Cray X1
Architecture and Performance Evaluation

Patrick H. Worley

Oak Ridge National Laboratory

SIAM Annual Meeting
July 11-15, 2005
Hilton New Orleans
New Orleans, LA

Acknowledgements

- Some slides provided by John Levesque of Cray, Inc.
- Research sponsored by 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.

Who Am I?

Senior Research Staff at Oak Ridge National Laboratory (ORNL), working on

- Performance evaluation of HPC architectures
- Performance analysis and modeling research
- Parallel algorithm design and optimization for global atmospheric and ocean models

and focusing (most recently) on the

- Cray X1E, XT3, XD1, X1
- SGI Altix
- IBM p690 cluster with HPS interconnect

at ORNL.

Outline of Talk

- A brief overview of the architecture.
- A sampling of microkernel, kernel, and application performance results that illustrate the capabilities and peculiarities of the X1 architecture.
- Setting the stage for the later talks, taking care of common preliminaries.

Caveats

- The performance results described in this talk were collected at various times over the past 2 years. The system software has improved over time, and some of the quantitative results may no longer be accurate. However, the qualitative performance behavior has not changed significantly.
- Many sites (including ORNL) are currently transitioning from the X1 to the X1E architectures, which involves:
  - Faster processors
  - Lower memory latency
  - Increased contention for memory bandwidth
  - Increased contention for interconnet bandwidth

  We are just now determining the impact of these changes on application performance.

Phoenix (prior to June 2005)

Cray X1 with 128 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 512 processors (MSPs), 2048 GB of memory , and ~6500 GF/s peak performance.

Cray X1 Multi-Streaming Processor

12.8 Gflops (64 bit)

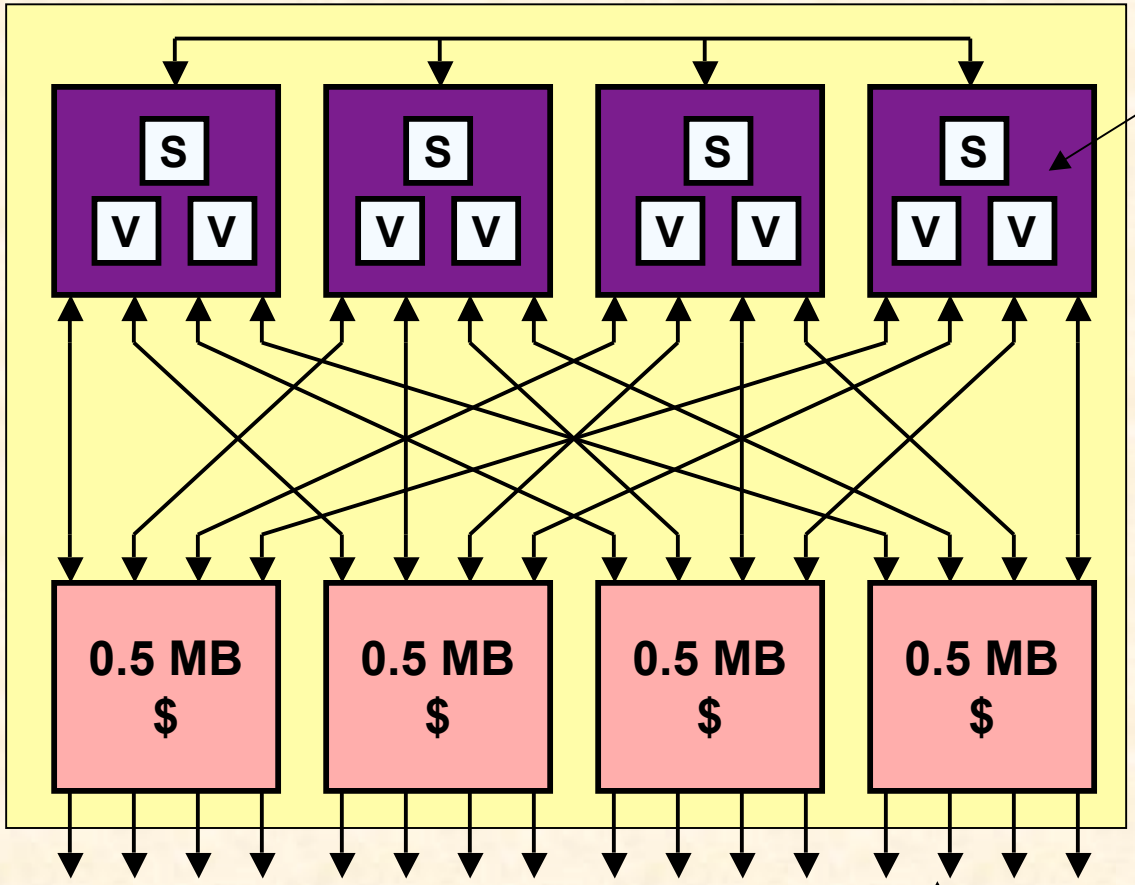
25.6 Gflops (32 bit)

51 GB/s ↑

25-41 GB/s ↓

2 MB Ecache

At frequency of 400/800 MHz

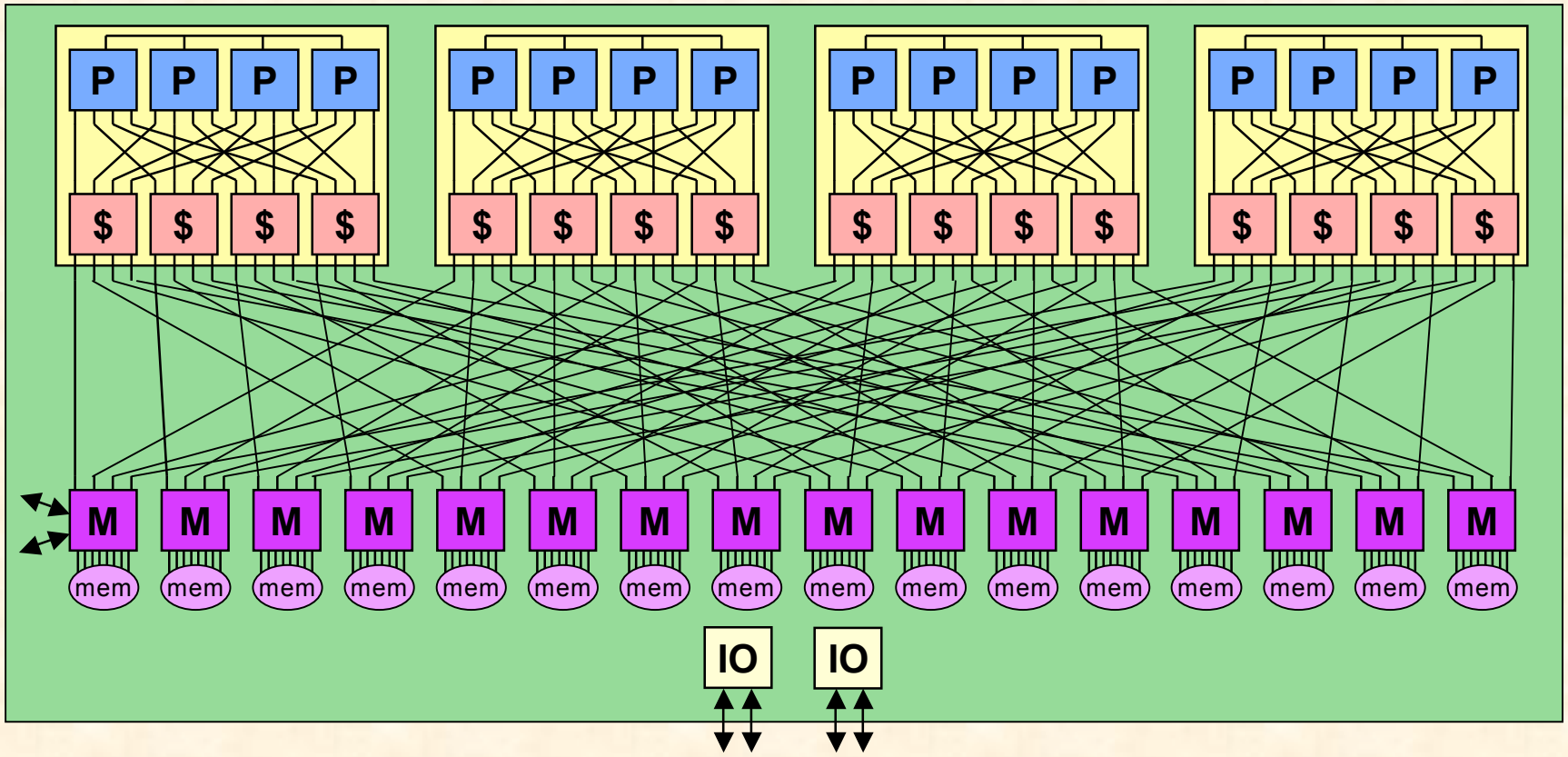


To local memory and network:

25.6 GB/s ↑

12.8 - 20.5 GB/s ↓

Cray X1 Node



Inter node network:

- 2 ports per M chip
- 1.6 GB/s full duplex per link
- 102.4 Gbyte/s to the network

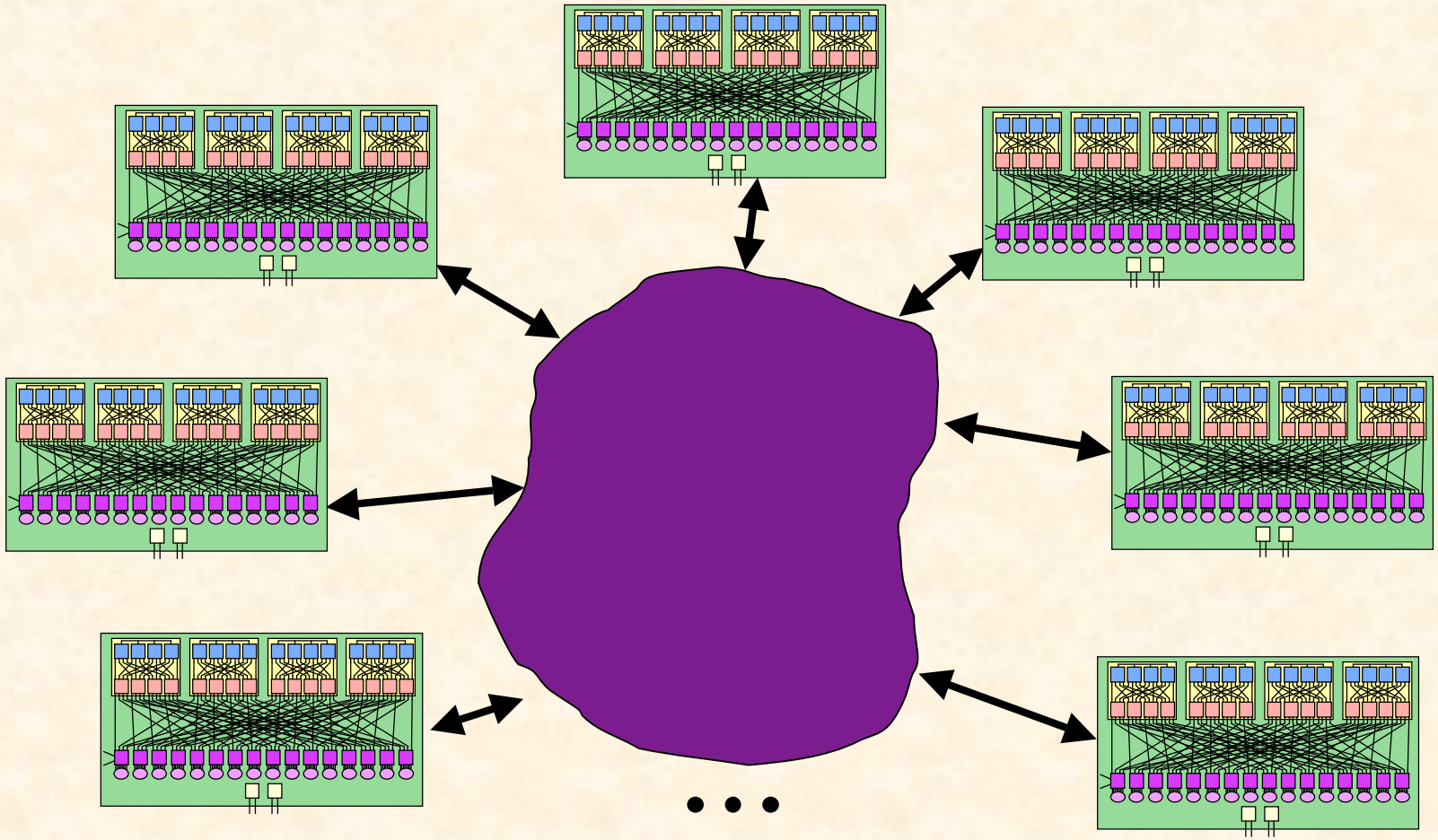
I/O connections:

- 4 ports per node
- 1.2 GB/s full duplex per link

Local memory:

