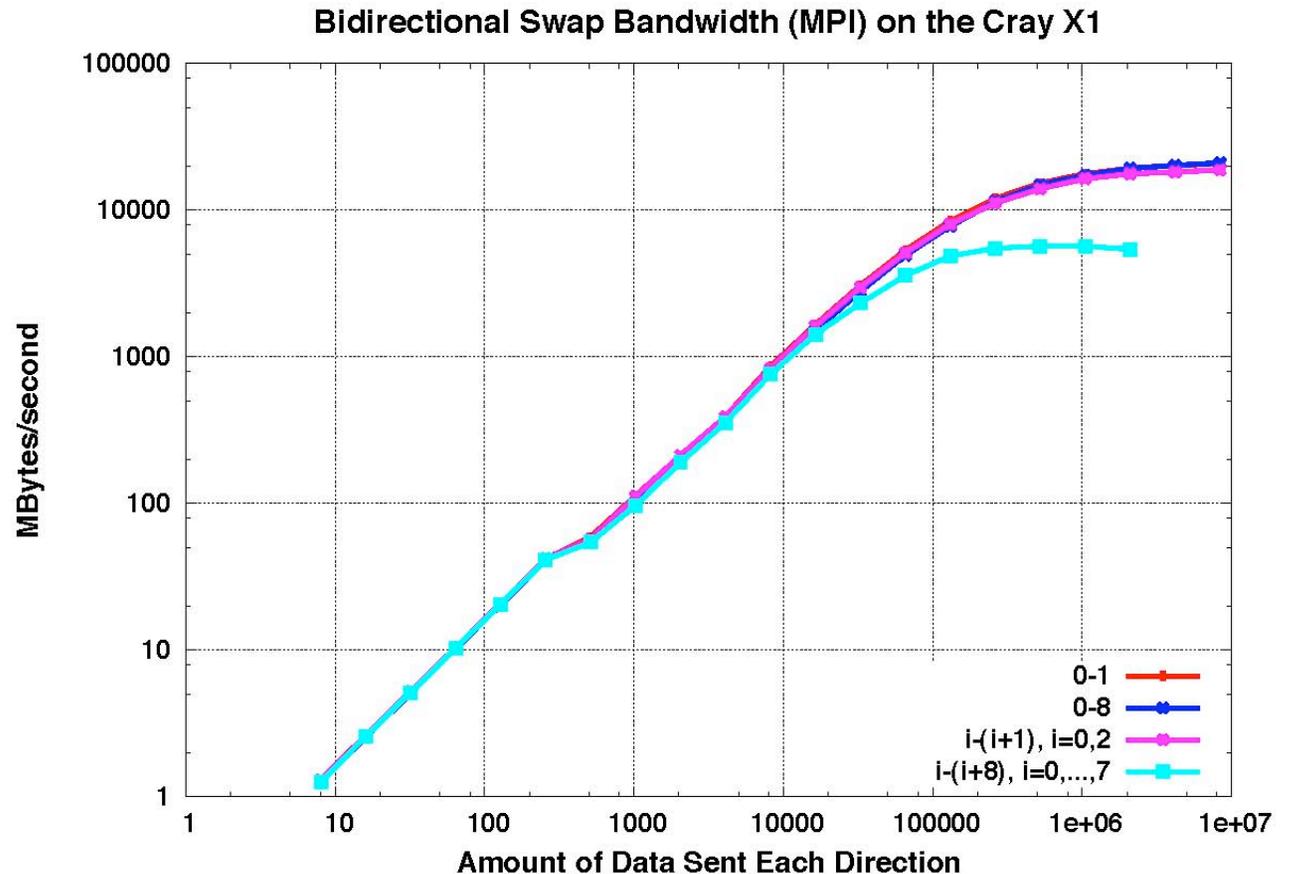
processor i swaps data with processor j. Depending on i and j, this can be within an SMP node or between SMP nodes.

i-(i+j), i=1,n

n processor pairs (i,j) swap data simultaneously. Depending on j, this will be within an SMP node or between SMP nodes (or both).

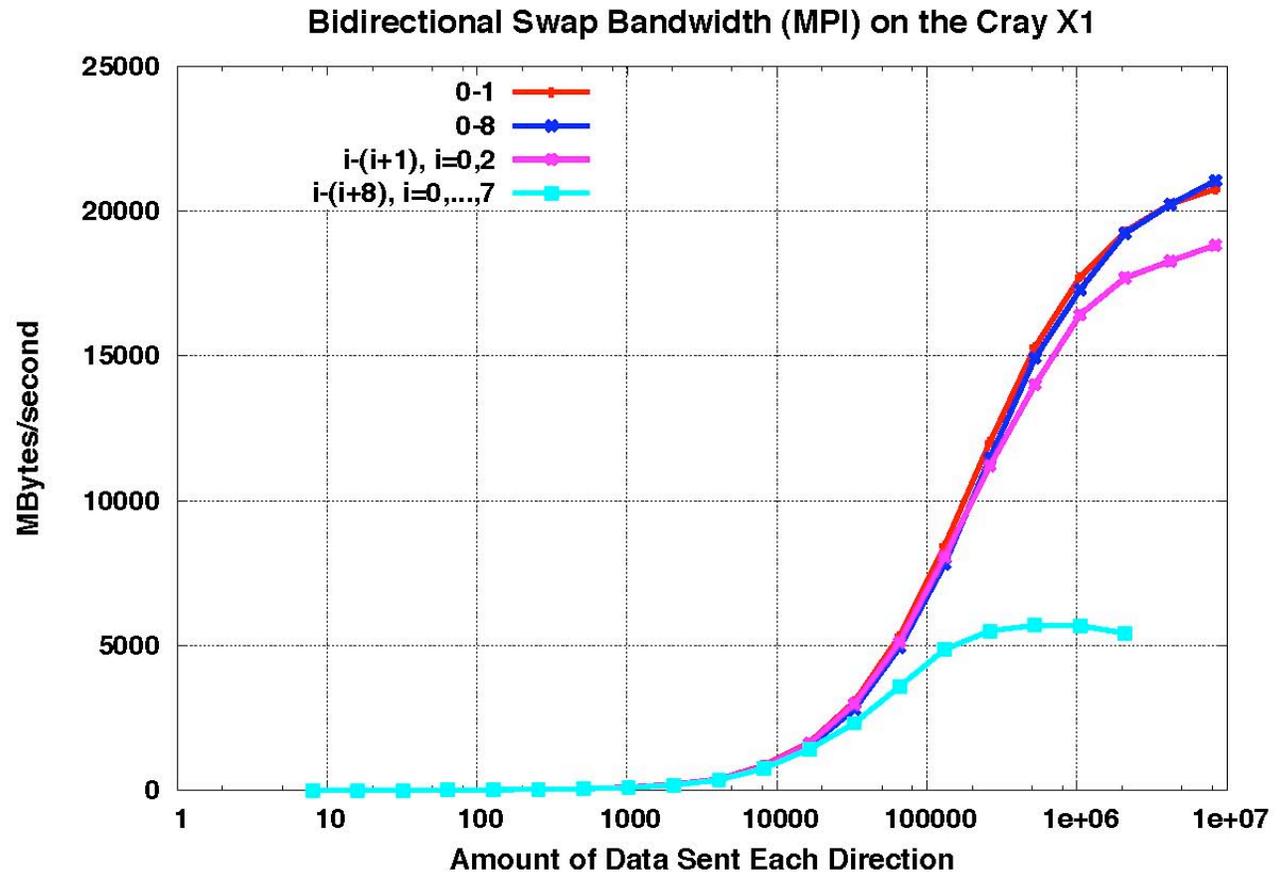
COMMTEST SWAP Benchmark on X1

Comparing performance of SWAP for different communication patterns. All performance is similar except for the experiment in which 8 pairs of processors swap simultaneously. In this case, contention for internode bandwidth limits the single pair bandwidth.



