Catamount N-Way Performance on XT5

Ron Brightwell, Suzanne Kelly, Jeff Crow

Scalable System Software Department

Sandia National Laboratories

rbbrigh@sandia.gov

Cray User Group Conference

May 6, 2009

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Catamount N-Way Lightweight Kernel

- Third-generation compute node operating system
- No virtual memory support
  - No demand paging
- Virtual addressing
  - Provides protection for the OS and privileged processes
- Multi-core processor support via Virtual Node Mode
  - One process per core
  - Memory is divided evenly between processes on a node
  - Processes are completely mapped when started
  - Physically contiguous address mappings

  - No support for POSIX-style shared memory regions
  - No support for threads
    - Previous generation LWK supported threads and OpenMP
    - Support was (reluctantly) removed in 2003 at Cray's request

Sandia's Huge Investment in MPI

- All Sandia HPC applications written in MPI
- Several are more than 15 years old
- More than a billion dollars invested in application development
- MPI has allowed for unprecedented scaling and performance
- Performance portability is critical for application developers
- Mixed-mode programming (MPI+threads) not very attractive

Message Passing Limitations on Multicore Processors

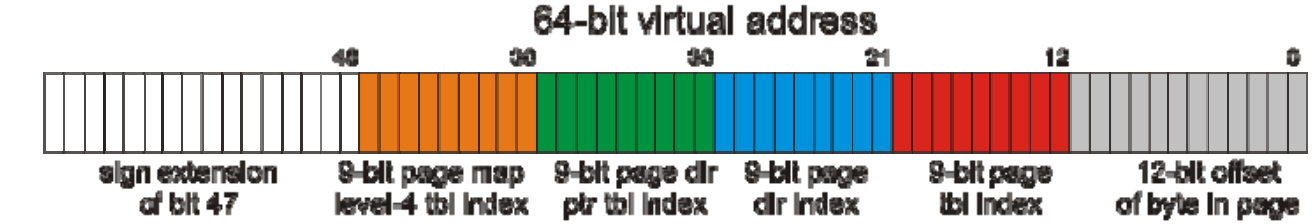
- Multi-core processors stress memory bandwidth performance
- MPI compounds the problem
  - Semantics require copying messages between address spaces
  - Intra-node MPI messages use memory-to-memory copies
  - Most implementations use POSIX-style shared memory
    - Sender copies data in
    - Receiver copies data out
- Alternative strategies
  - OS page remapping between source and destination processes
    - Trapping and remapping is expensive
    - Serialization through OS creates bottleneck
  - Network interface offload
    - Serialization through NIC creates bottleneck
    - NIC much slower relative to host processor