- 200 GB/s peak bw
- 16-32 GB per node

NUMA Scalable up to 1024 Nodes



- 16 parallel networks for bandwidth
- Global shared memory across machine

System Characteristics

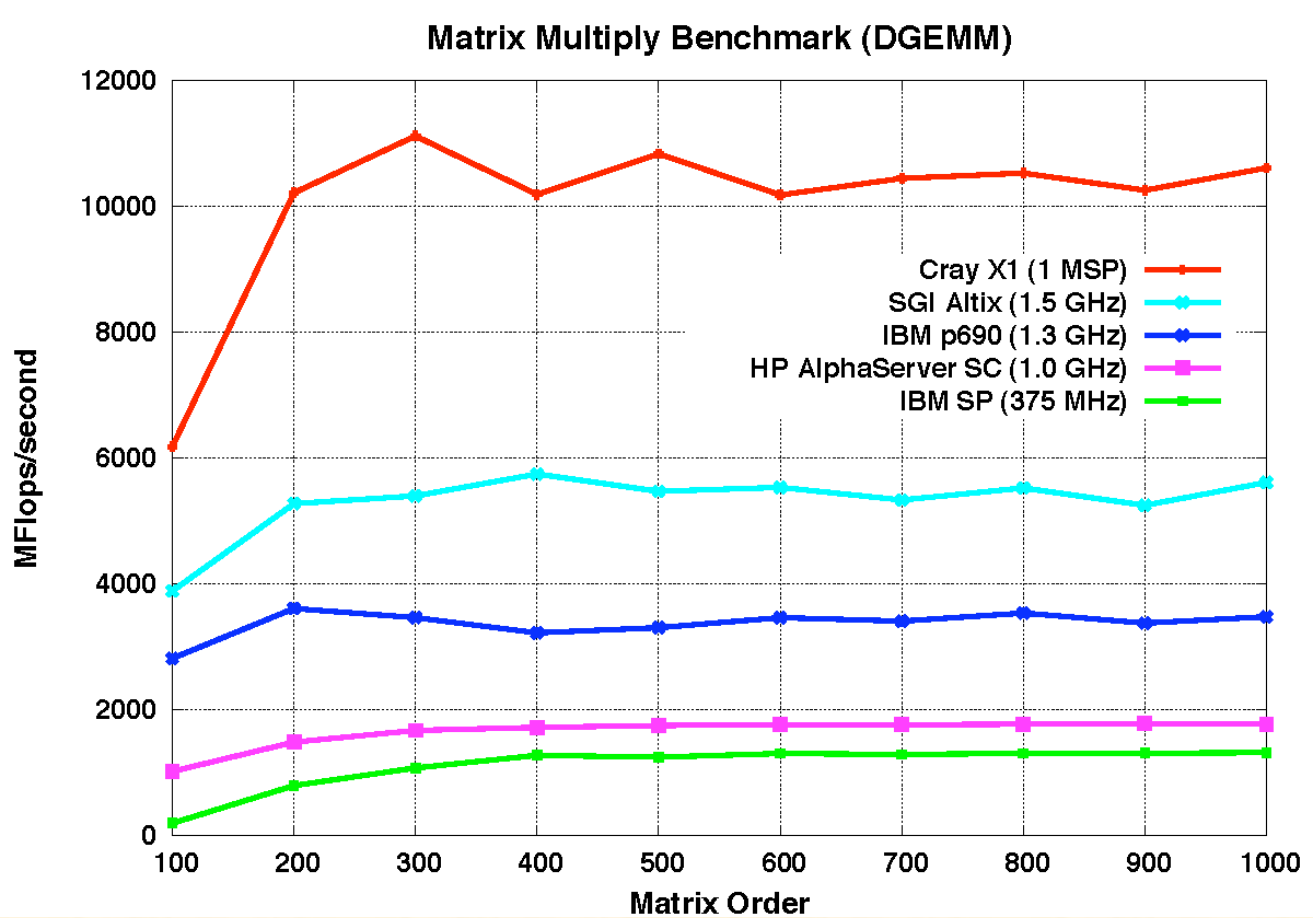
- Serial performance*
  - Performance range
  - Runtime vs. compile time loop bounds
  - Impact of memory subsystem on performance
- Communication performance
  - MPI protocols
  - MPI vs. SHMEM vs. Co-Array Fortran
  - Distance
  - Contention

* Some data collected by Tom Dunigan, also at ORNL.

DGEMM Benchmark

Comparing performance of vendor-supplied routines for matrix multiply. Cray X1 experiments used routines from the Cray scientific library libsci.

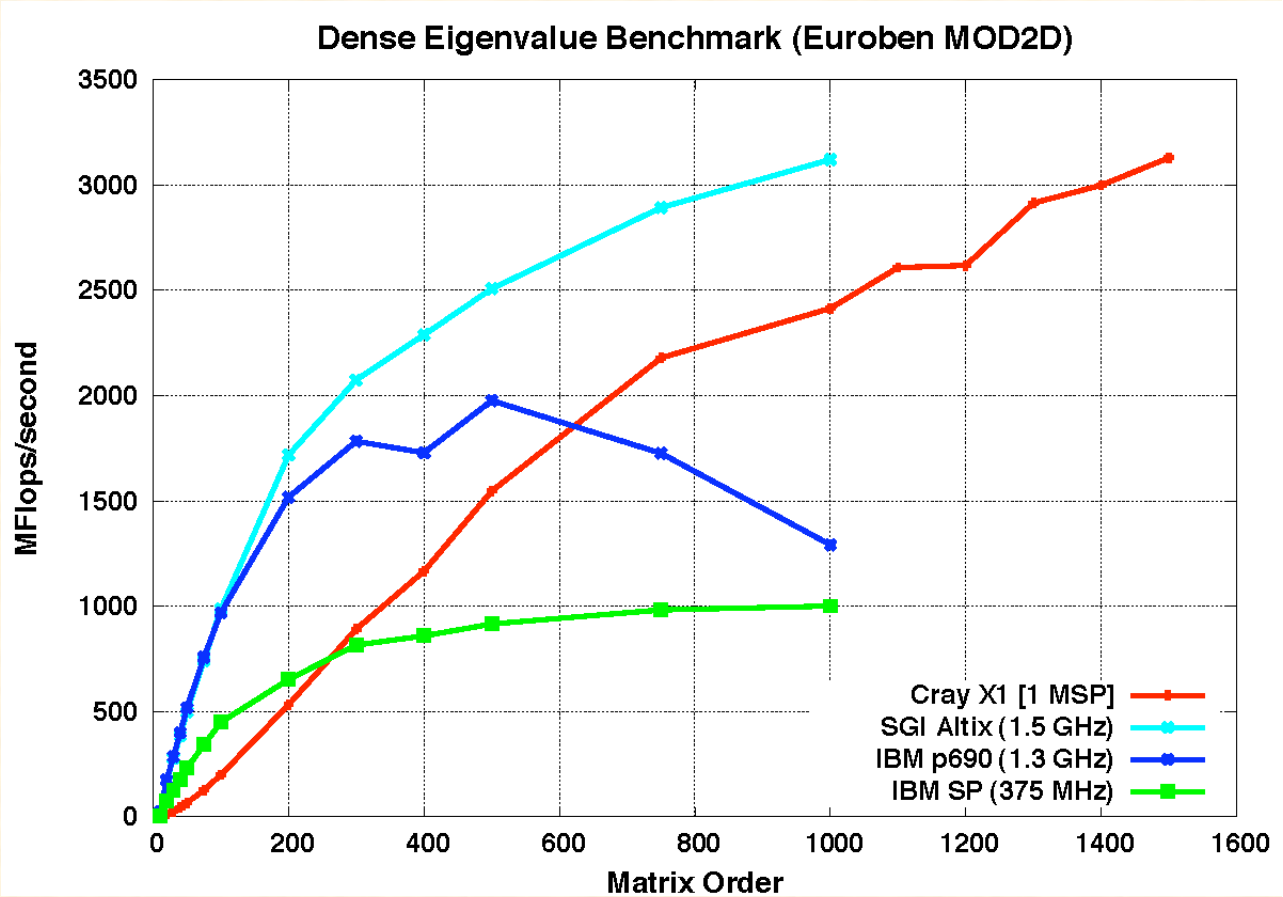
Good performance achieved, reaching 80% of peak relatively quickly.



Euroben MOD2D Benchmark

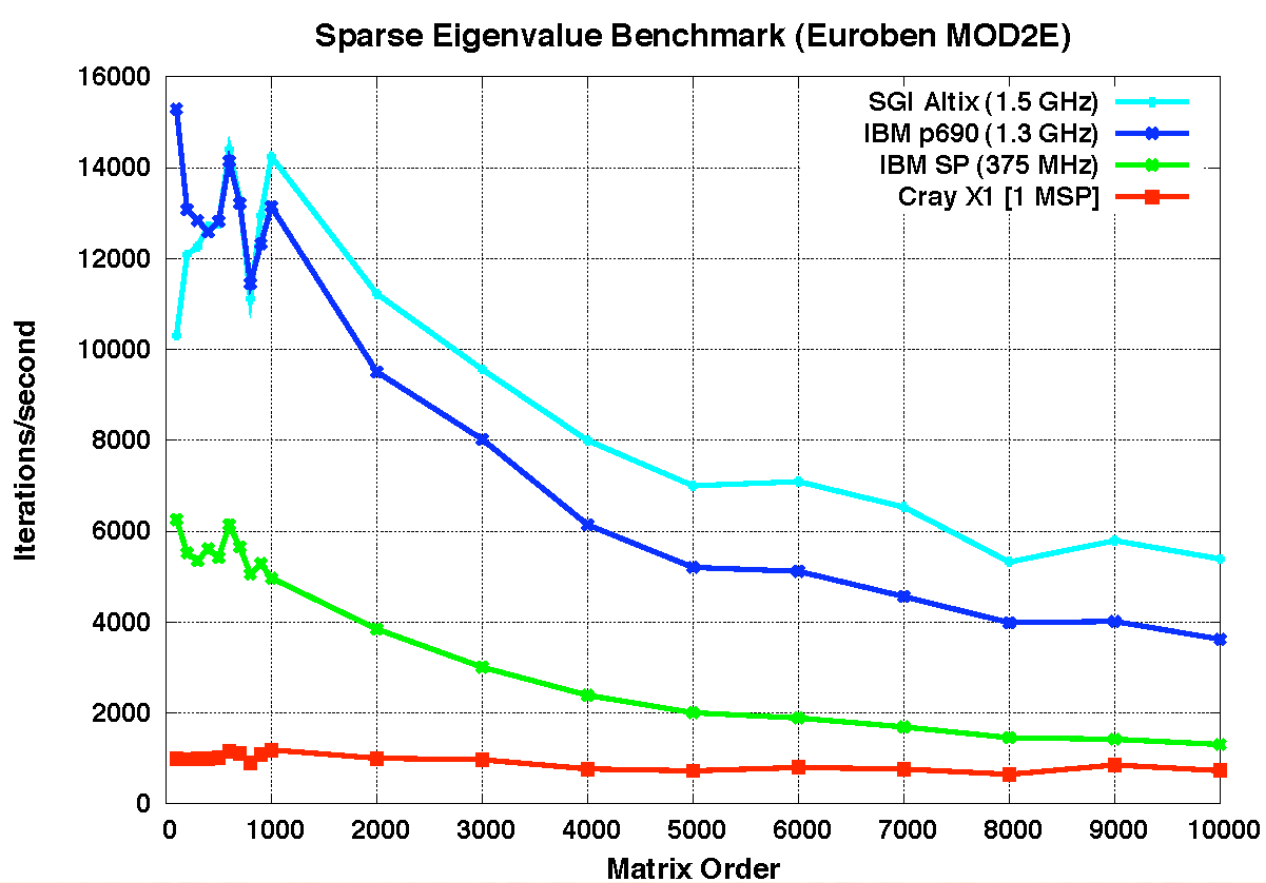
Comparing performance of vendor-supplied routines for dense eigenvalue analysis. Cray X1 experiments used routines from the Cray scientific library libsci.

Performance still growing with problem size for Cray and SGI. Performance of IBM systems has peaked.



Euroben MOD2E Benchmark

Comparing performance of Fortran code for sparse eigenvalue analysis. Aggressive compiler options were used on the X1, but code was not restructured and compiler directives were not inserted. Performance is improving for larger problem sizes, so some streaming or vectorization is being exploited. Performance is poor compared to other systems.