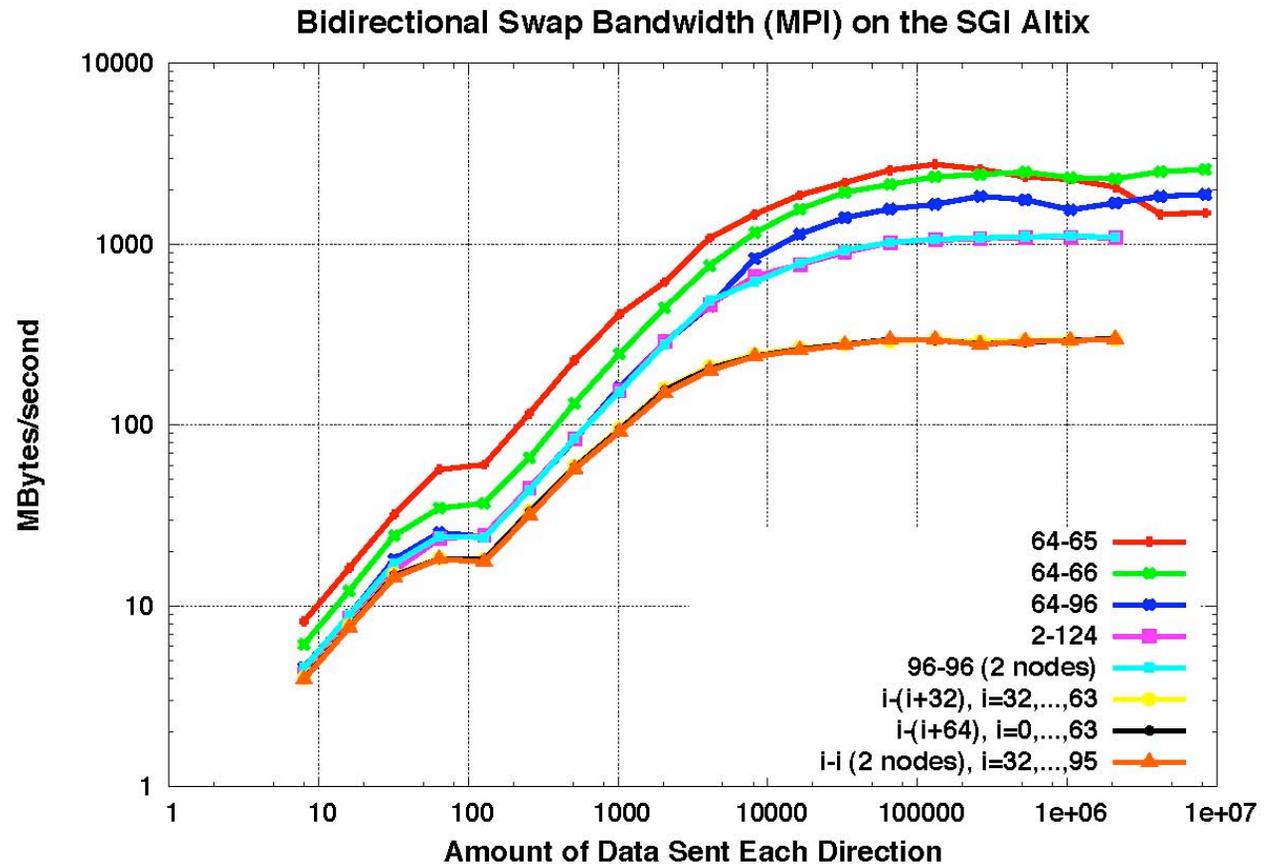
COMMTEST SWAP Benchmark on X1

Comparing performance of SWAP for different communication pattern, plotted on a log-linear scale. The single pair bandwidth has not reached its peak yet, but the two pair experiment bandwidth is beginning to reach its maximum.



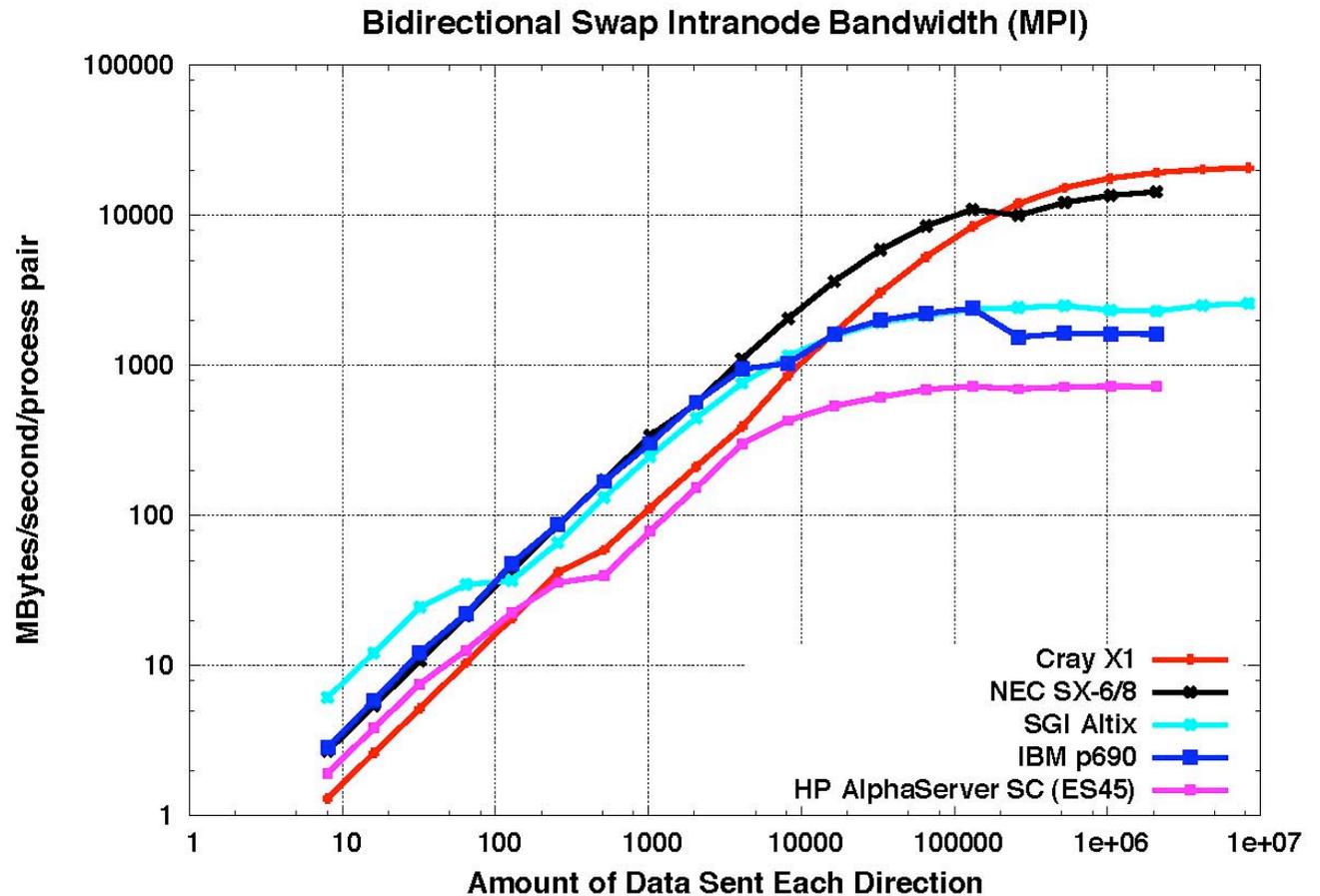
COMMTEST SWAP Benchmark on Altix

Comparing performance of SWAP for different communication pattern, plotted on a log-linear scale. The single pair bandwidth has not reached its peak yet, but the two pair experiment bandwidth is beginning to reach its maximum.



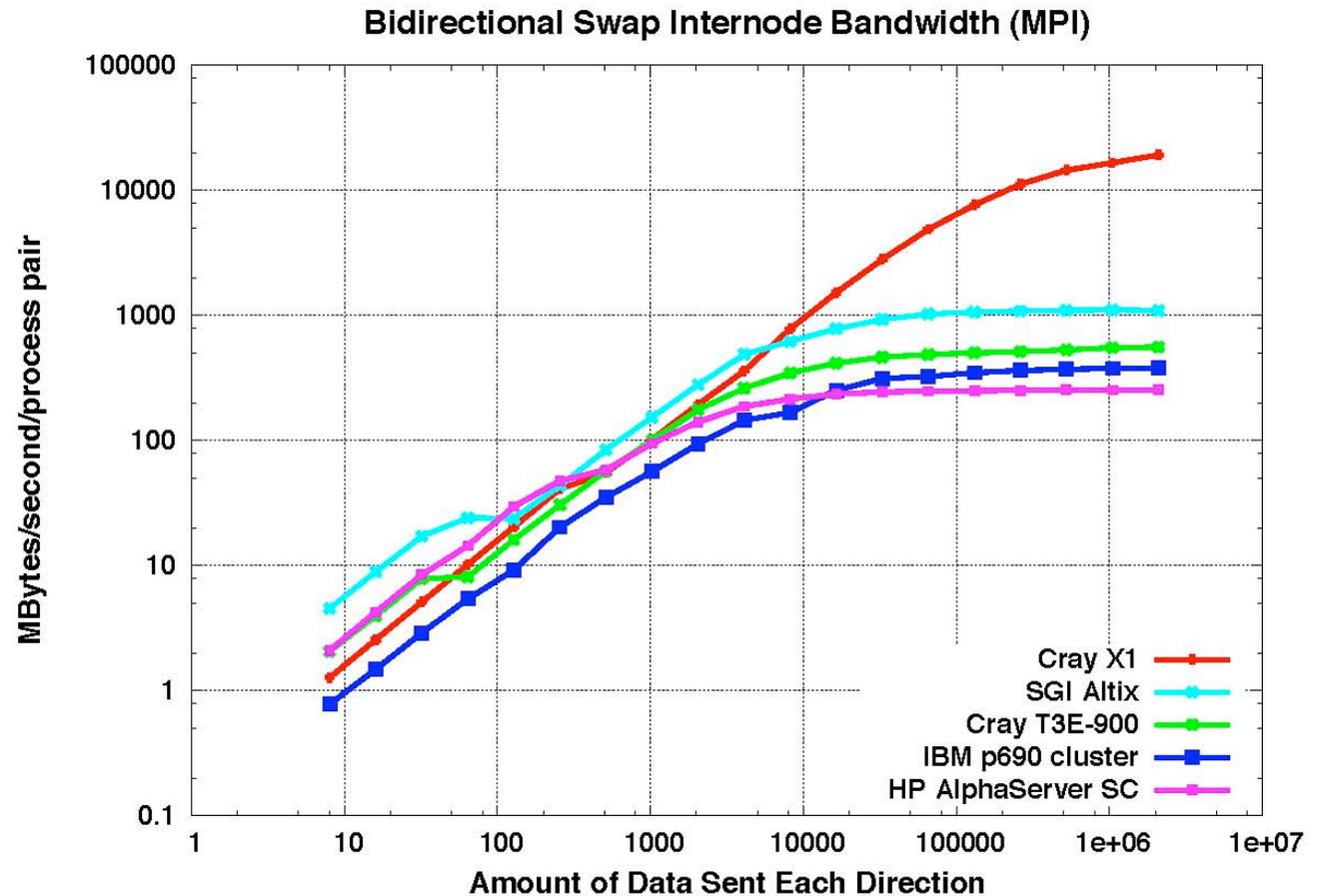
COMMTEST SWAP Benchmark

Comparing performance of SWAP for different platforms. Experiment measures bidirectional bandwidth between two processors in the same SMP node.