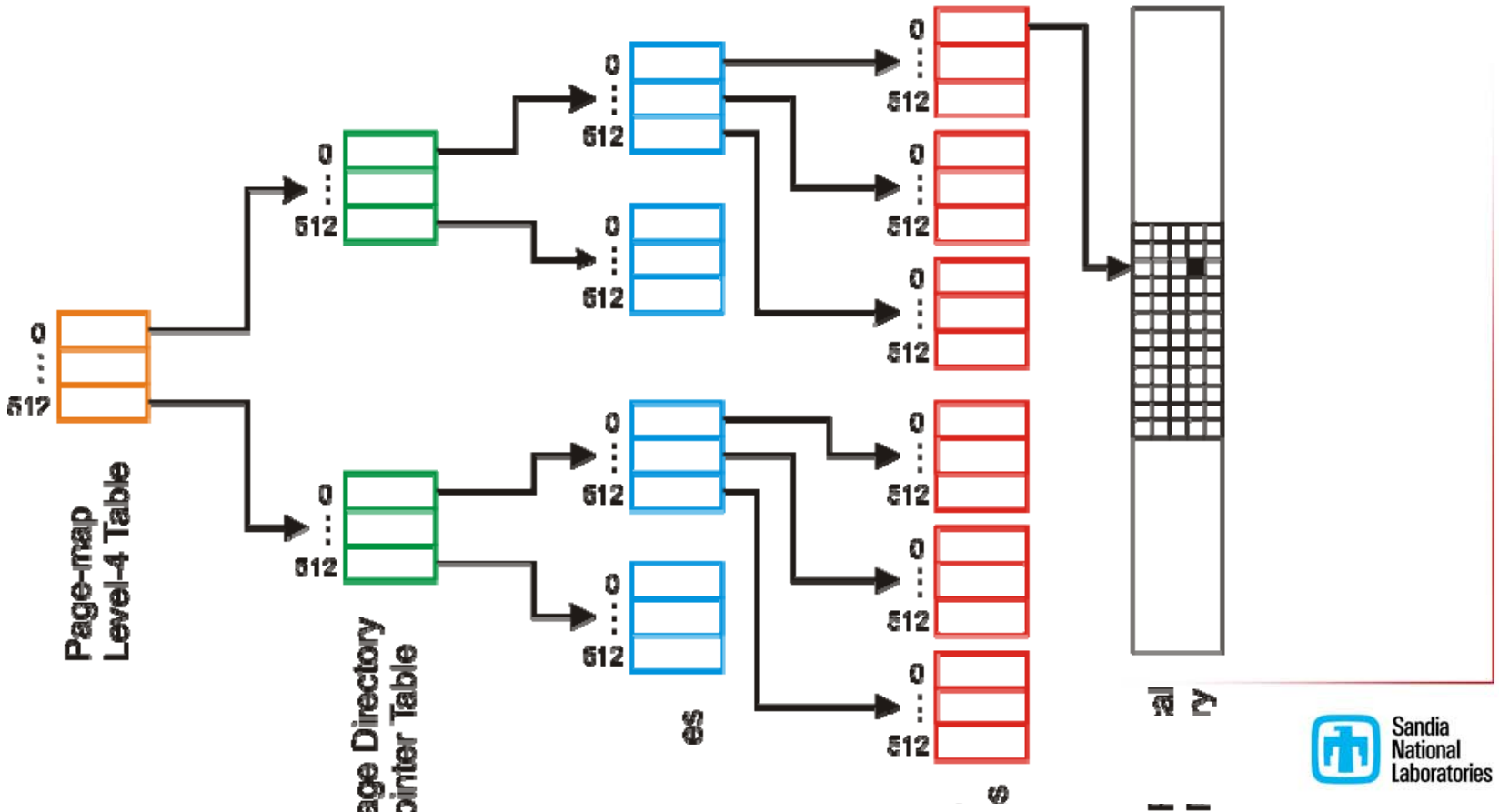
Intra-Node MPI for Cray XT with Catamount

- Uses Portals library for all messages
- Interrupt driven Portals implementation
  - "Generic" Portals (GP)
  - OS does memory copy between processes (<=512 KB)
  - OS uses SeaStar NIC (>512 KB)
  - Single copy
  - Serialization through OS
- NIC-based Portals implementation
  - "Accelerated" Portals (AP)
  - SeaStar does DMA between processes
  - Still need OS trap to initiate send
  - Single copy
  - Serialization through OS and SeaStar
- Both approaches create load imbalance

X86-64 Long PAE Page Table Lookup

(52-bit physical space, still 64-bit virtual space per process)