PSTSWM

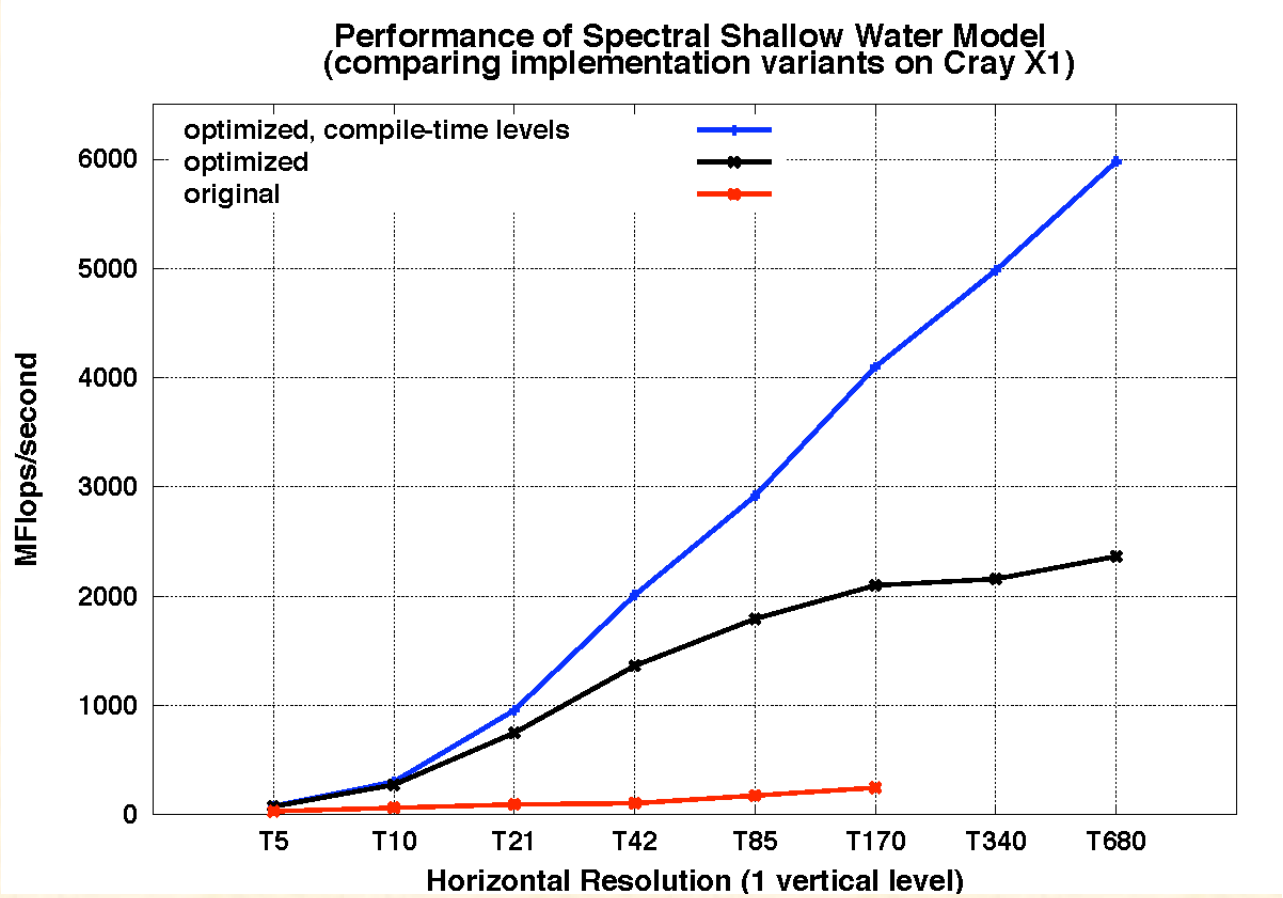
The Parallel Spectral Transform Shallow Water Model represents an important computational kernel in spectral global atmospheric models. As 99% of the floating-point operations are multiply or add, it runs well on systems optimized for these operations. PSTSWM exhibits little reuse of operands as it sweeps 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. PSTWM 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 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

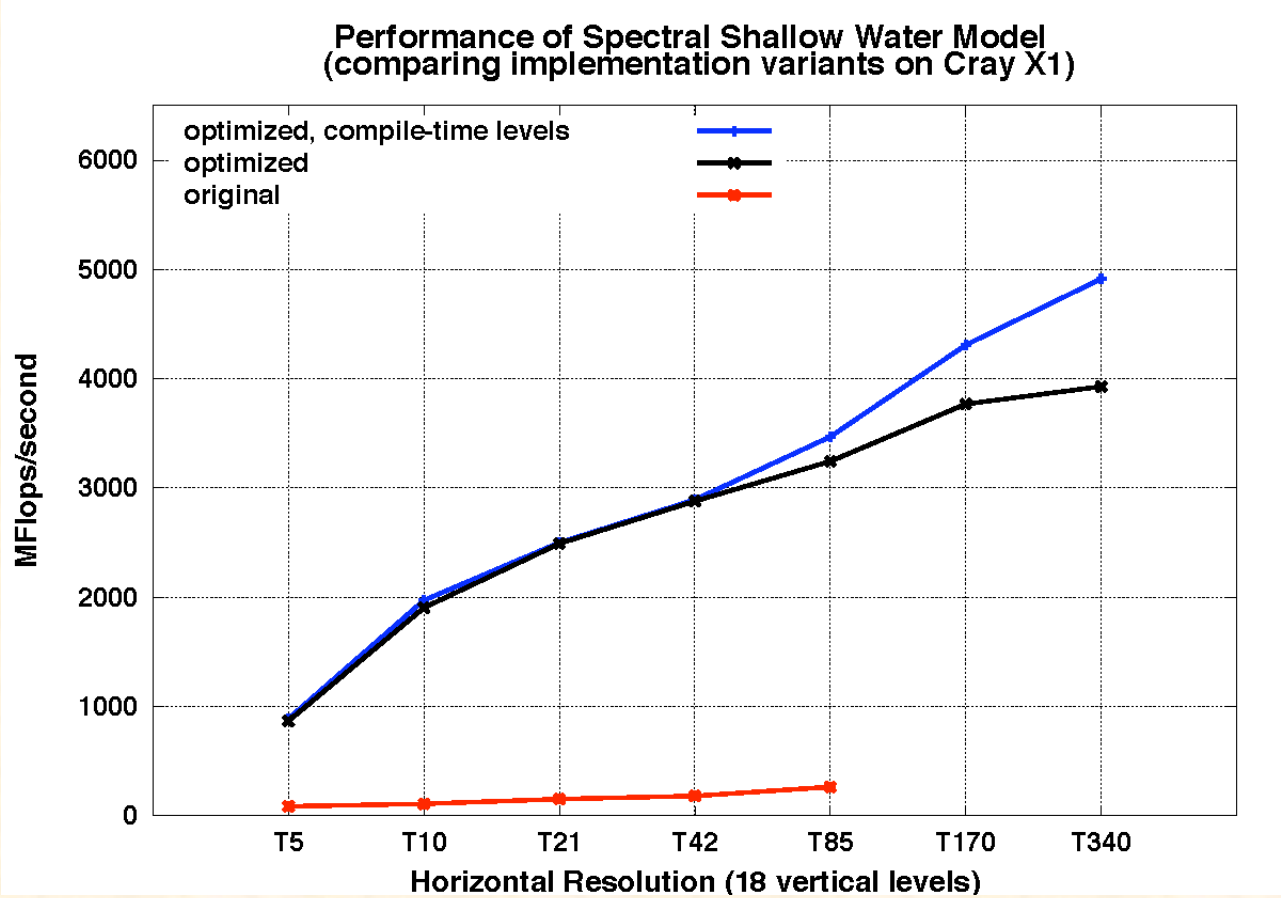
PSTSWM Implementation Comparisons

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.



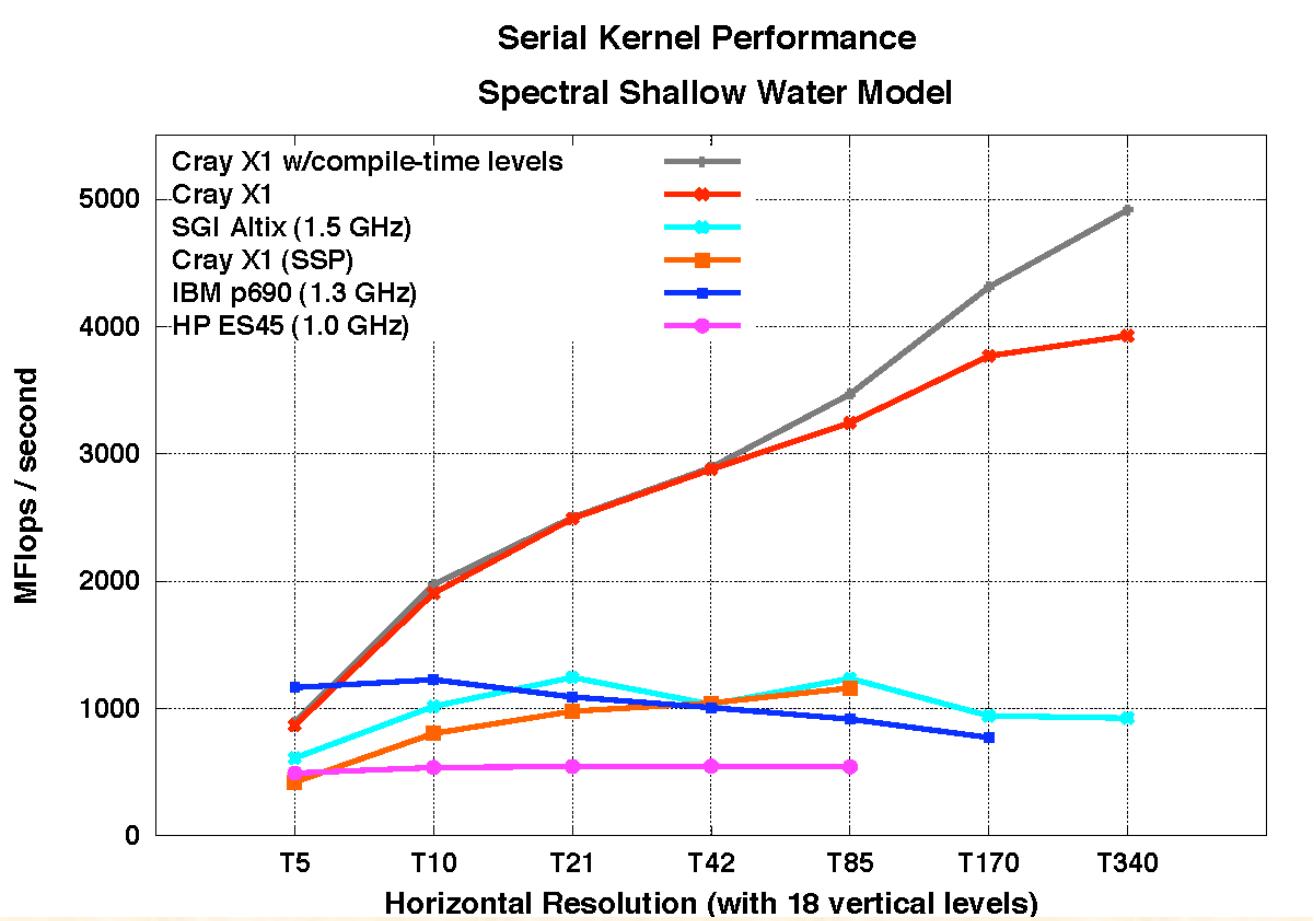
PSTSWM Implementation Comparisons

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 Serial 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.

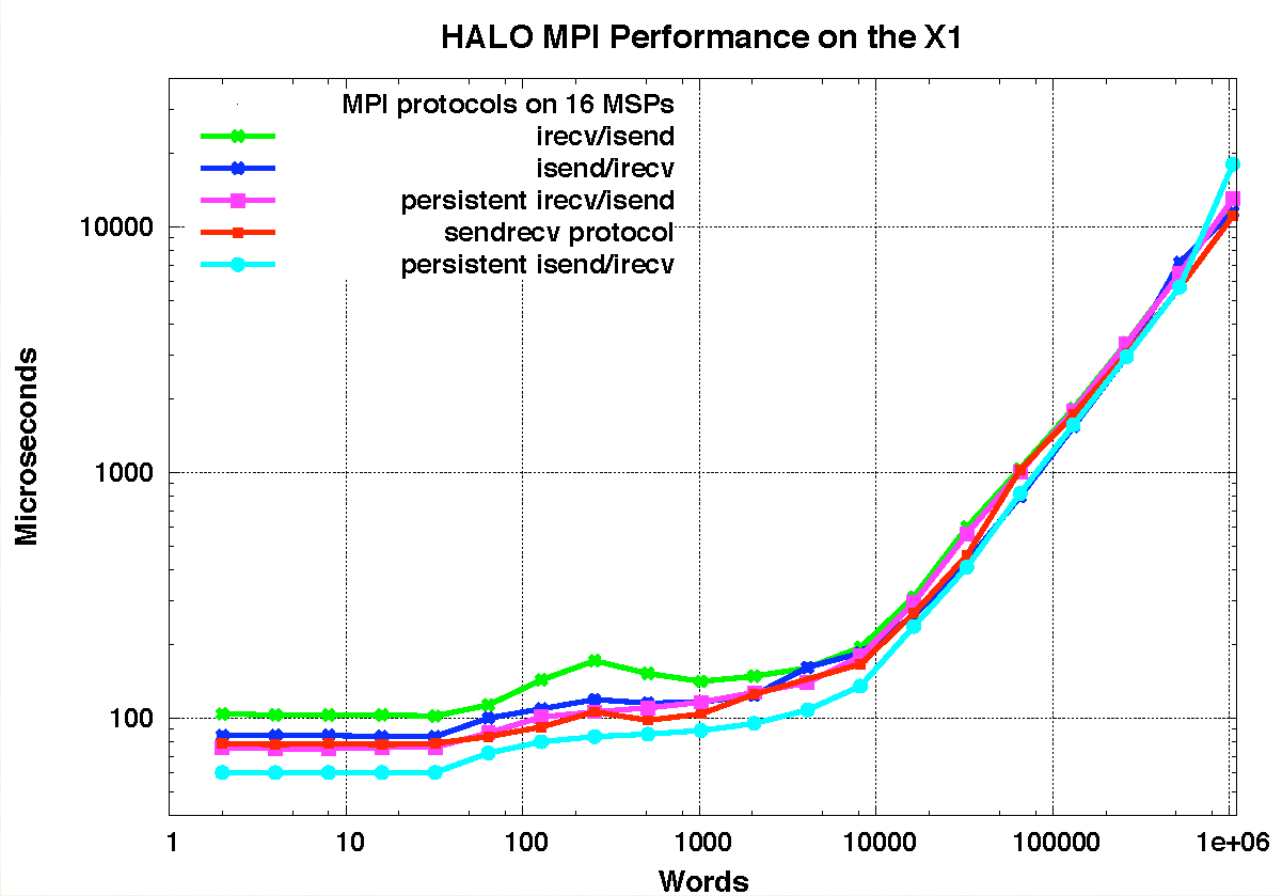


Moral?