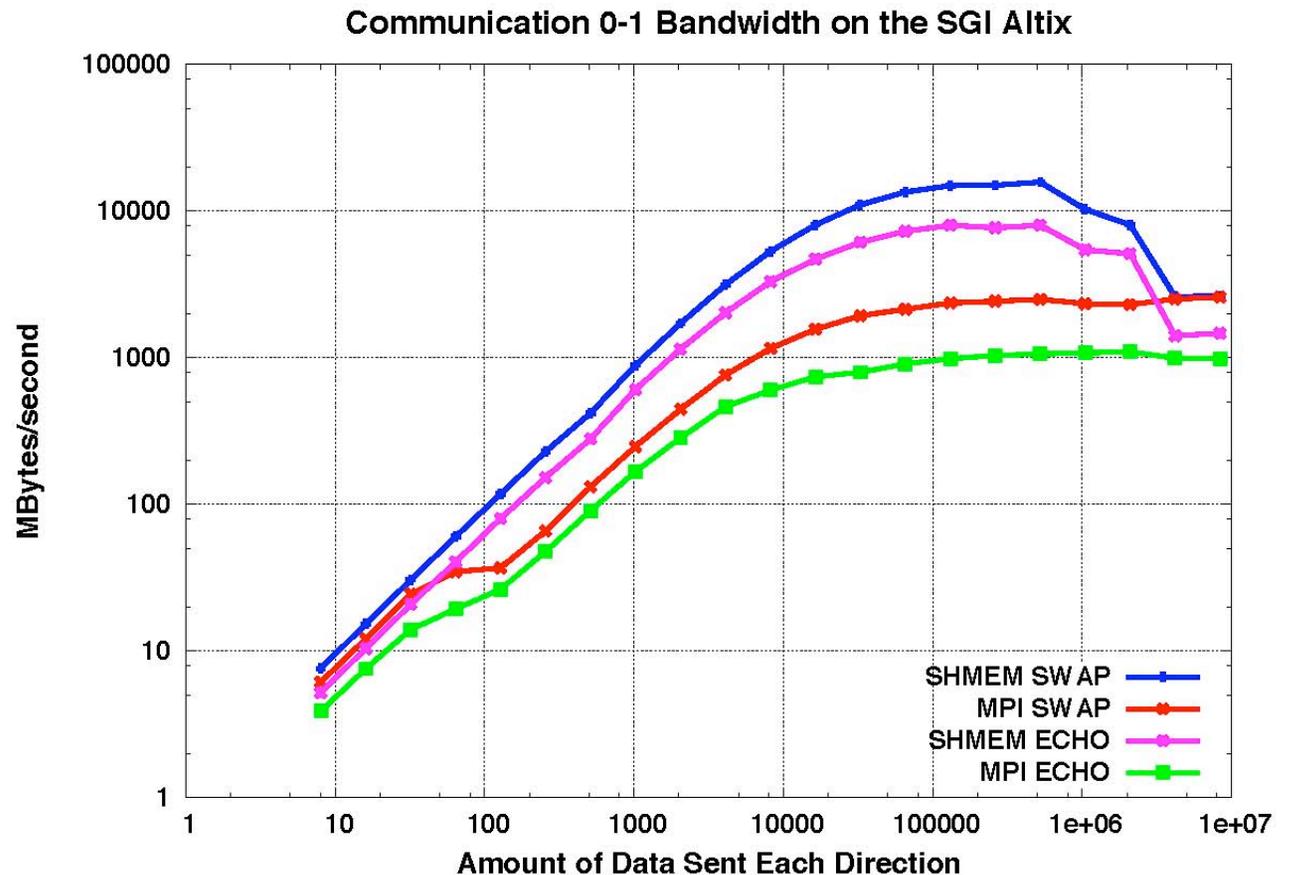
COMMTEST SWAP Benchmark

Comparing performance of SWAP for different platforms. Experiment measures bidirectional bandwidth between two processors in different SMP nodes.



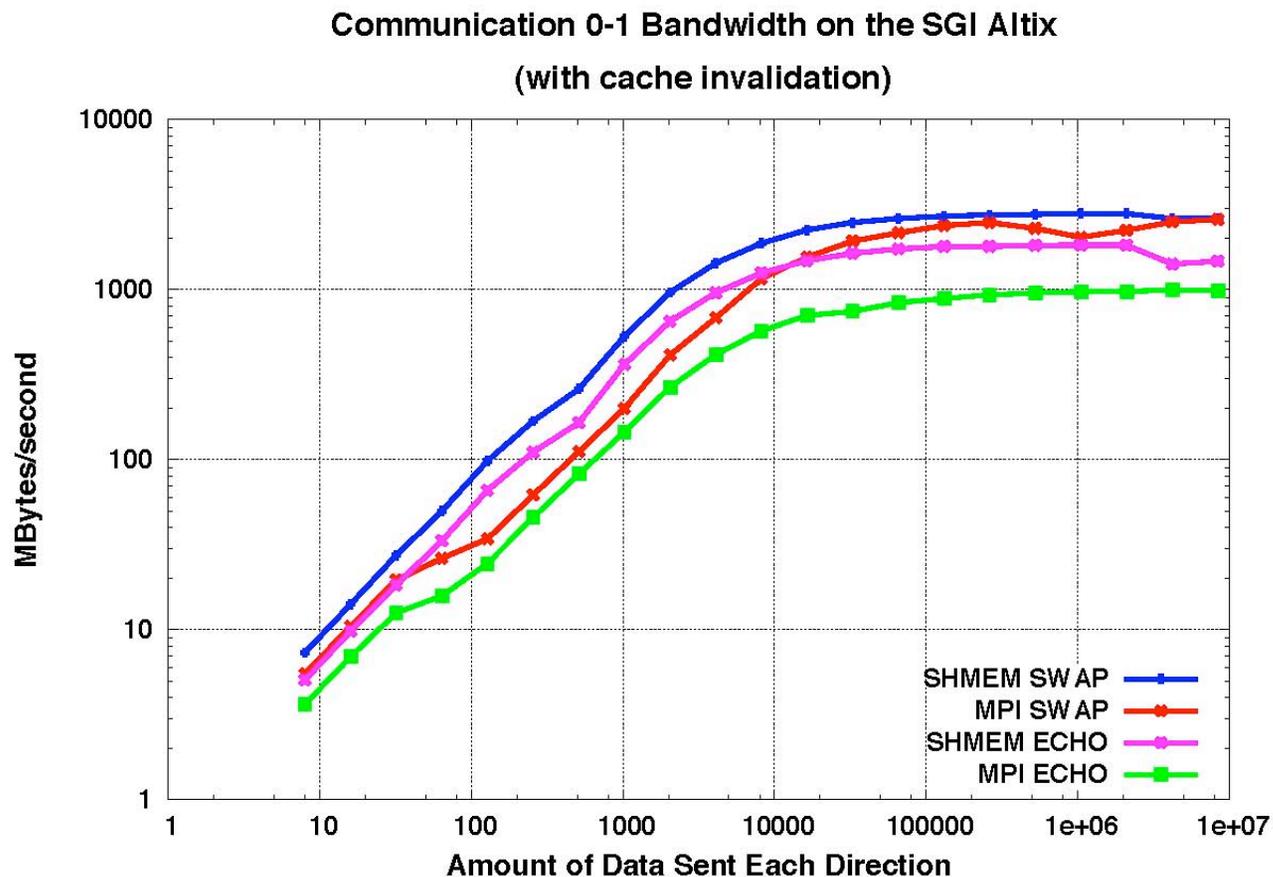
MPI vs. SHMEM 64-66 Comparison on Altix

Comparing MPI and SHMEM performance for 64-66 experiment, looking at both SWAP (bidirectional bandwidth) and ECHO (unidirectional bandwidth). SHMEM performance is better for all message sizes.