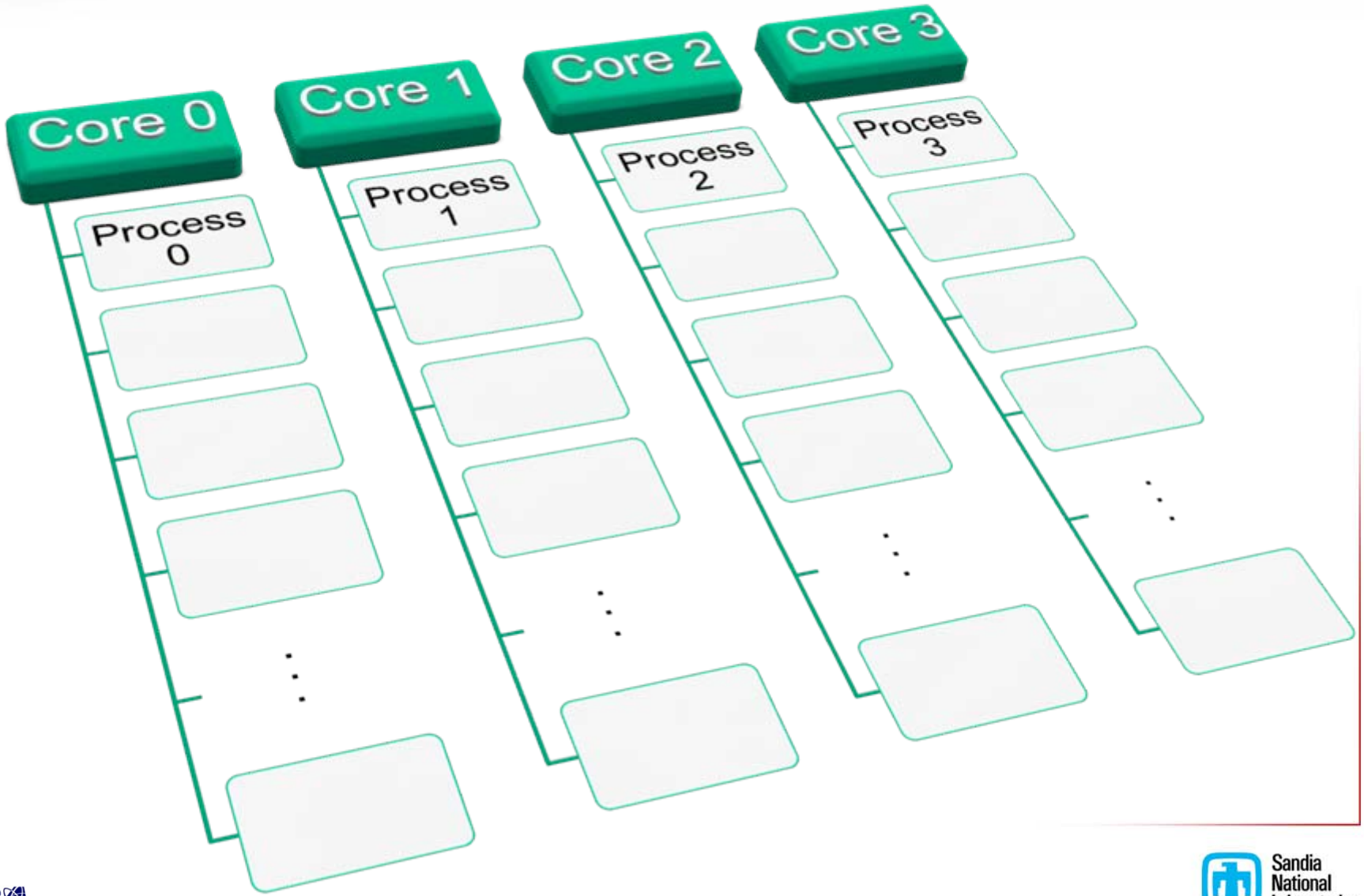




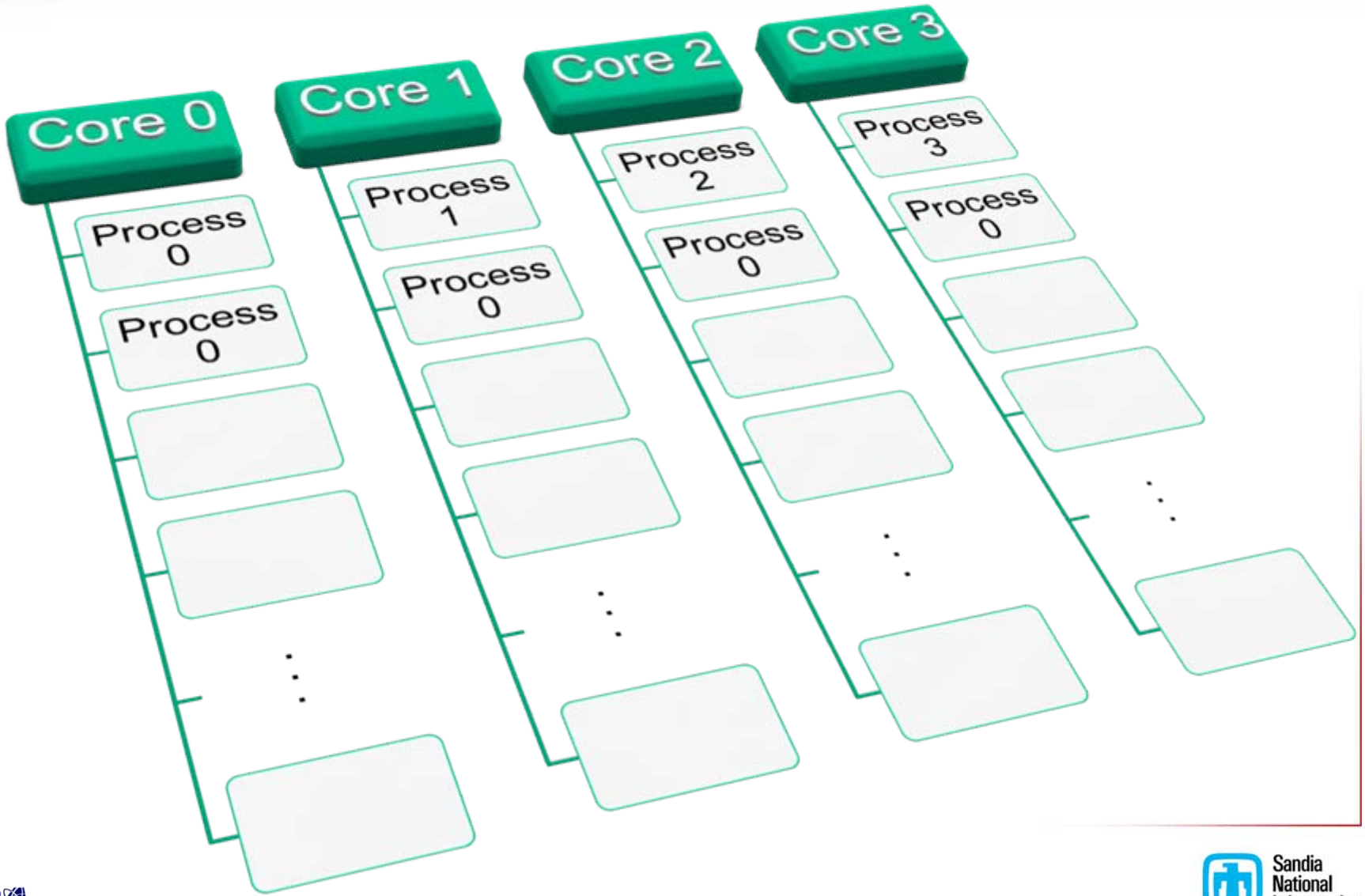
NNSA National Nuclear Security Administration