- Not all codes are a good match for the Cray X1 architecture. In particular, scalar performance is poor; Code had better vectorize (and/or stream).
- Don't expect good performance without making some attempt at optimization.
- The more information you give the compiler, the better job it can do.
- Large problems typically run better than small problems (relative to nonvector platforms), due to increased vector lengths and good memory subsystem performance.
- Memory access pattern can limit performance on the X1.

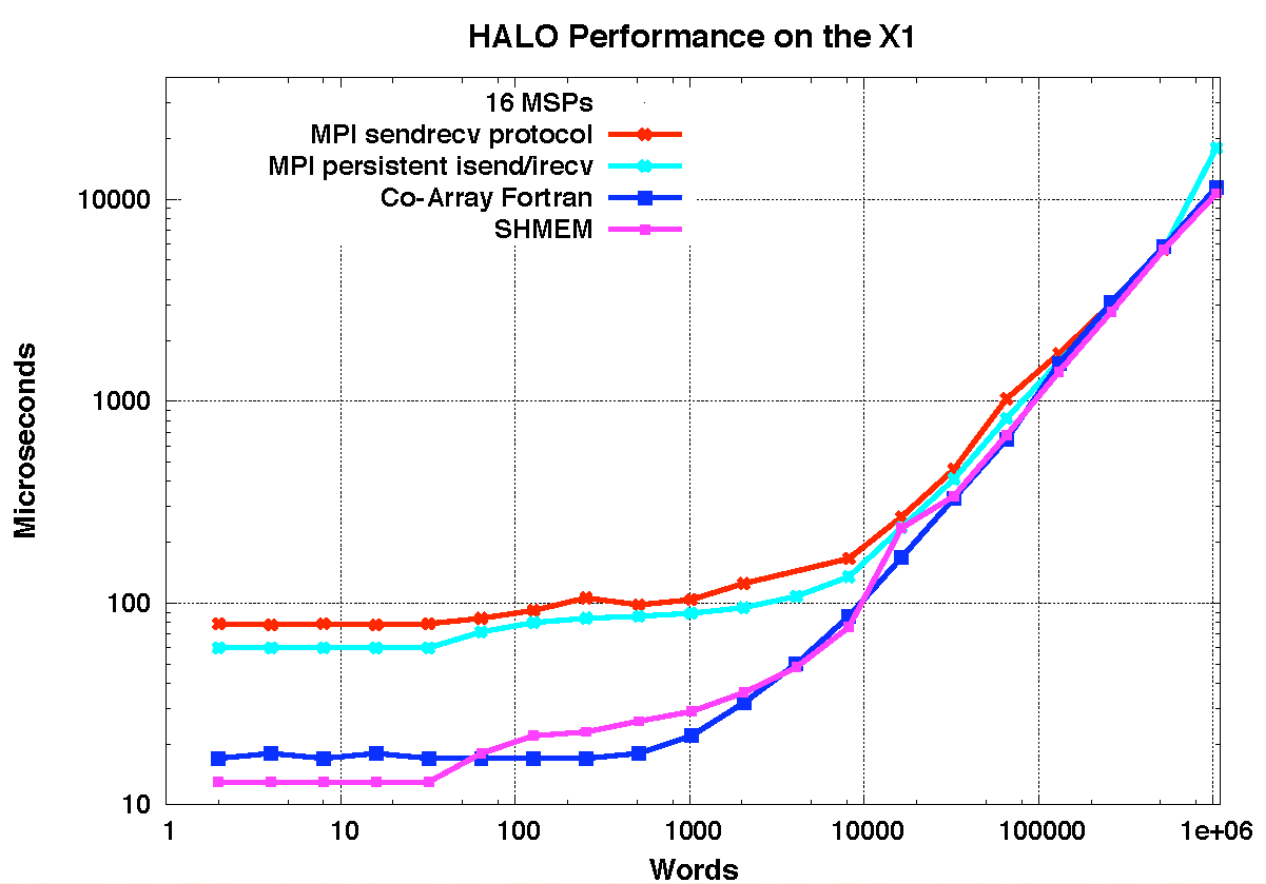
HALO MPI Protocol Comparison

Comparing performance of different MPI implementations of Allan Wallcraft's HALO benchmark on 16 MSPs. Persistent isend/irecv is always best. For codes that can not use persistent commands, MPI_SENDRECV is also a reasonable choice.



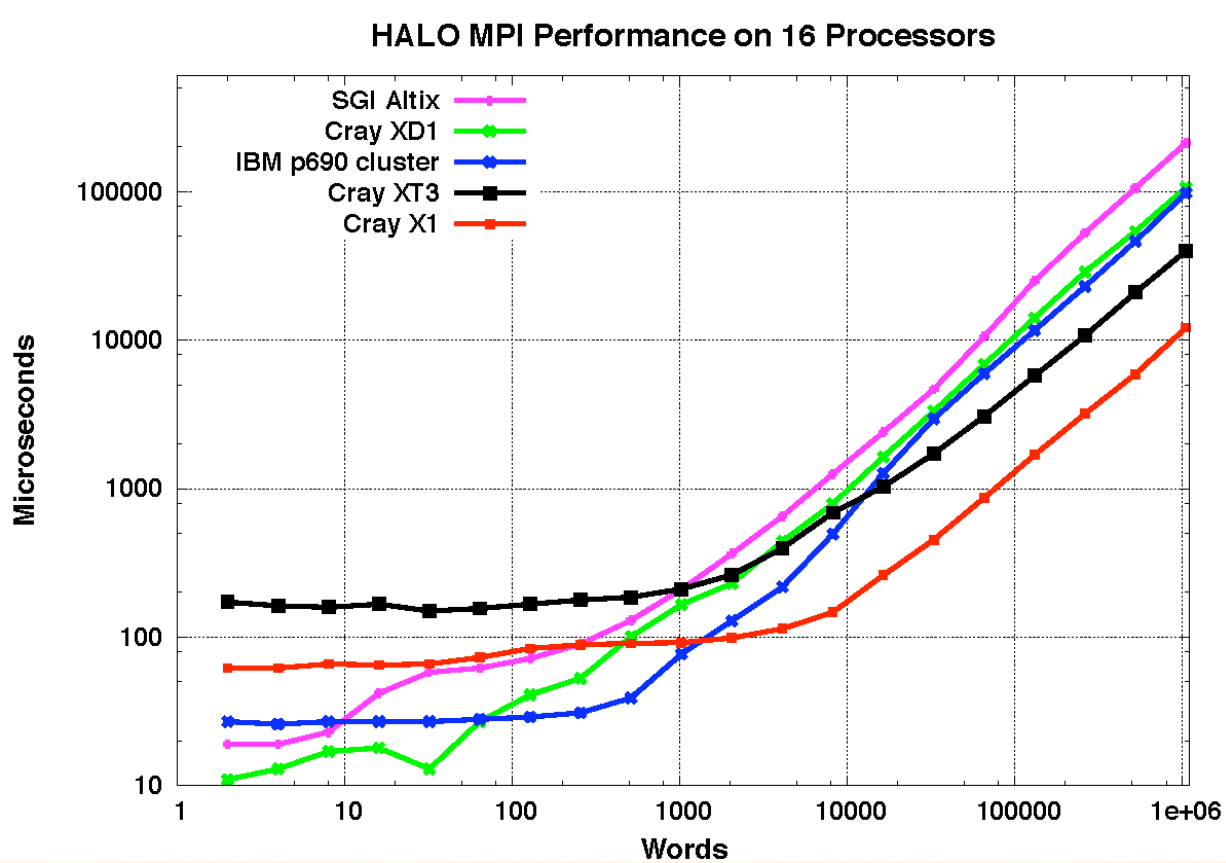
HALO Exchange Paradigm Comparison

Comparing performance of MPI, SHMEM, and Co-Array Fortran implementation HALO benchmark on 16 MSPs. SHMEM and Co-Array Fortran are substantial performance enhancers for this benchmark for small halos.



HALO Benchmark

Comparing HALO performance using MPI on 16 MSPs of the Cray X1 and 16 processors of the IBM p690 (within a 32 processor SMP), the SGI Altix, the Cray XD1, and the Cray XT3. Achievable bandwidth is much higher on the X1. For small halos, the XD1 MPI HALO performance is similar to the X1 SHMEM HALO performance.



SWAP Experiments

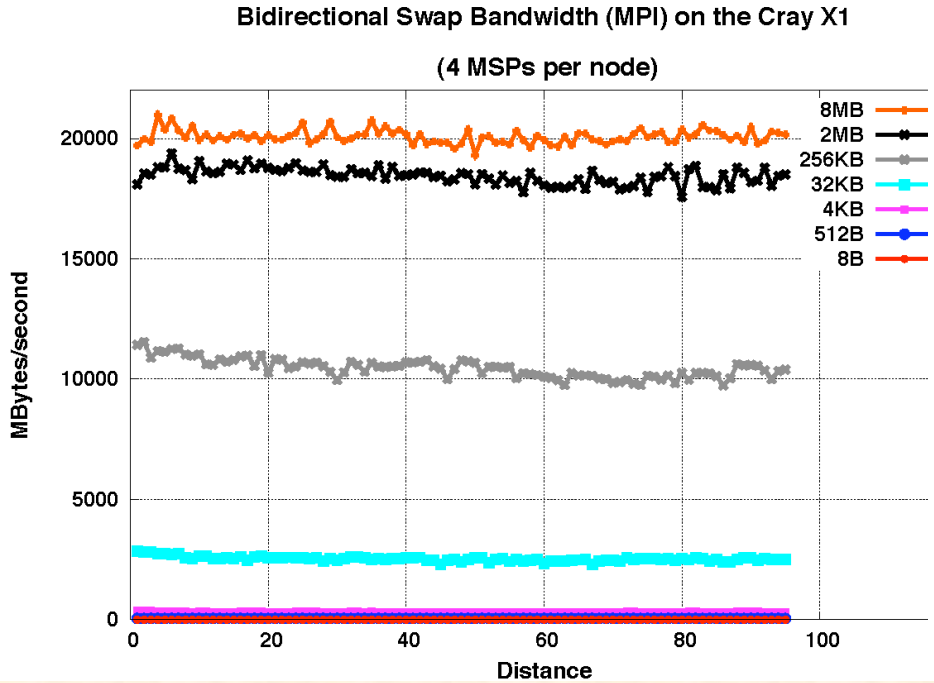
i-j

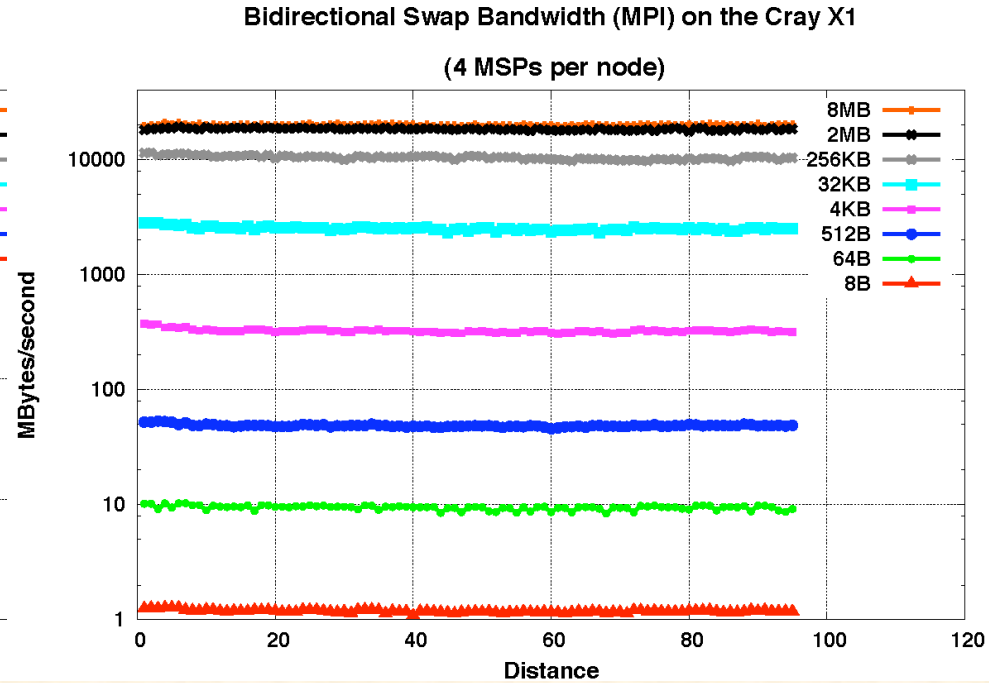
processor i swaps data with processor j. Depending on i and j, this can be within an SMP node or between SMP nodes.