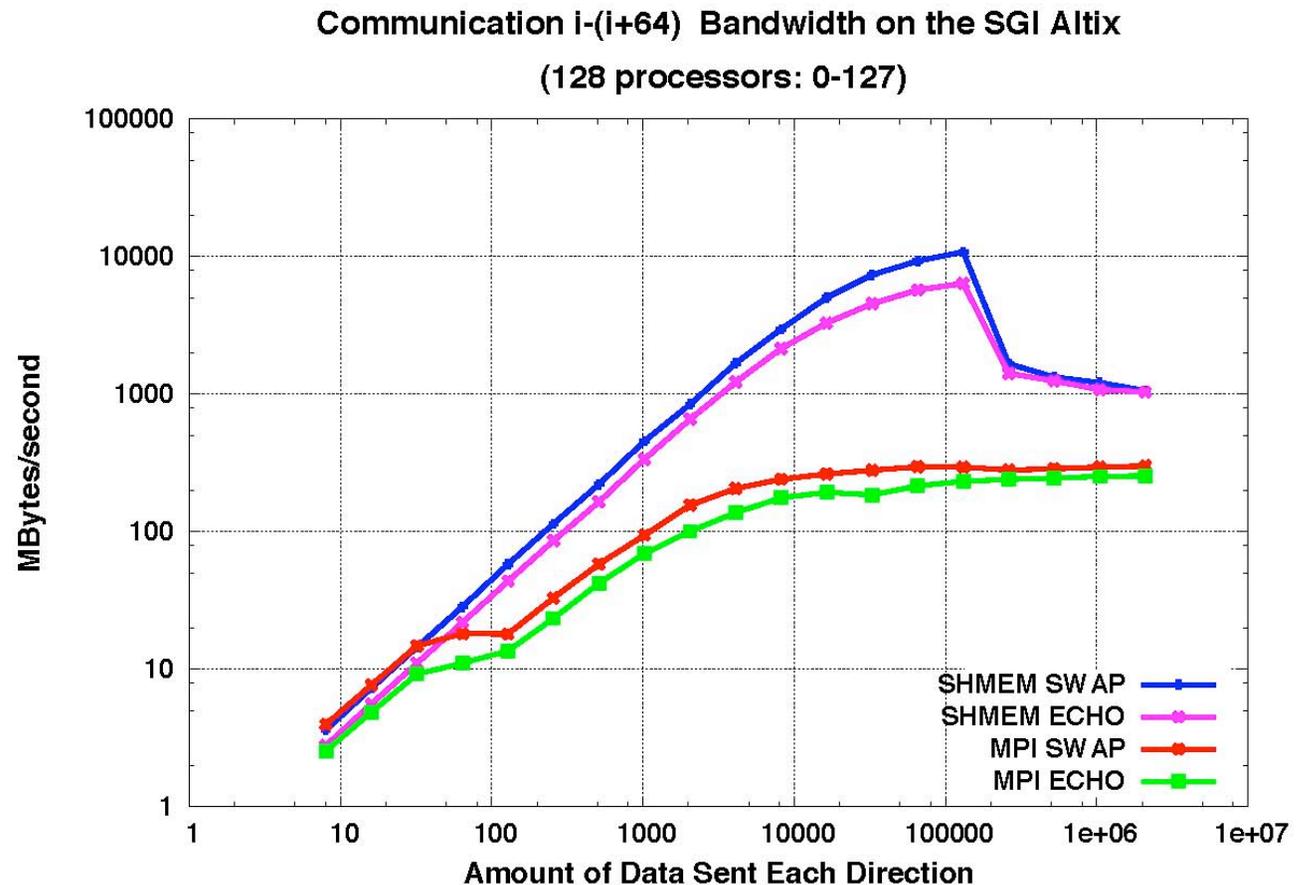
MPI vs. SHMEM 64-66 Comparison on Altix

Comparing MPI and SHMEM performance for 64-66 experiment when invalidating the cache first, looking at both SWAP (bidirectional bandwidth) and ECHO (unidirectional bandwidth). SHMEM performance is still better for all message sizes, but not by as much.



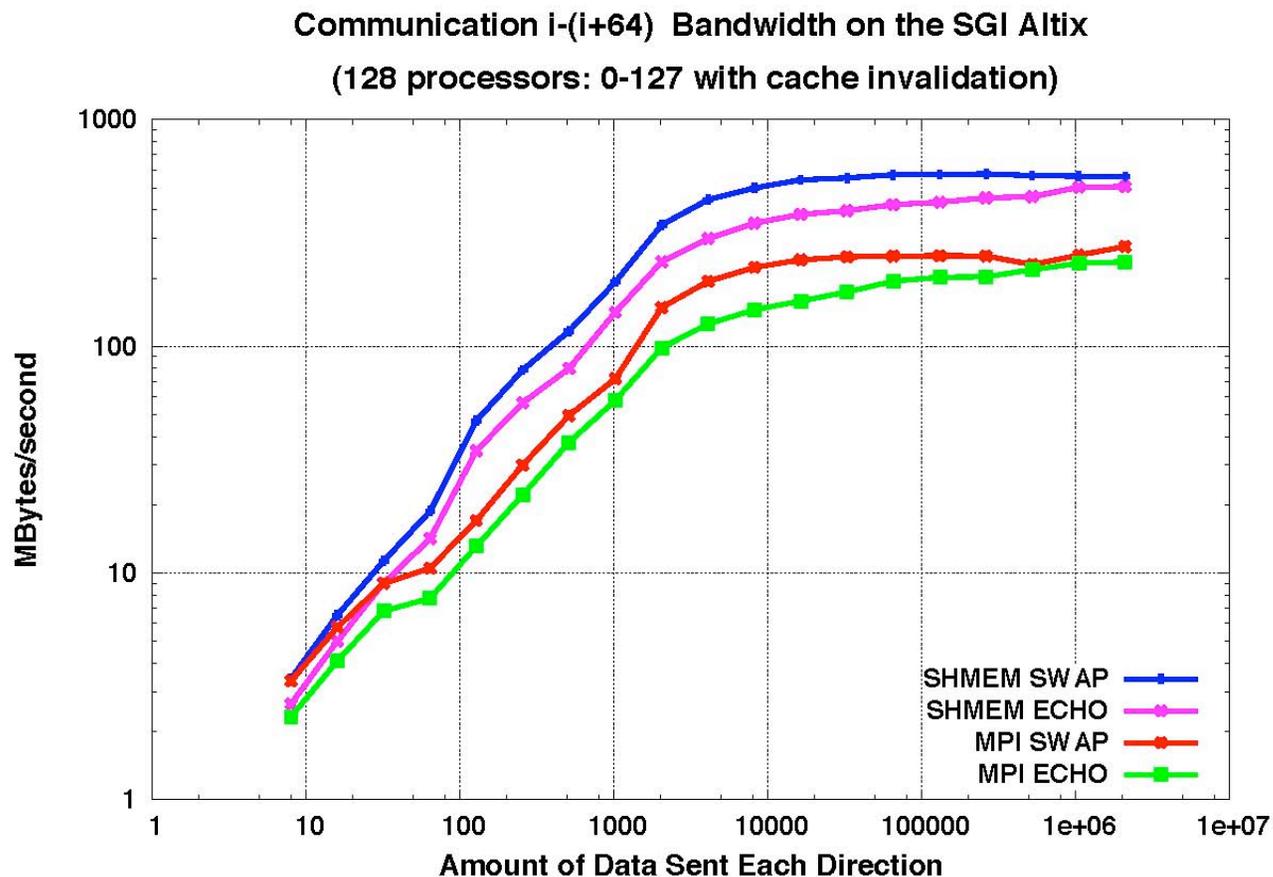
MPI vs. SHMEM $i-(i+64)$ Comparison on Altix

Comparing MPI and SHMEM performance for $i-(i+64)$ experiment, looking at both SWAP (bidirectional bandwidth) and ECHO (unidirectional bandwidth). Again, SHMEM performance is better for all message sizes.



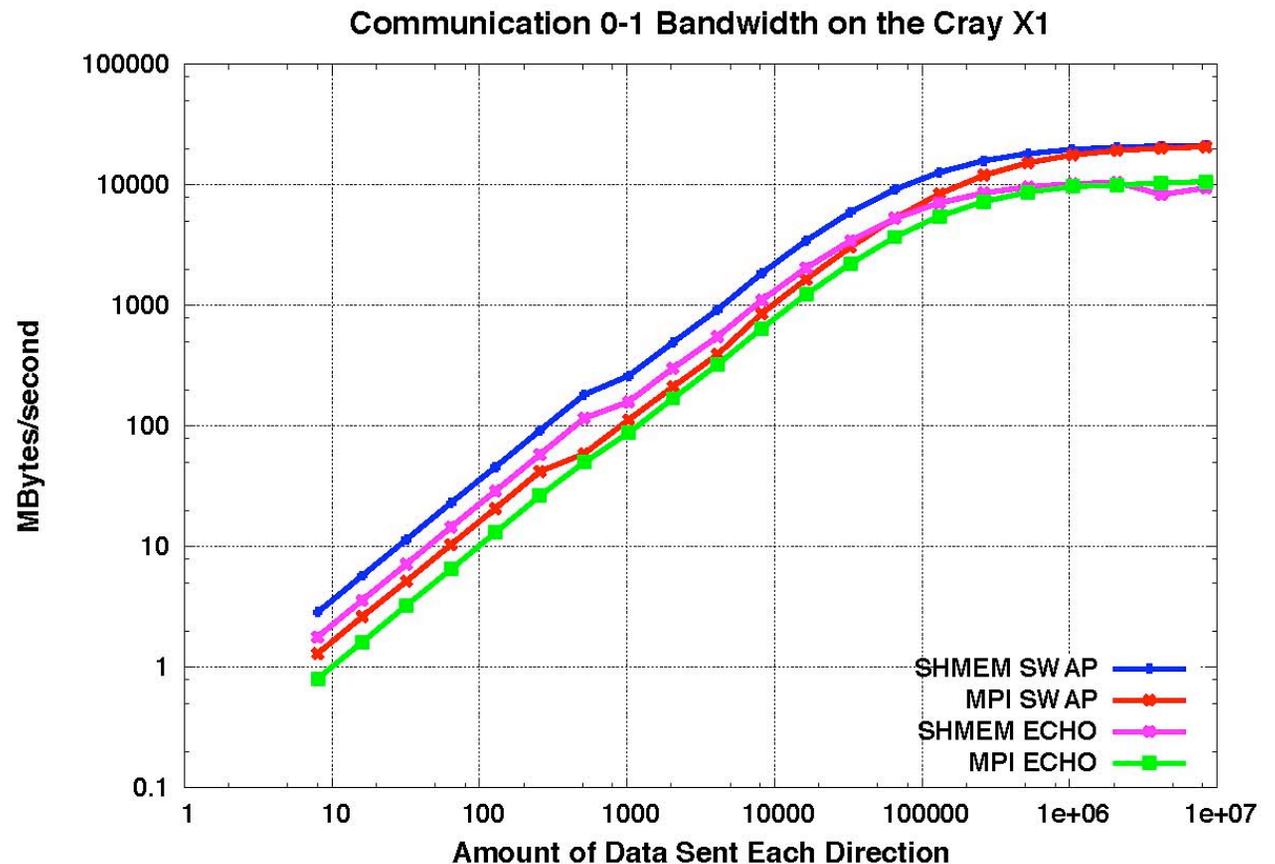
MPI vs. SHMEM i-(i+64) Comparison on Altix

Comparing MPI and SHMEM performance for i-(i+64) experiment with cache invalidation, looking at both SWAP (bidirectional bandwidth) and ECHO (unidirectional bandwidth). Again, SHMEM performance is better for all message sizes, although not by as much as without invalidating the cache.