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).

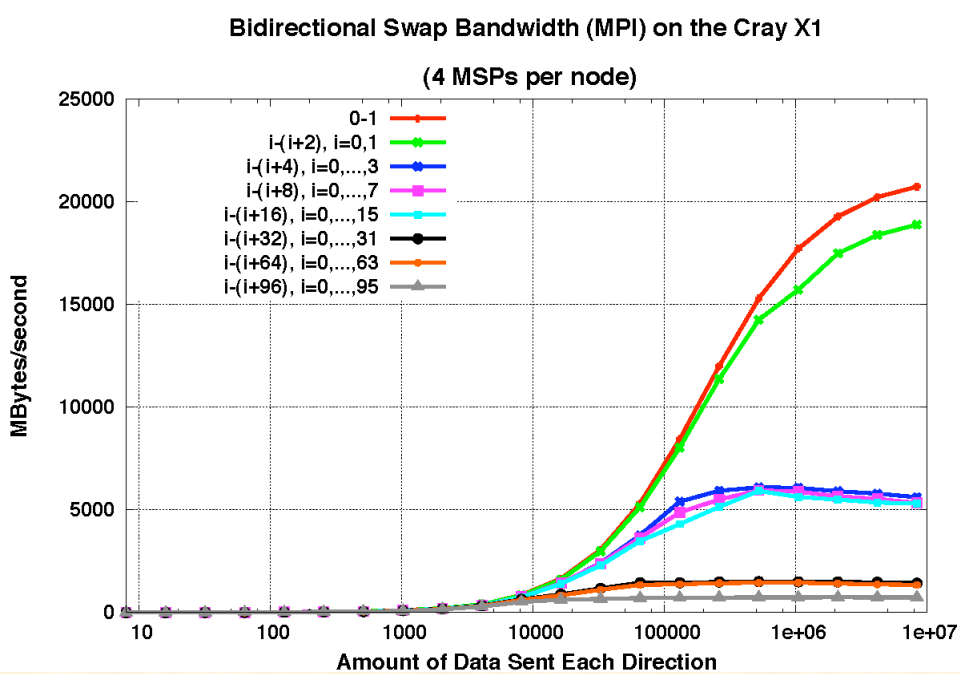
Topology and Distance

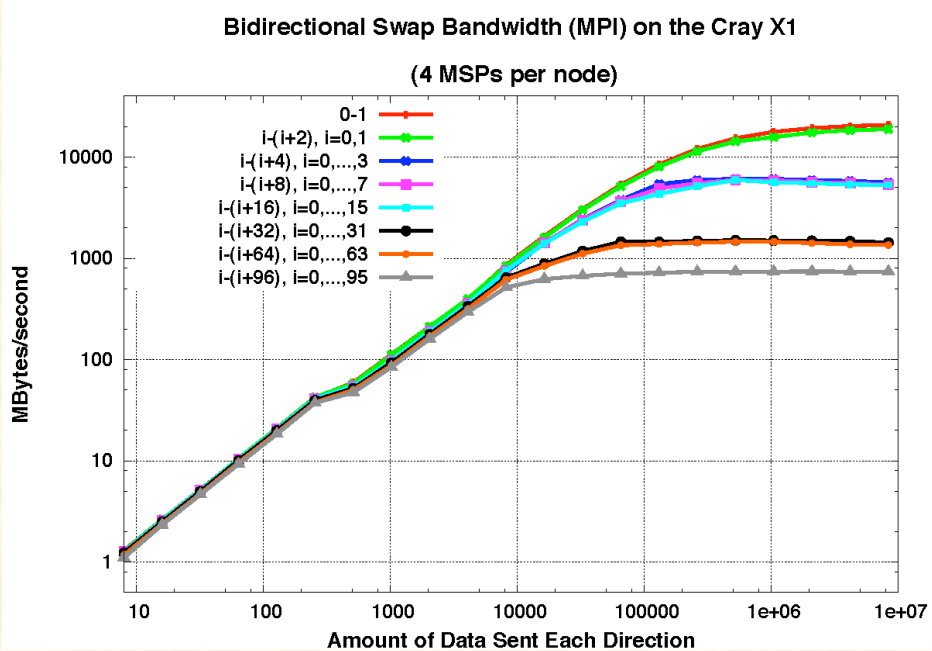




- Systems of size up to 512 MSPs have a 4-D hypercube interconnect.
- “Contiguous” MSPs used in experiments, but system was not dedicated.
- While bandwidth curves are somewhat noisy, there is little practical performance difference due to distance observable in these plots.

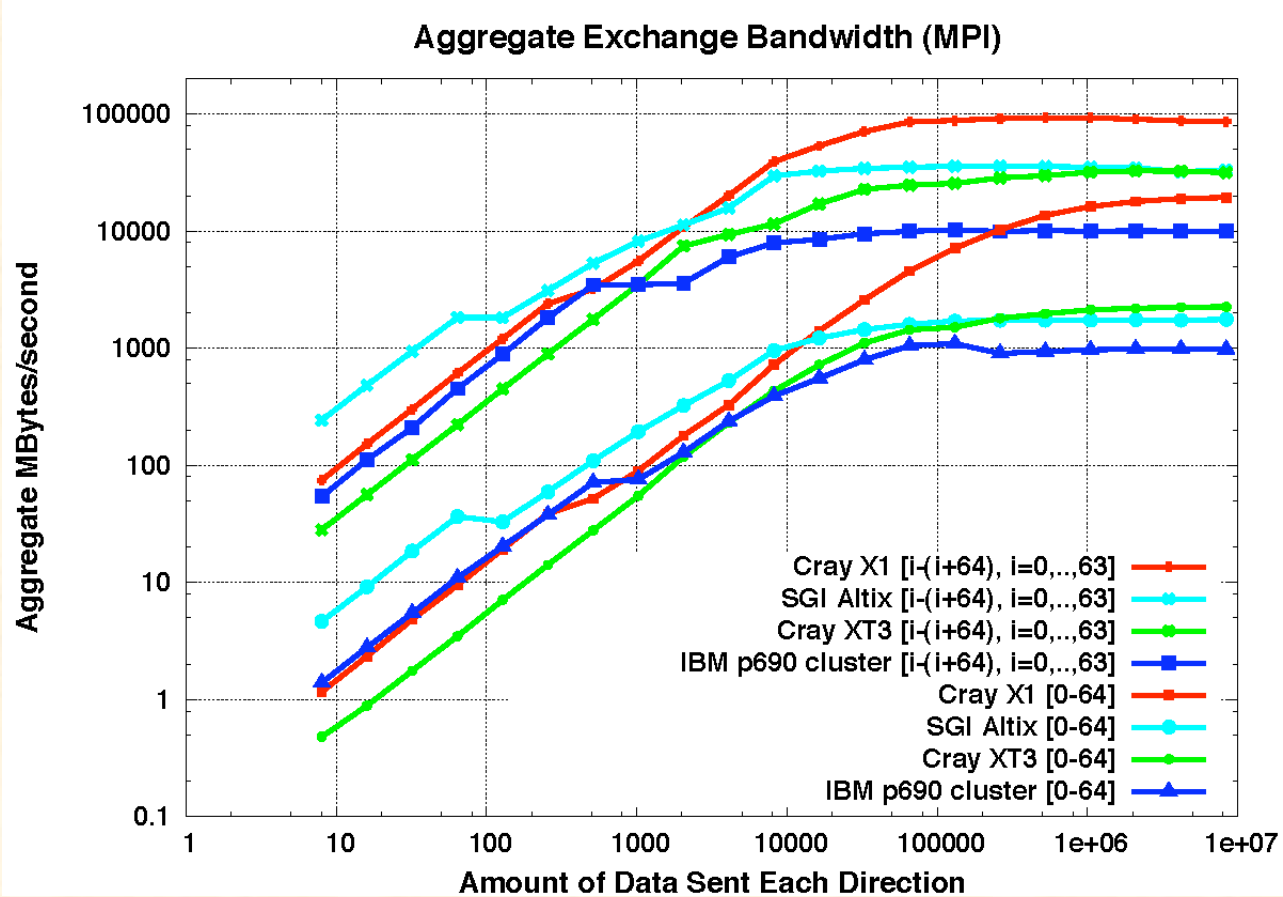
Contention: X1





- Performance degradation due to contention does occur, but the absolute performance is still very good on the X1 when compared to other platforms.

Contention: Platform Comparisons



- Comparing X1, XT3, Altix, and p690 cluster performance, for single pair and 64 simultaneous pairs experiments. Altix demonstrates best MPI latency among these platforms. XT3 large message performance in similar to that for the Altix.

Moral?

- The MPI protocol used can impact performance.
- MPI latency is mediocre. However, latency for Co-Array Fortran or SHMEM is excellent.
- MPI bandwidth is excellent for individual processor pairs. Performance degradation due to contention is well-behaved and is significant for the examined communication patterns only for messages > 8KBytes on the current system.
- MPI latency and bandwidth are relatively insensitive to distance between communicating processors on the current system.

Application Performance Examples

- POP Ocean code
  - Performance impact of vectorization and MPI tuning
  - Performance impact of Co-array Fortran
  - Performance importance of low latency
- GYRO Fusion code
  - Performance importance of high bandwidth

Parallel Ocean Program (POP)

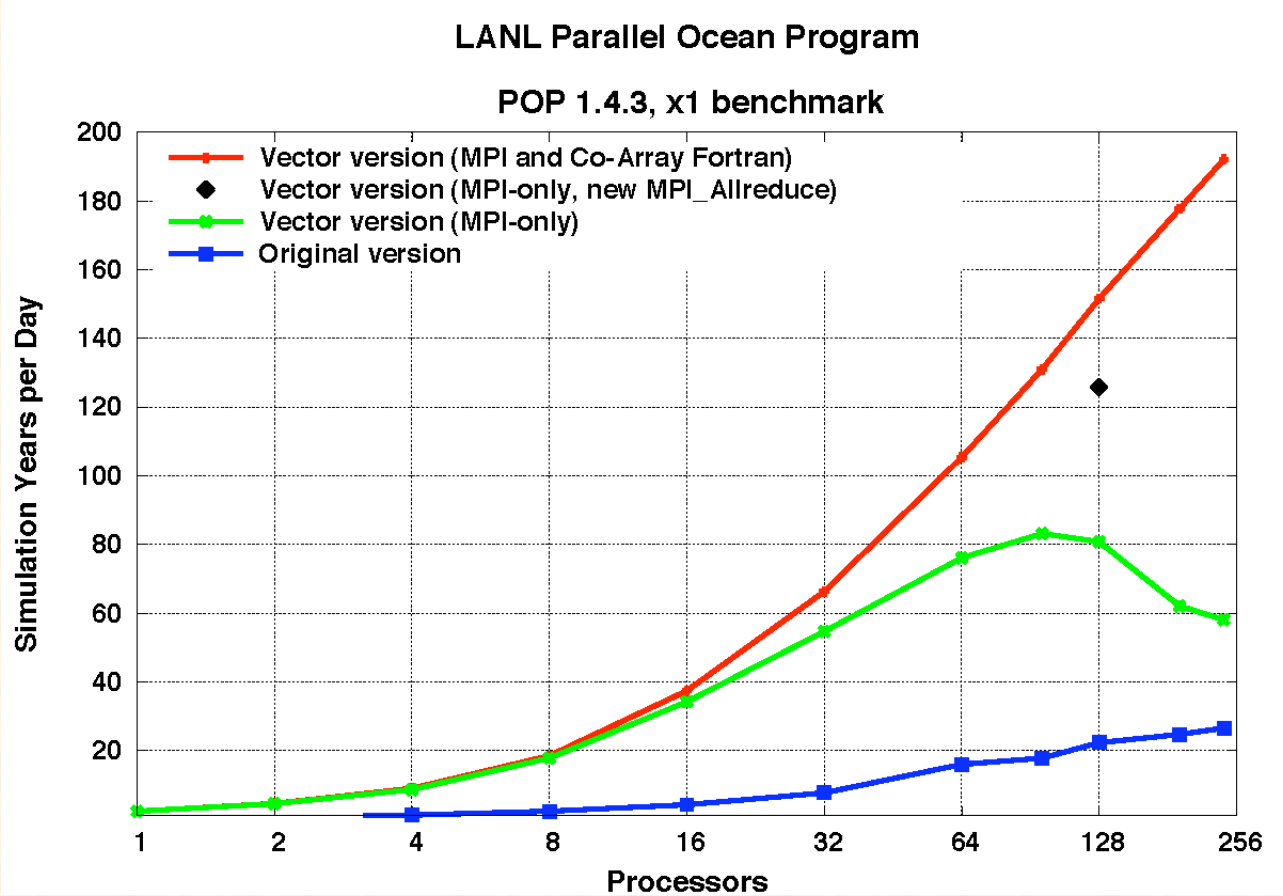
- Developed at Los Alamos National Laboratory. Used for high resolution studies and as the ocean component in the Community Climate System Model (CCSM)
- Ported to the Earth Simulator by Dr. Yoshikatsu Yoshida of the Central Research Institute of Electric Power Industry (CRIEPI).
- Initial port to the 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.
- The version of POP used in these experiments is a pure MPI code (i.e., does not use SMP parallelism). In the Cray X1 experiments POP is run with one process per MSP.