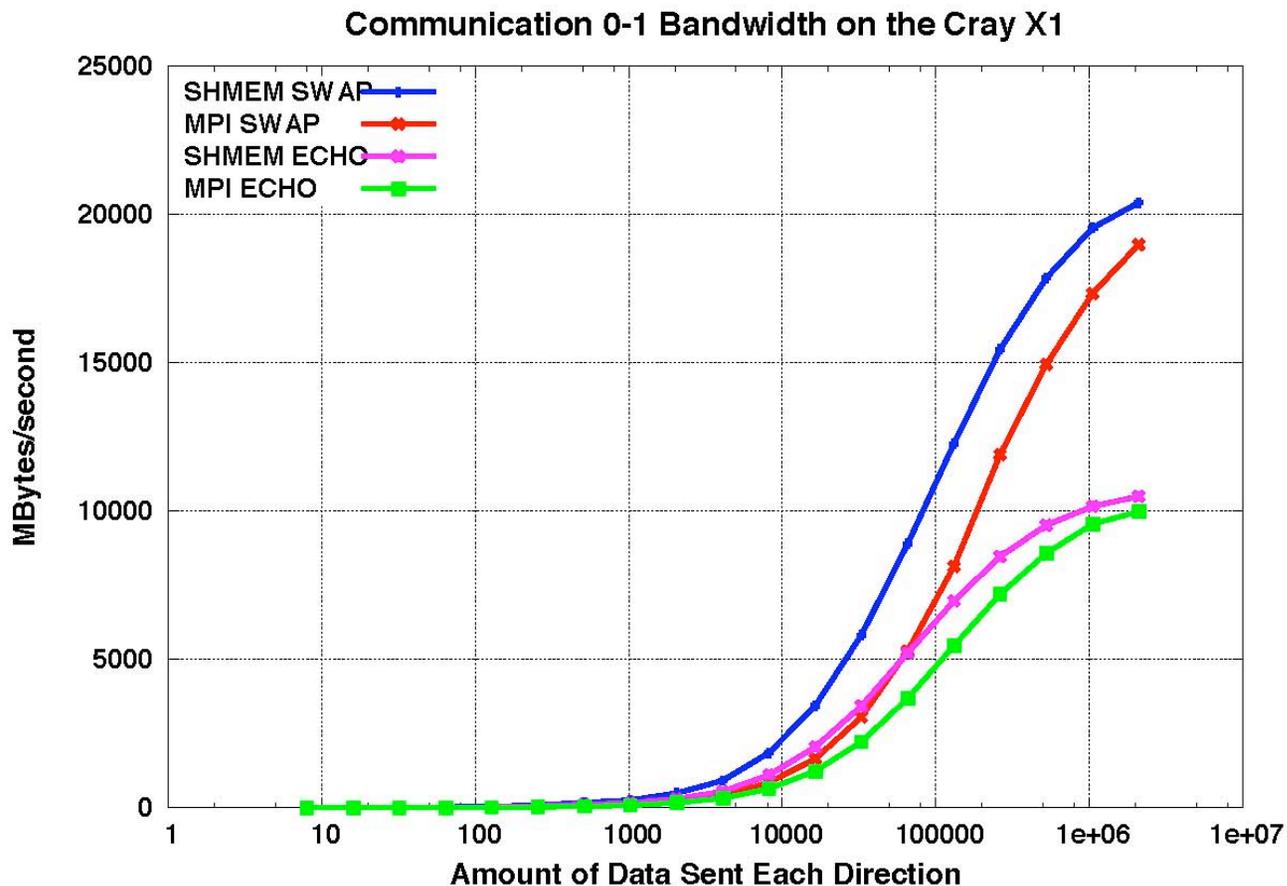
MPI vs. SHMEM 0-1 Comparison on X1

Comparing MPI and SHMEM performance for 0-1 experiment, looking at both SWAP (bidirectional bandwidth) and ECHO (unidirectional bandwidth). SHMEM performance is better for all but the largest messages.



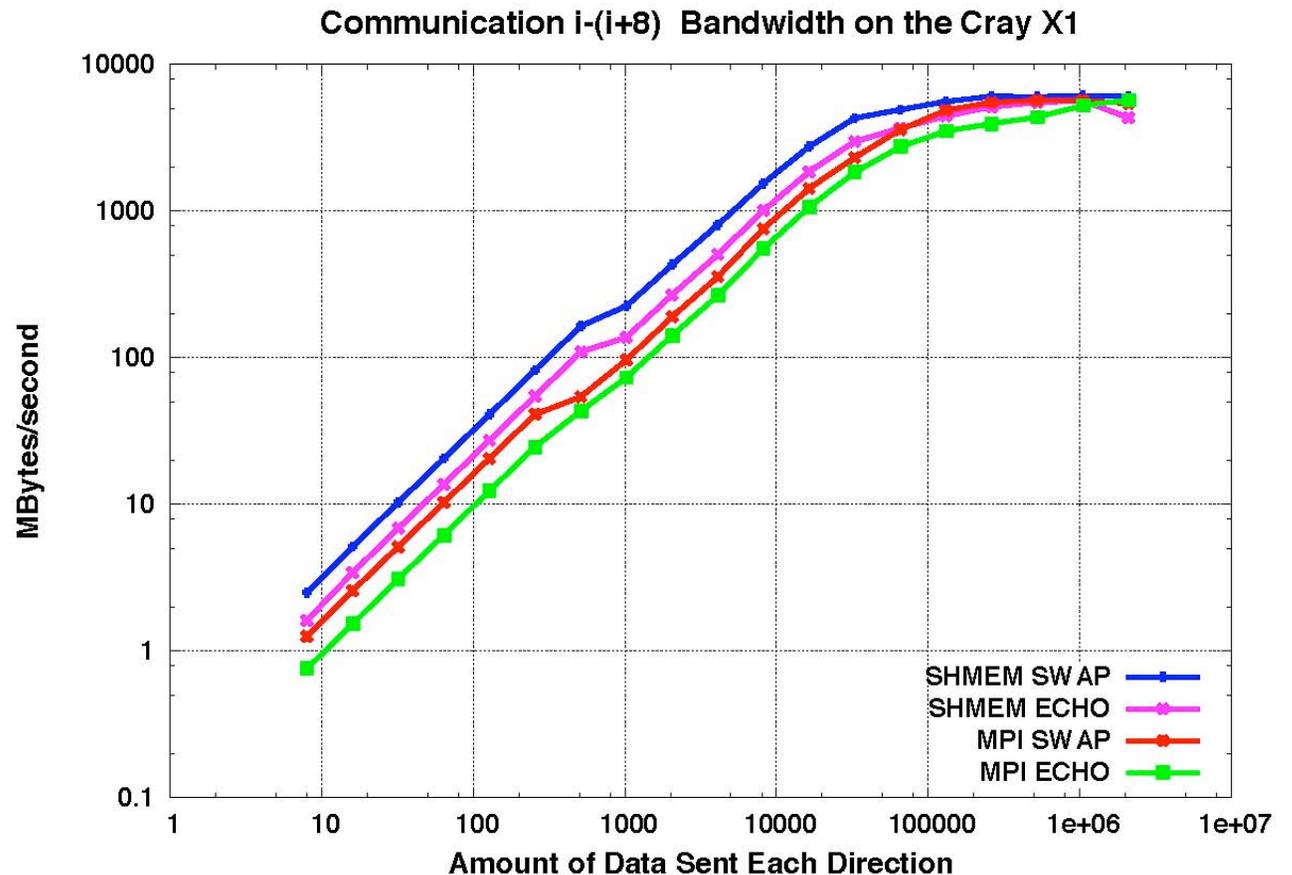
MPI vs. SHMEM 0-1 Comparison on X1

Comparing MPI and SHMEM performance for 0-1 experiment, using a log-linear scale. MPI performance is very near to that of SHMEM for large messages (when using SHMEM to implement two-sided messaging).



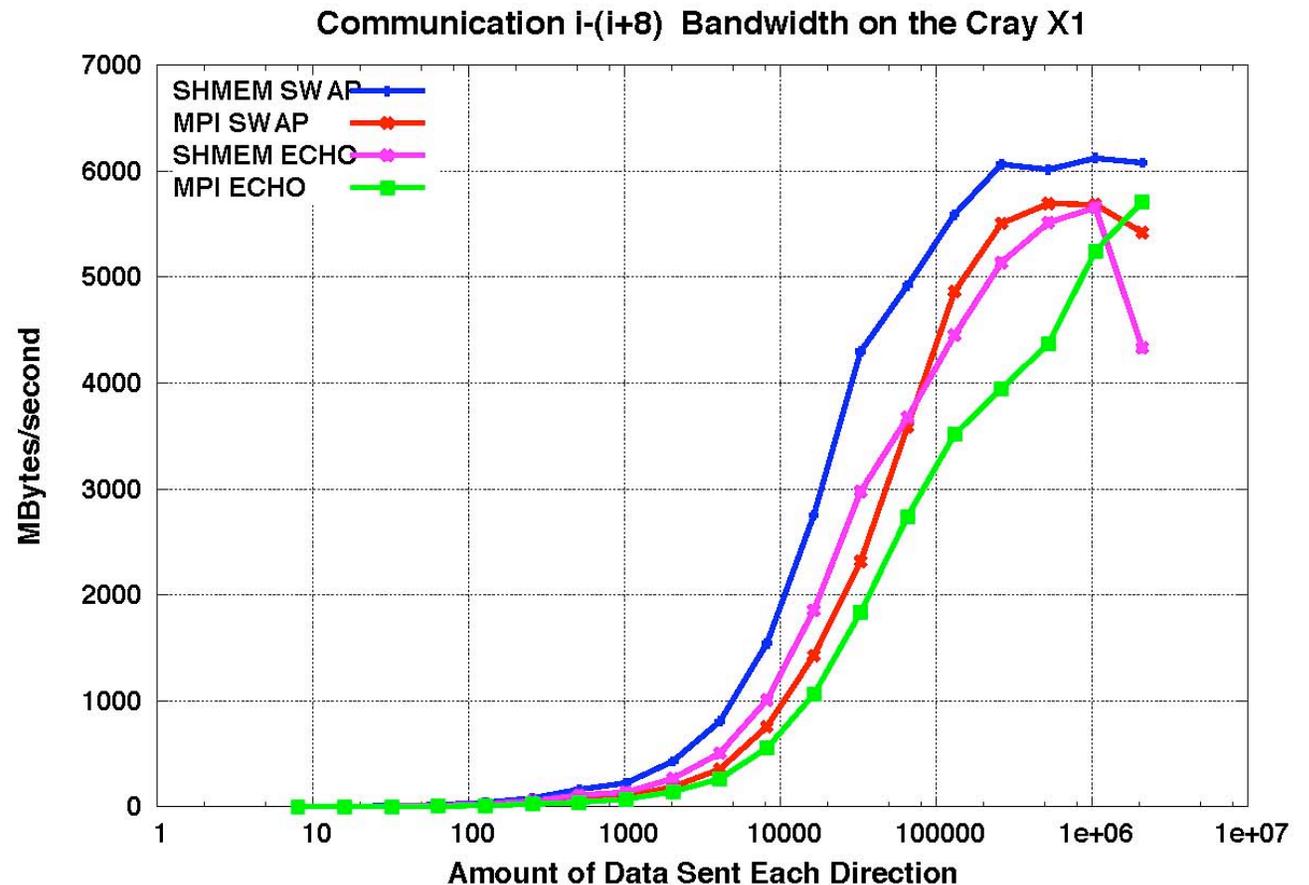
MPI vs. SHMEM i-(i+8) Comparison on X1

Comparing MPI and SHMEM performance for i-(i+8) experiment, looking at both SWAP (bidirectional bandwidth) and ECHO (unidirectional bandwidth). Again, SHMEM performance is better for all but the largest messages.



MPI vs. SHMEM $i-(i+8)$ Comparison on X1

Comparing MPI and SHMEM performance for $i-(i+8)$ experiment, using a log-linear scale. MPI performance is very near to that of SHMEM for large messages (when using SHMEM to implement two-sided messaging). For the largest message sizes, MPI ECHO bandwidth exceeds MPI SWAP bandwidth.



PSTSWM Description

The Parallel Spectral Transform Shallow Water Model represents an important computational kernel in spectral global atmospheric models. As 99% of the floating-point operations are multiply or add, it runs well on systems optimized for these operations. PSTSWM exhibits little reuse of operands as it sweeps through the field arrays; thus it exercises the memory subsystem as the problem size is scaled and can be used to evaluate the impact of memory contention in SMP nodes. PSTSWM is also a parallel algorithm testbed, and all array sizes and loop bounds are determined at runtime. This makes it difficult for the X1 compiler to identify which loops to vectorize or stream.

PSTSWM Experiment Particulars

These experiments examine serial performance, both using one processor and running the serial benchmark on multiple processors simultaneously. Performance is measured for a range of horizontal problems resolutions for 1, 18, and 66 vertical levels.

Horizontal Resolutions

T5:	8 x 16
T10:	16 x 32
T21:	32 x 64
T42:	64 x 128
T85:	128 x 256
T170:	256 x 512
T340:	512 x 1024
T680:	1024 x 2048

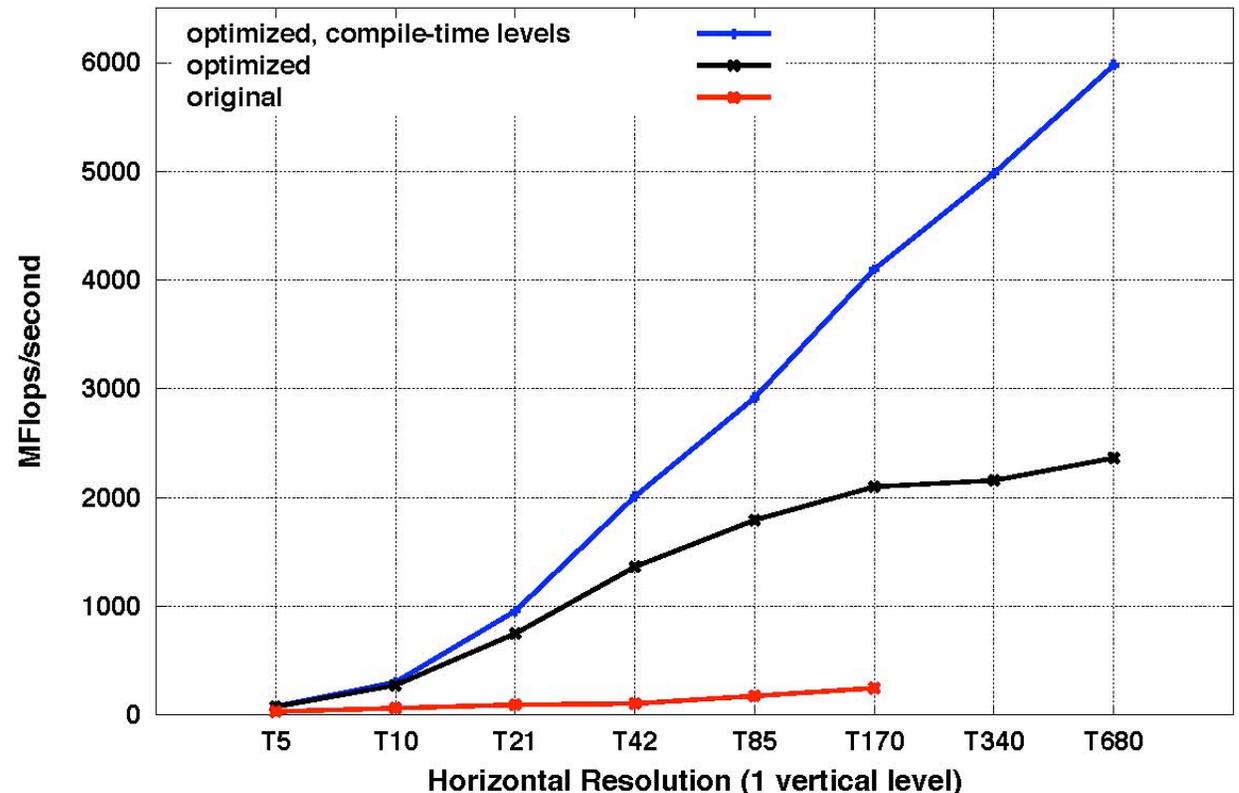
PSTSWM Code Versions

- Original (unvectorized) code
- Port to X1
 - changing loops and local array definitions for select routines
- Port to X1 with compile-time specification of number of vertical levels
- Math libraries used for block FFTs when available (and when performance enhancers compared to bundled Fortran implementation).

PSTSWM Implementation Comparisons on X1

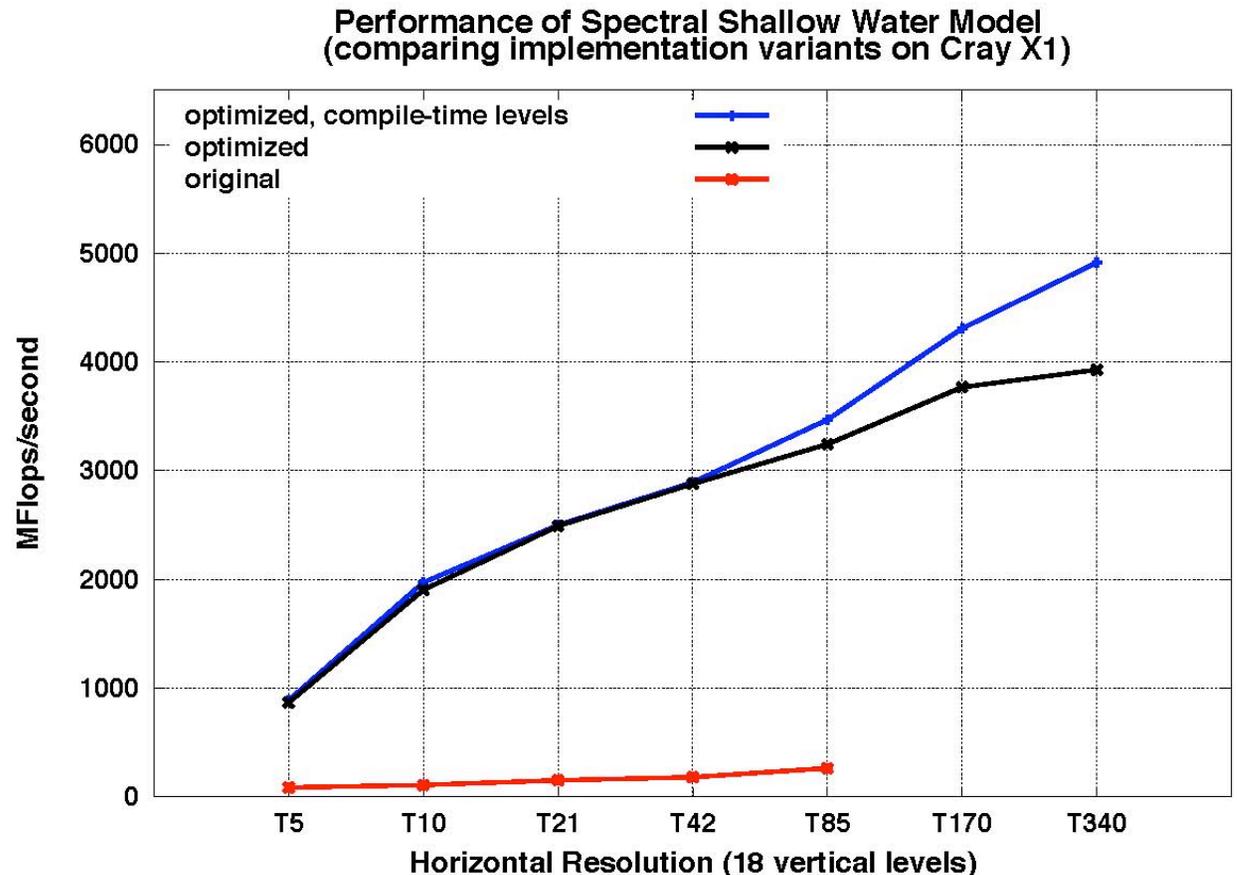
Comparing performance of different code versions for one vertical level. Code modifications are crucial for this code. Fixing vertical dimension at compile time improves performance for large problem sizes. (Default compiler optimization were as good as more aggressive settings.)