POP Experiment Particulars

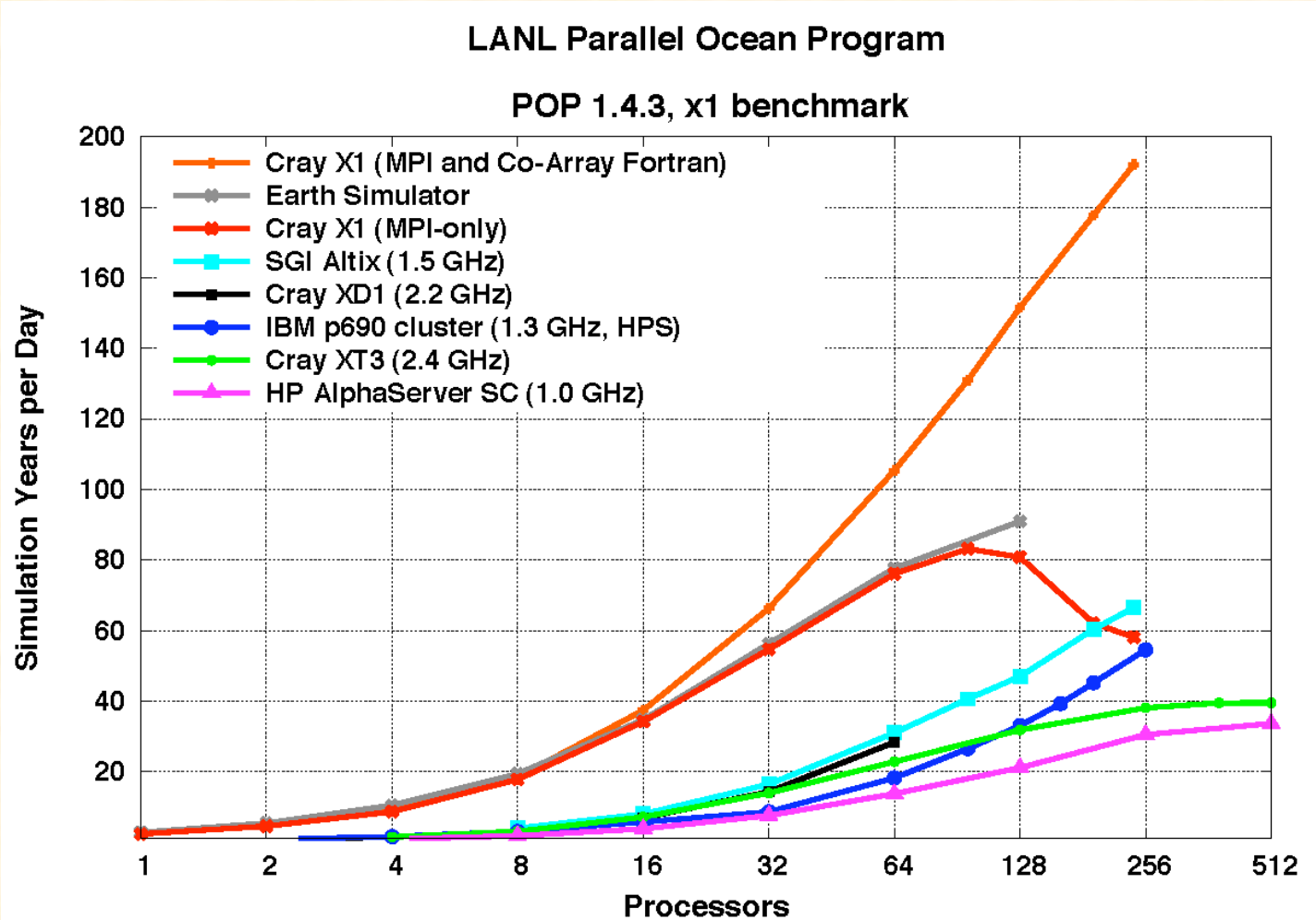
- Two primary computational phases
  - Baroclinic: 3D with limited nearest-neighbor communication; scales well.
  - Barotropic: dominated by solution of 2D implicit system using conjugate gradient solves; scales poorly.
- One fixed size benchmark problem
  - One degree horizontal grid (“by one” or “x1”)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

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.

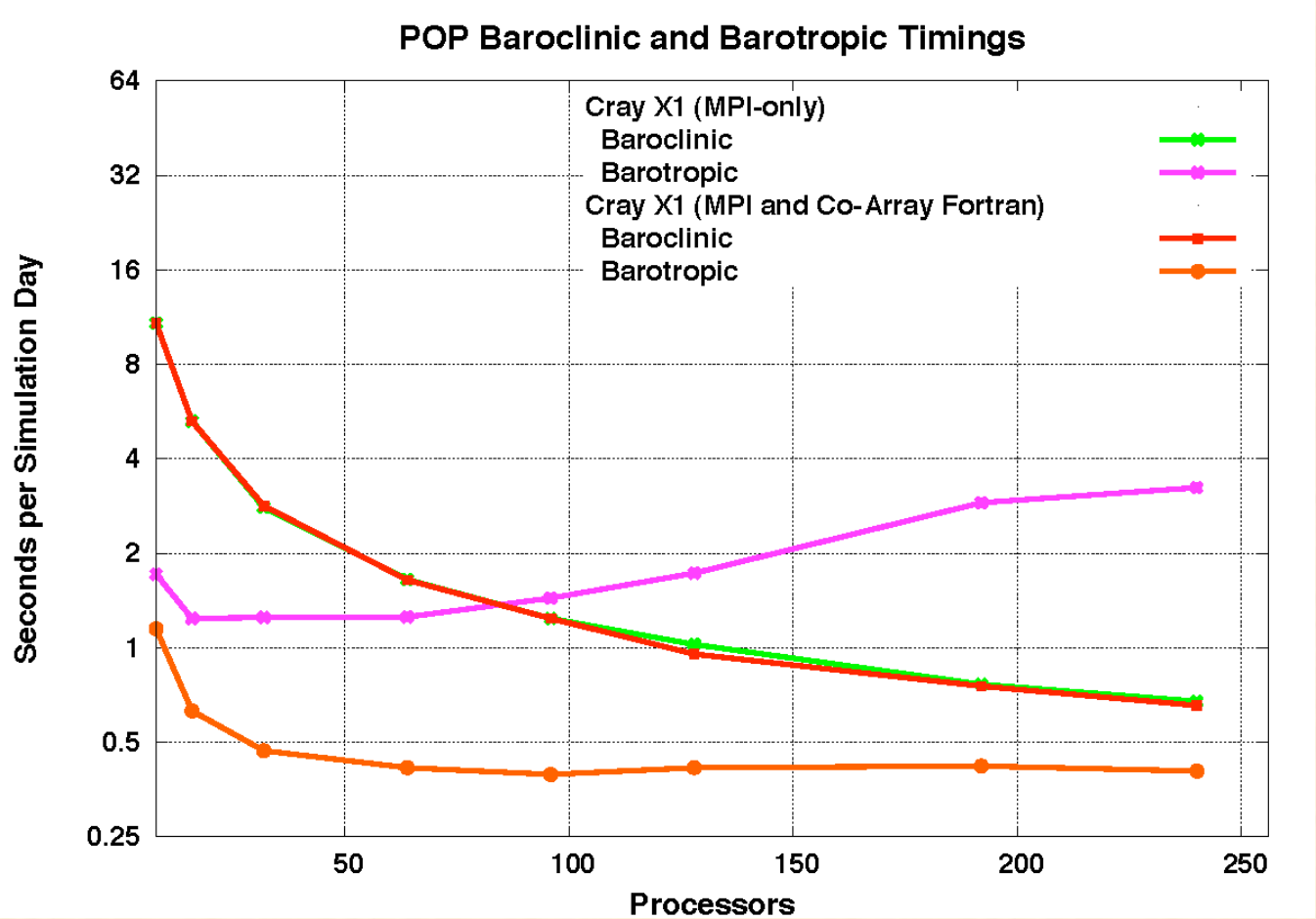


POP Platform Comparisons



- Earth Simulator results courtesy of Dr. Y. Yoshida. X1 performance is excellent when using Co-Array Fortan.

POP Performance Diagnoses



- Examining time spent in baroclinic and barotopic phases for the X1, with and without Co-Array Fortran. Co-Array Fortran was used to reimplement halo update and allreduce in the barotropic conjugate gradient solver. The Co-Array Fortran implementation of the barotropic is more than 3 times faster than the MPI implementation for 64 processors, and scales much better.

POP Performance Summary

- Small communication latency was required to achieve good scalability for the POP “x1” benchmark.
- Good performance on the X1 was 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.

GYRO

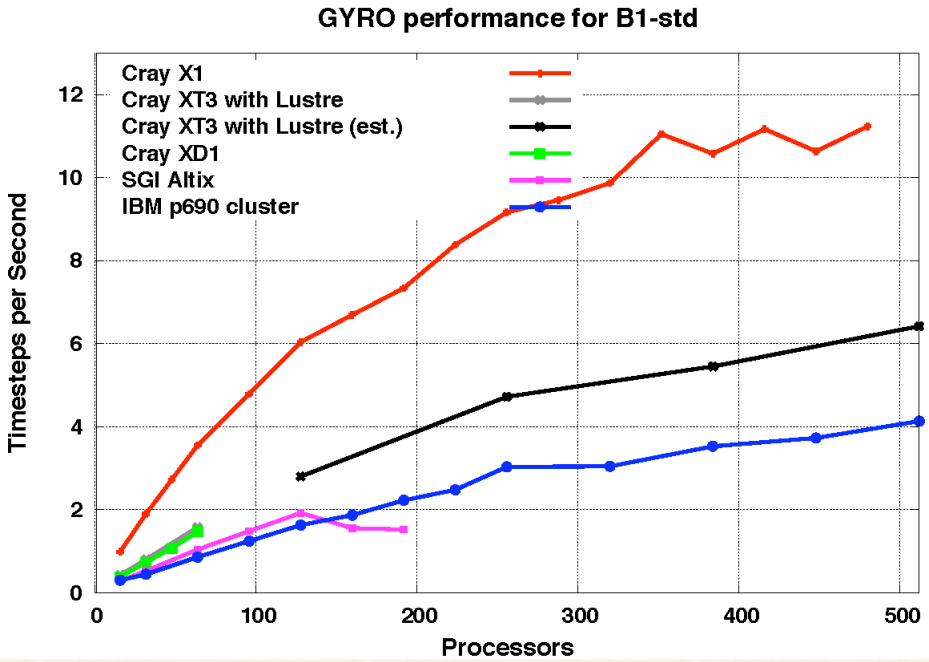
- GYRO is an Eulerian gyrokinetic-Maxwell solver developed by R.E. Walsh and J. Candy at General Atomics. It is used to study plasma microturbulence in fusion research.
- GYRO comes with ports to a number of different platforms. The port and optimization on the Cray X1 is primarily due to Mark Fahey of ORNL.
- GYRO is a pure MPI code (i.e., does not use SMP parallelism). In the Cray X1 experiments GYRO is run with one process per MSP.

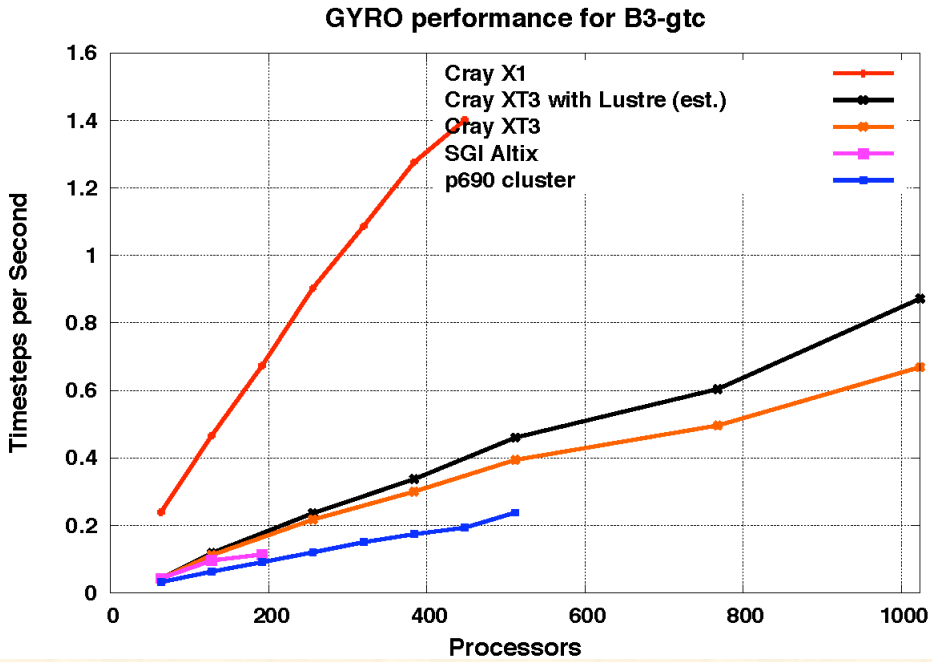
GYRO Experiment Particulars

Two benchmark problems, both time dependent:

1. B1-std
   - 16-mode simulation of electrostatic turbulence using kinetic electrons and ions and electron collisions. Duration is 500 timesteps.
2. B3-gtc
   - 64-mode adiabatic electron case. It is run on multiples of 64 processors. Duration is 100 timesteps.

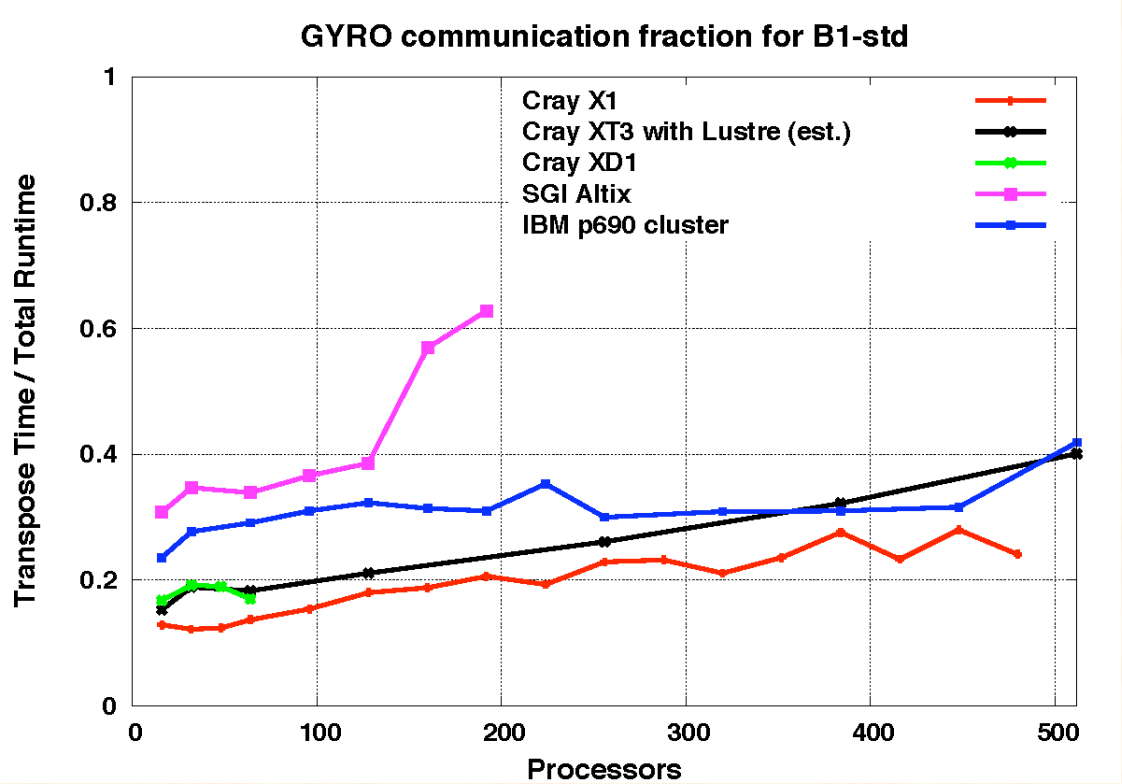
GYRO Platform Throughput Comparison





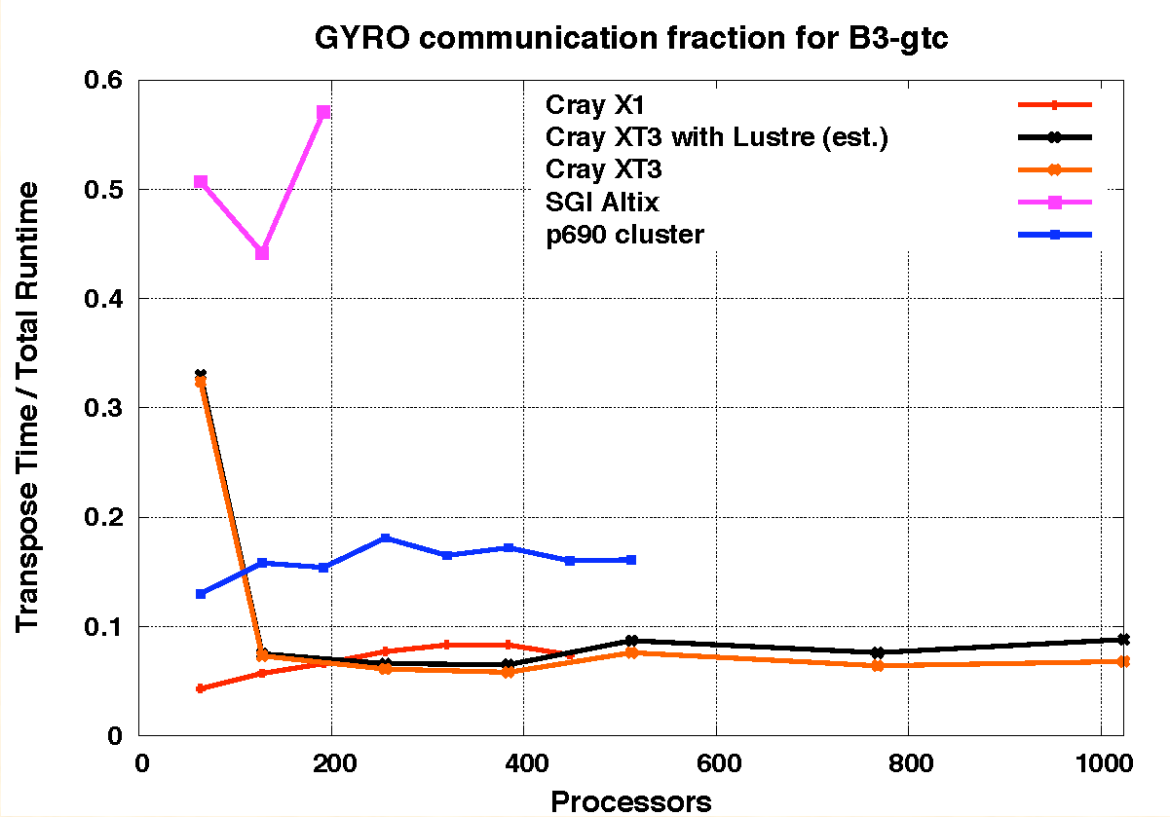
- Examining GYRO performance for both benchmarks on the Cray, IBM, and SGI systems.

GYRO Communication Analysis: B1-std



- Examining the fraction of time spent in the transposes used in the parallel implementation of GYRO for the different platforms and for B1-std. As the entire computational domain is being remapped during the transposes, the communication is bandwidth limited for all but the largest processor counts. The transposes are implemented using MPI_Alltoall, so the efficiency of the collective implementation is also important. All of the systems are scaling well except the Altix. The advantage of the high bandwidth performance on the X1 is especially evident.

GYRO Communication Analysis: B3-gtc



- Examining the fraction of time spent in the transposes used in the parallel implementation of GYRO for the different platforms and for B3-gtc. As before, performance is scaling well on all systems except the Altix. The smallest processor count, representing the largest message sizes, causes a performance problem on the XT3, but it scales extremely well for larger processor counts, doing as well as the X1 (relative to the processor speed).

GYRO Performance Summary

- High bandwidth is required to achieve good scalability for the GYRO benchmarks.
- All three Cray systems provide sufficient bandwidth to allow good scalability, especially for the large B3-gtc benchmark. In particular, communication/computation ratio is similar for all Cray systems for the GYRO benchmarks.

User Advice

- Serial optimization comes first.
  - Target appropriate granularity
  - Determine whether load imbalance causes serial performance problems to appear as communication overhead
- Good performance on the X1 requires good vectorization.
  - Need large vector fraction
  - Need reasonable vector lengths for loops that do vectorize (target is either 64 or 256)
- Memory access patterns can be important.
  - X1 has a cache

  However, memory subsystem performance is a strength of this architecture.

User Advice (cont.)

- Interprocessor communication performance *can be* excellent.
  - Point-to-point and cross-sectional bandwidth are very high
  - Latency is low if use SHMEM or Co-Array Fortran (or UPC?)
- Multilevel parallelism means that there are multiple opportunities for performance optimization.
  - Vectorization
  - Streaming
  - Threads (e.g. OpenMP)
  - Processes (distributed memory parallelism)
- Multiple interprocessor communication options also provide an opportunity for performance optimization.
  - MPI
  - SHMEM
  - Co-Array Fortran / UPC

Measurement Gotchas

- Task Migration
  - Timings can be (significantly) perturbed if your job is migrated during a timing run.
- Timer overhead
  - Do not use `gettimeofday` for timing. Instead, use `rtc` (Fortran) or `_rtc` (C), to minimize overhead. MPI_WTIME cost is reasonable for coarse grain measurements.
  - Use timers sparingly, and outside inner loops. Even with `rtc`, perturbation due to instrumentation can be high.
  - Run with timers both enabled and disabled, to determine perturbation.
- Performance variability
  - Activity on the front-end and on filesystems have perturbed performance in the past (may be resolved now?)
  - Always measure performance multiple times.

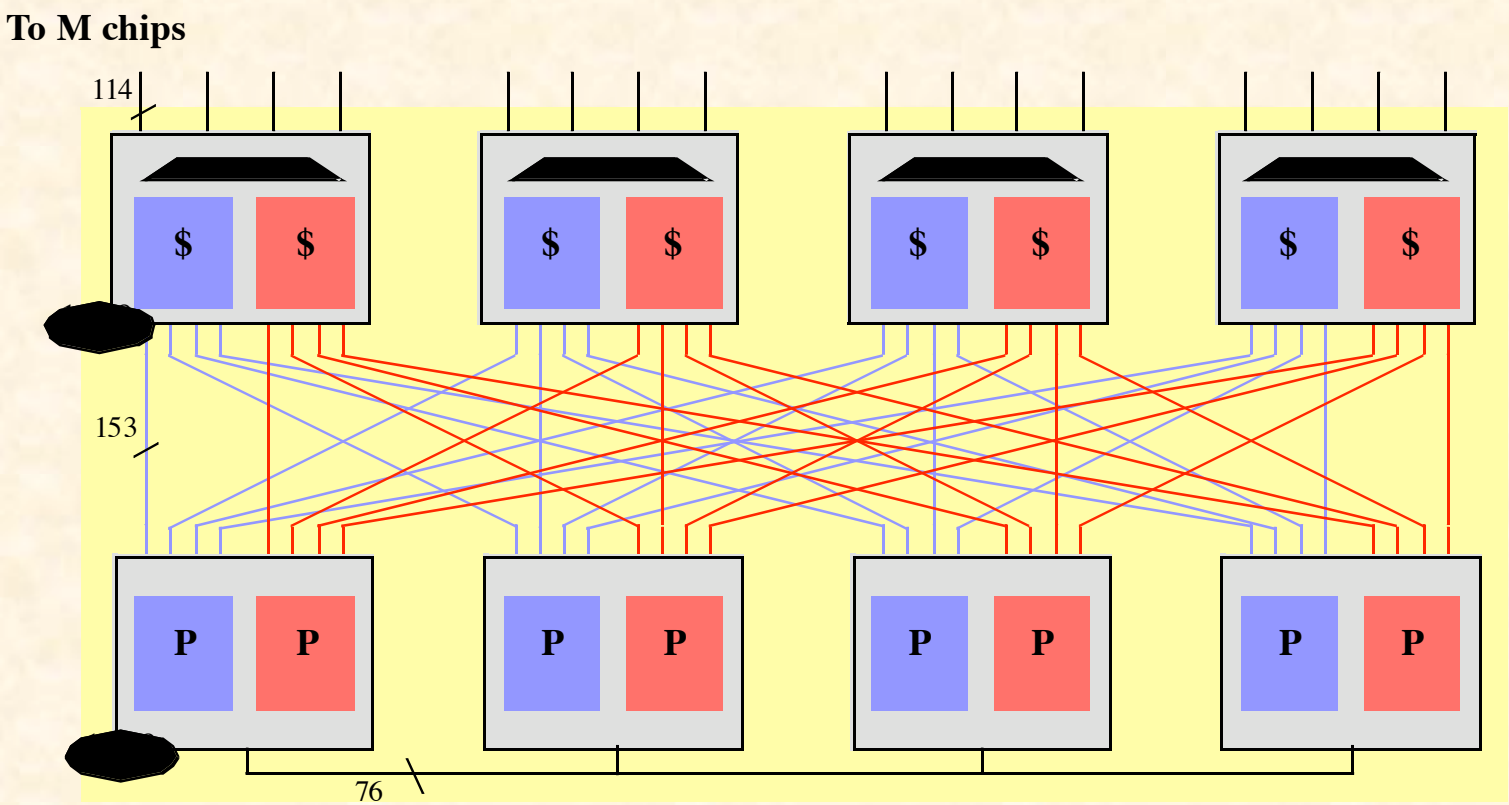
Phoenix (after July 2005)