Performance of Spectral Shallow Water Model
(comparing implementation variants on Cray X1)



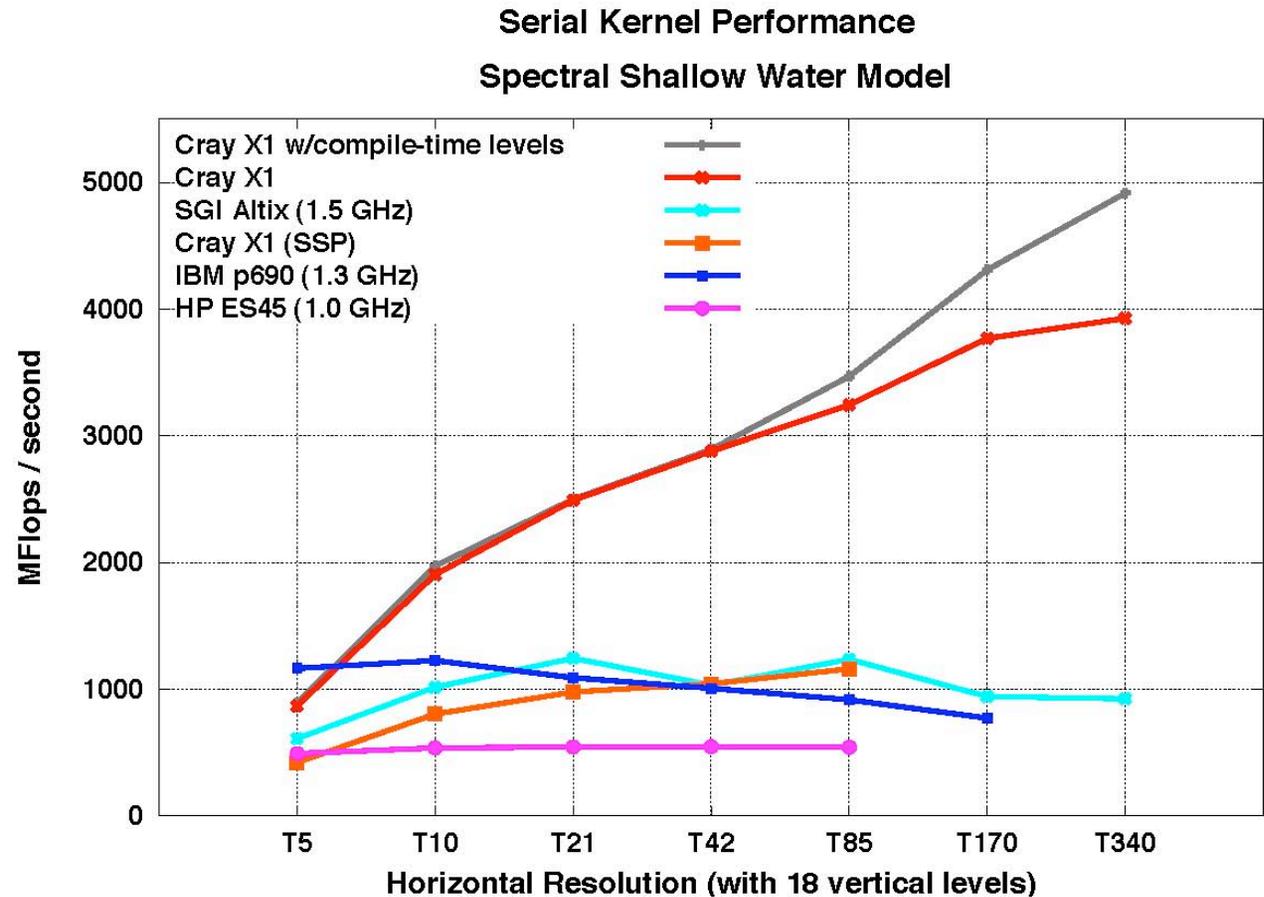
PSTSWM Implementation Comparisons on X1

Comparing performance of different code versions for 18 vertical levels. Improvement due to fixing vertical dimension at compile time not as dramatic as for one vertical level.



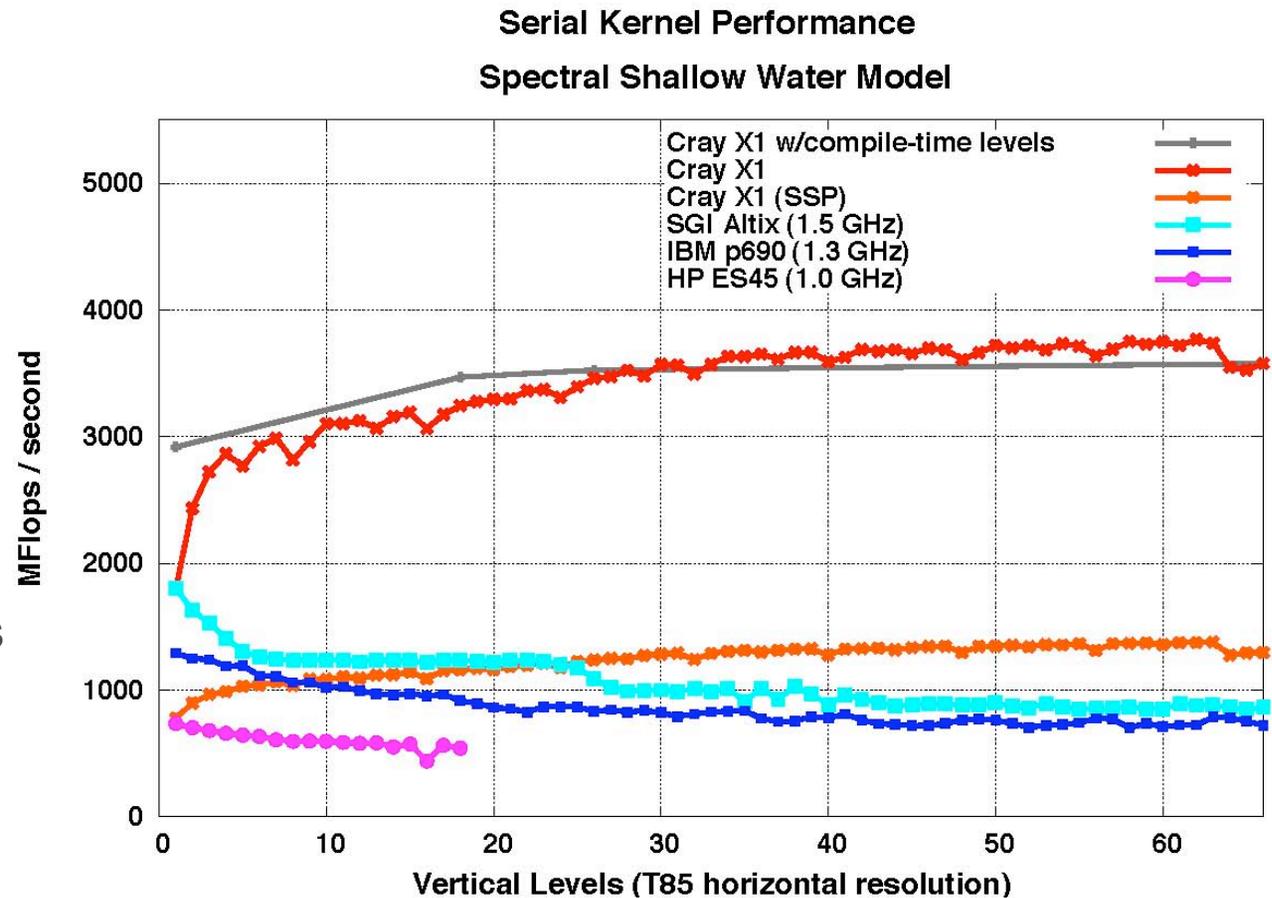
PSTSWM Processor Benchmark

Comparing single processor performance with PSTSWM for 18 vertical levels. X1 MSP version performance scaling well with problem size, and even performance of SSP version exceeds p690 processor performance for the larger problem sizes.