Cray X1E with 256 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 1134 MHz and one 2-way superscalar unit running at 567 MHz per SSP
- 2 MB Ecache per MSP
- 16 GB of memory per node

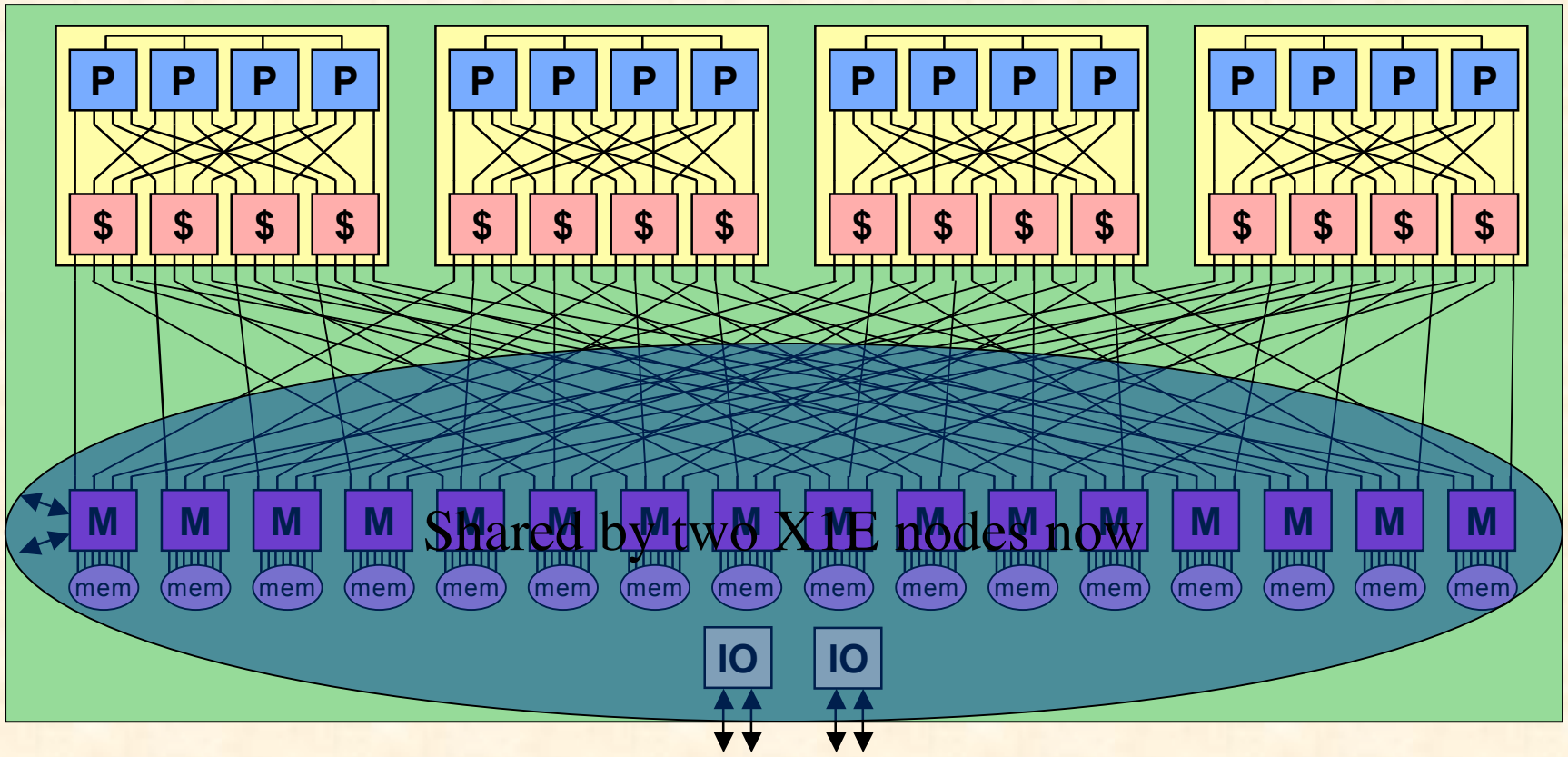
for a total of 1024 processors (MSPs), 2048 GB of memory , and ~18600 GF/s peak performance.

X1E MCM



- Re-implement P and E chips in 0.08mm IC technology
- Place two MSPs on each MCM
  - ⇒ Double the processor density (2 –4 MSP nodes/module)
- MCM frequency increase (400 MHz – 565 MHz, 1.41x)

Cray X1 Node



Inter node network:

- 2 ports per M chip
- 1.6 GB/s full duplex per link
- 102.4 Gbyte/s to the network

I/O connections:

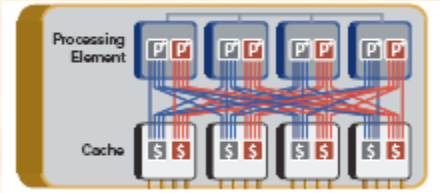
- 4 ports per node
- 1.2 GB/s full duplex per link

Local memory:

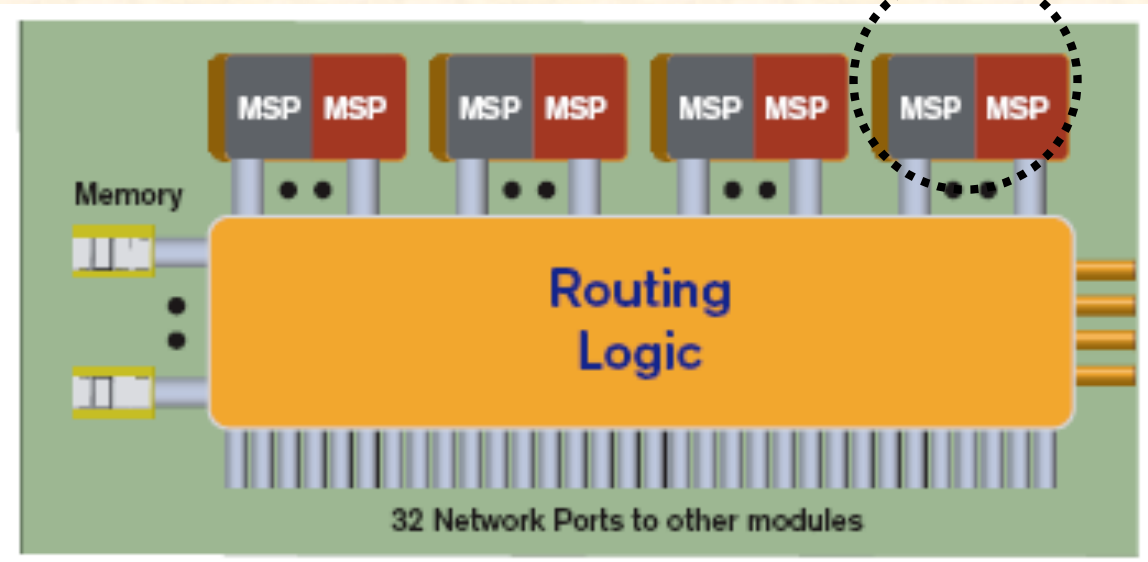
- 200 GB/s peak bw
- 16-32 GB per node

Cray X1E Compute Module

- Hardware building block
- Contains 8 MSPs organized as two 4-MSP nodes
- Total of 144 GFLOPS per module



Local Memory Peak BW
= 16 sections x 12.8 GB/s/section
= 204.8 GB/s



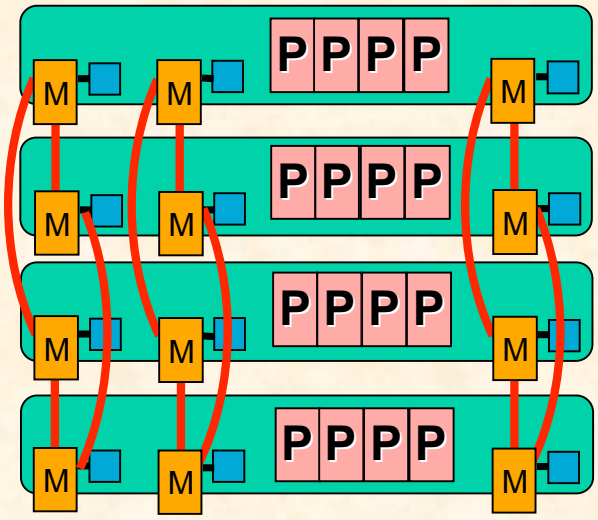
SPC I/O channels
1.2 x 2 x 2 = 4.8 GB/s

Interconnect network
3.2 x 2 x 16 =102.4 GB/s (51.2 GB/s per direction)

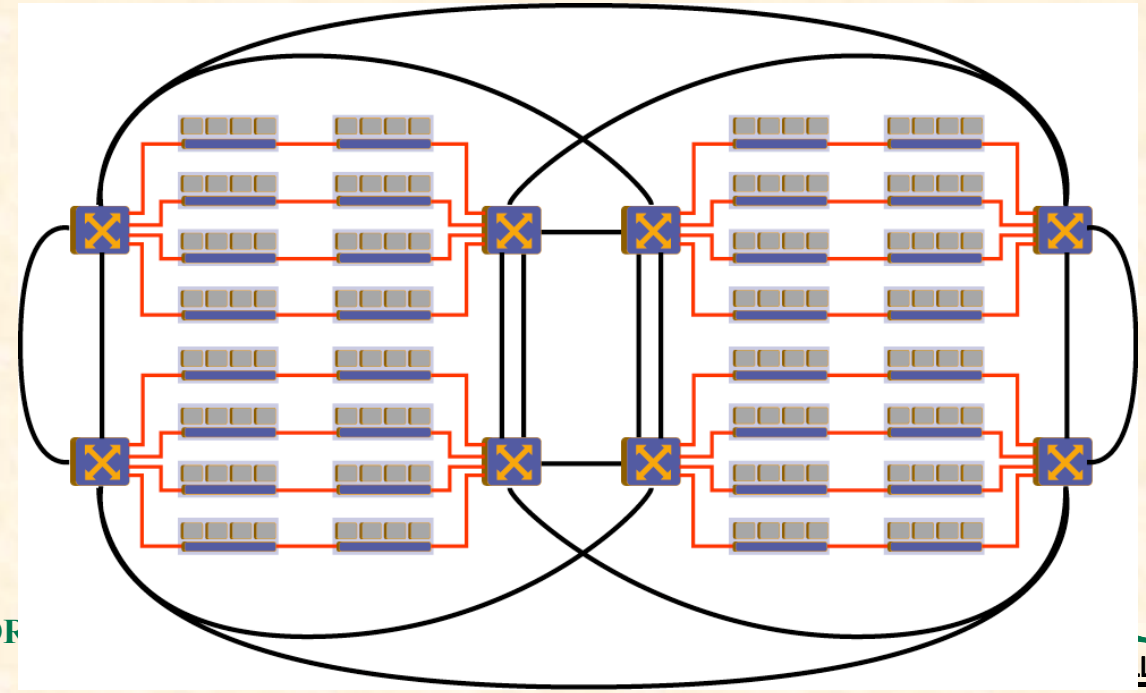
Cray X1E Interconnect Network

- Topology depends on the # of compute modules
  - ≦ 4 modules : direct connection
  - ≦ 128 modules : hypercube
  - ＞ 128 modules : modified 2D torus

direct connection



hypercube



For More Information …

- http://www.ccs.ornl.gov
- http://www.csm.ornl.gov/evaluation
- http://www.csm.ornl.gov/~worley
- worleyph@ornl.gov