PSTSWM Processor Benchmark

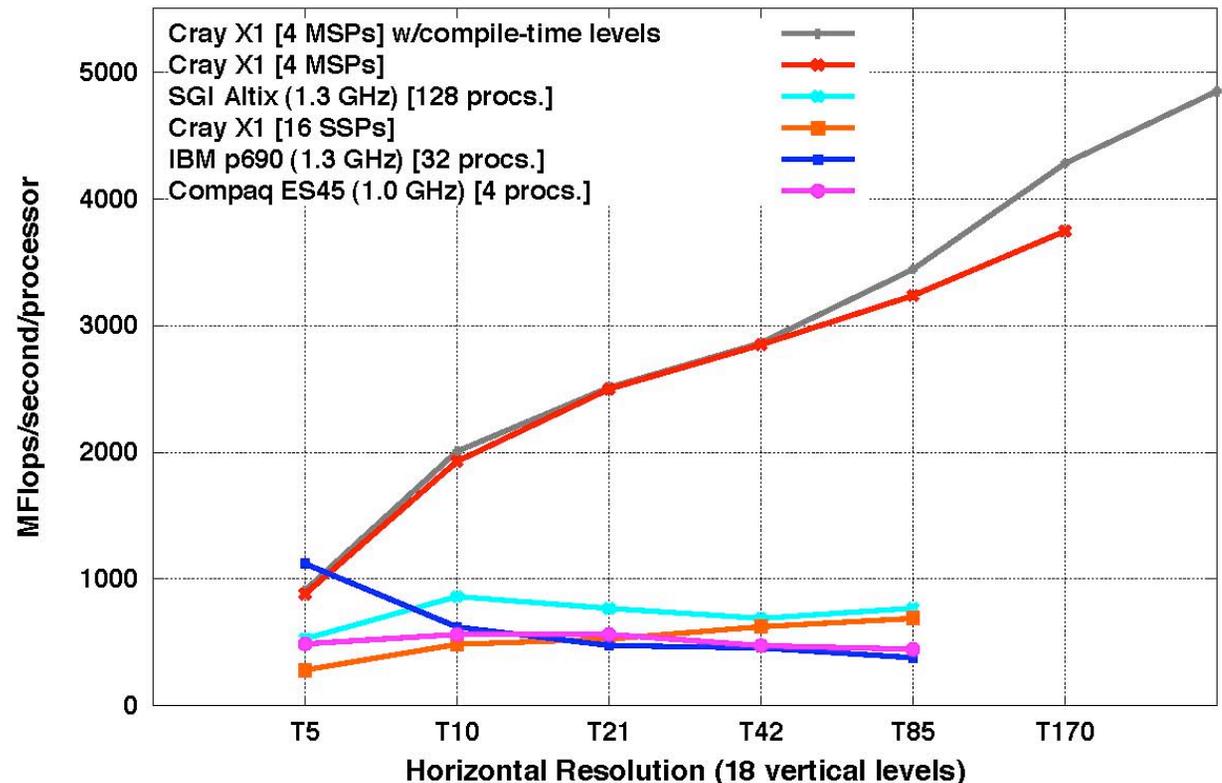
Comparing single processor performance with PSTSWM for T85 horizontal resolution and a range of numbers of vertical levels. Performance of SSP version exceeds that of p690 and Altix processor performance for larger numbers of vertical levels due to the superior processor/memory bandwidth of the X1 .



PSTSWM SMP Node Benchmark

Comparing per processor performance when solving same problem simultaneously on all processors in SMP node. X1 MSP data are only data indicating no performance degradation when compared to single processor experiment. Appears to indicate additional advantage to X1 over other systems for scale-up type experiments (for this code).

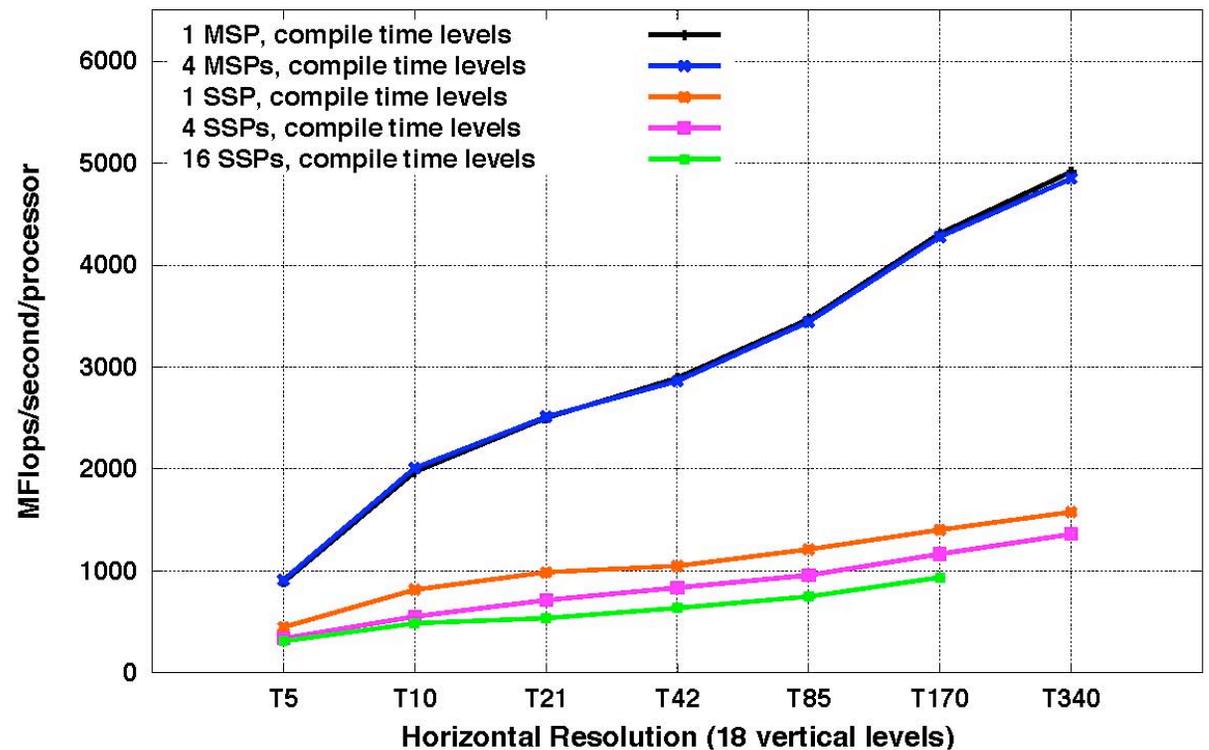
Performance of Spectral Shallow Water Model
(all processors in SMP node experiments)



PSTSWM SMP Node Benchmark on X1

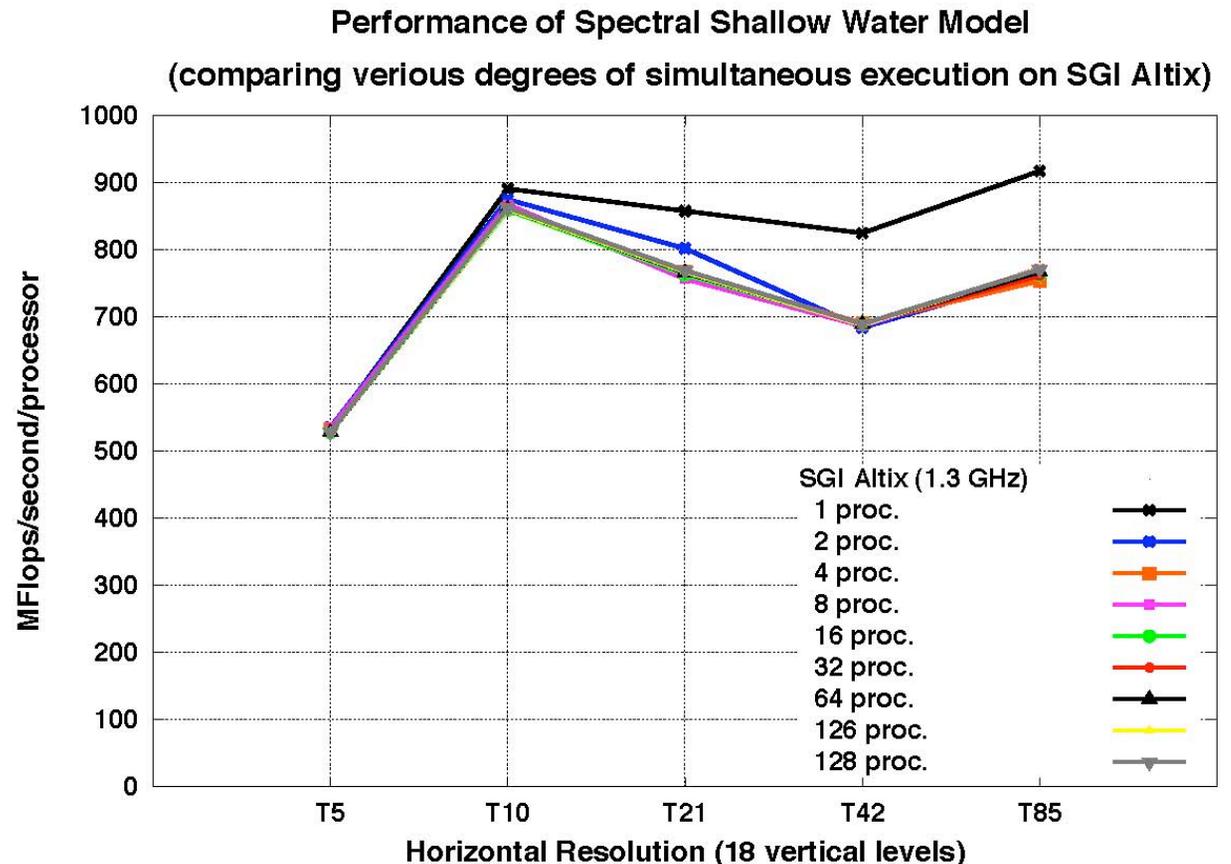
Comparing per processor performance when solving same problem simultaneously on all processors in X1 SMP node. SSP mode sees more contention, and MSP mode achieves better overall throughput for larger problem sizes.

Performance of Spectral Shallow Water Model
(comparing SSP vs. MSP vs. SMP on Cray X1)



PSTSWM SMP Node Benchmark on Altix

Comparing per processor performance when solving same problem simultaneously on all processors in Altix SMP node. dplace used to force use of contiguous processors. Most of performance degradation due to using neighboring processors (sharing access to local memory). Rest of degradation due to using processors in same 4 processor C brick.



Conclusions?

- Both systems work.
- We need more experience with application codes.
- Cray X1
 - SHMEM and Co-Array Fortran performance can be superior to MPI. However, we hope that MPI small message performance can be improved.
 - Both SSP and MSP modes of execution work fine. MSP mode should be preferable for fixed size problem scaling, but which is better is application and problem size specific.
 - The X1 is a vector system, and there is no avoiding using vector-friendly code in order to achieve good performance.
- SGI Altix
 - Stability is poor (this week). This is a known OS issue, not HW.
 - Processor performance and scalability are very good (for a nonvector system).

Questions ? Comments ?

For further information on these and other evaluation studies, visit

<http://www.csm.ornl.gov/evaluation> .