Sandia National Laboratories

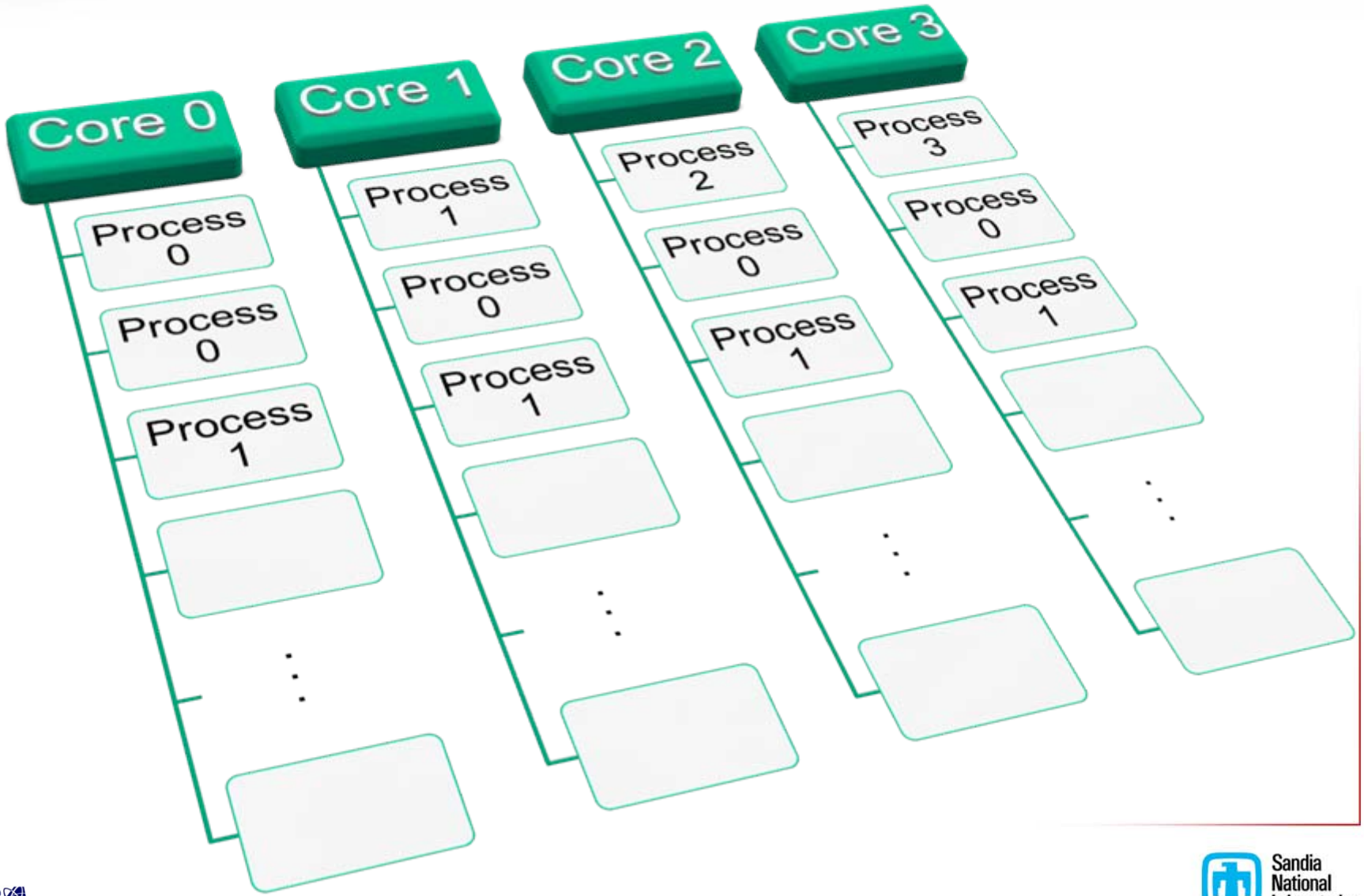
PML4 Mappings



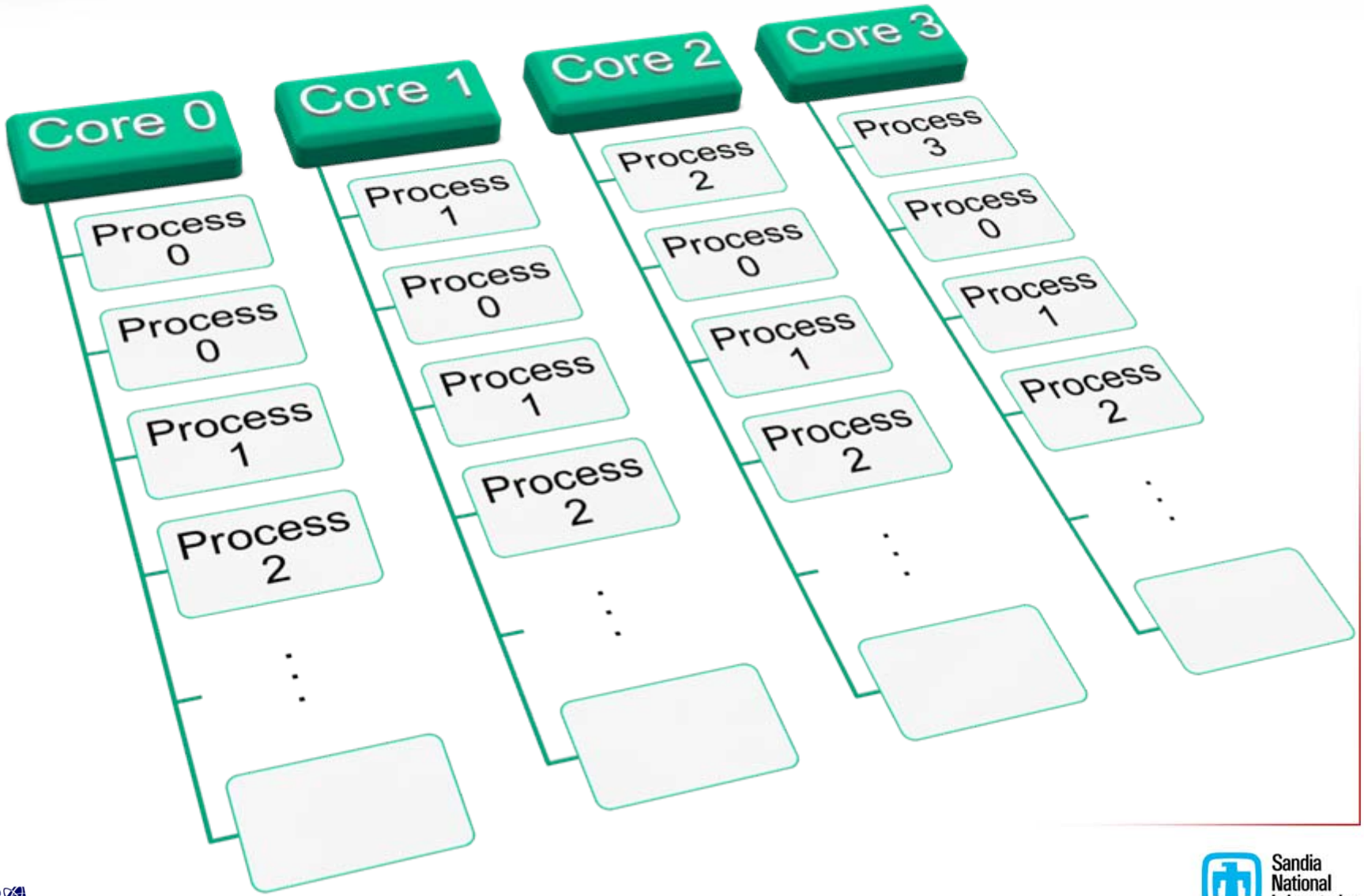
PML4 Mappings



PML4 Mappings



PML4 Mappings



SMARTMAP: Simple Mapping of Address Region Tables for Multi-core Aware Programming

- Direct access shared memory between processes
  - User-space to user-space
  - No serialization through the OS
  - Access to “remote” address by flipping a few bits
- Each process still has a separate virtual address space
  - Everything is “private” and everything is “shared”
  - Processes can be threads
- Allows MPI to eliminate *all* extraneous memory-to-memory copies on node
  - Single-copy MPI messages
  - No extra copying for non-contiguous datatypes
  - In-place collective operations
- Not just for MPI
  - Can emulate POSIX-style shared memory regions
  - Supports one-sided put/get operations
  - Can be used by applications directly

SMARTMAP Limitations on X86-64

- Limited to 511 processes per node
  - 512 PML4 slots
- Limited to 512 GB per process

- Won’t stress these anytime soon

Simplicity of a Lightweight Kernel

OS Code

```
static void initialize_shared_memory( void )
{
   extern VA_PML4T_ENTRY *KN_pml4_table_cpu[];
   int cpu;

   for( cpu=0 ; cpu < MAX_NUM_CPUS ;cpu++ ) {
     VA_PML4T_ENTRY * pml4 = KN_pml4_table_cpu[ cpu ];
       if( !pml4 ) continue;
         KERNEL_PCB_TYPE * kpcb =
                   (KERNEL_PCB_TYPE*)KN_cur_kpcb_ptr[cpu];
         if( !kpcb ) continue;
         VA_PML4T_ENTRY dirbase_ptr =(VA_PML4T_ENTRY)
                (KVTOP( (size_t) kpcb->kpcb_dirbase )
                | PDE_P
                | PDE_W
                | PDE_U );
         int other;
         for( other=0 ; other<MAX_NUM_CPUS ; other++ ){
           VA_PML4T_ENTRY * other_pml4 =
                             KN_pml4_table_cpu[other];
           if( !other_pml4 ) continue;
           other_pml4[ cpu+1 ] = dirbase_ptr;
         }
    }
}
```

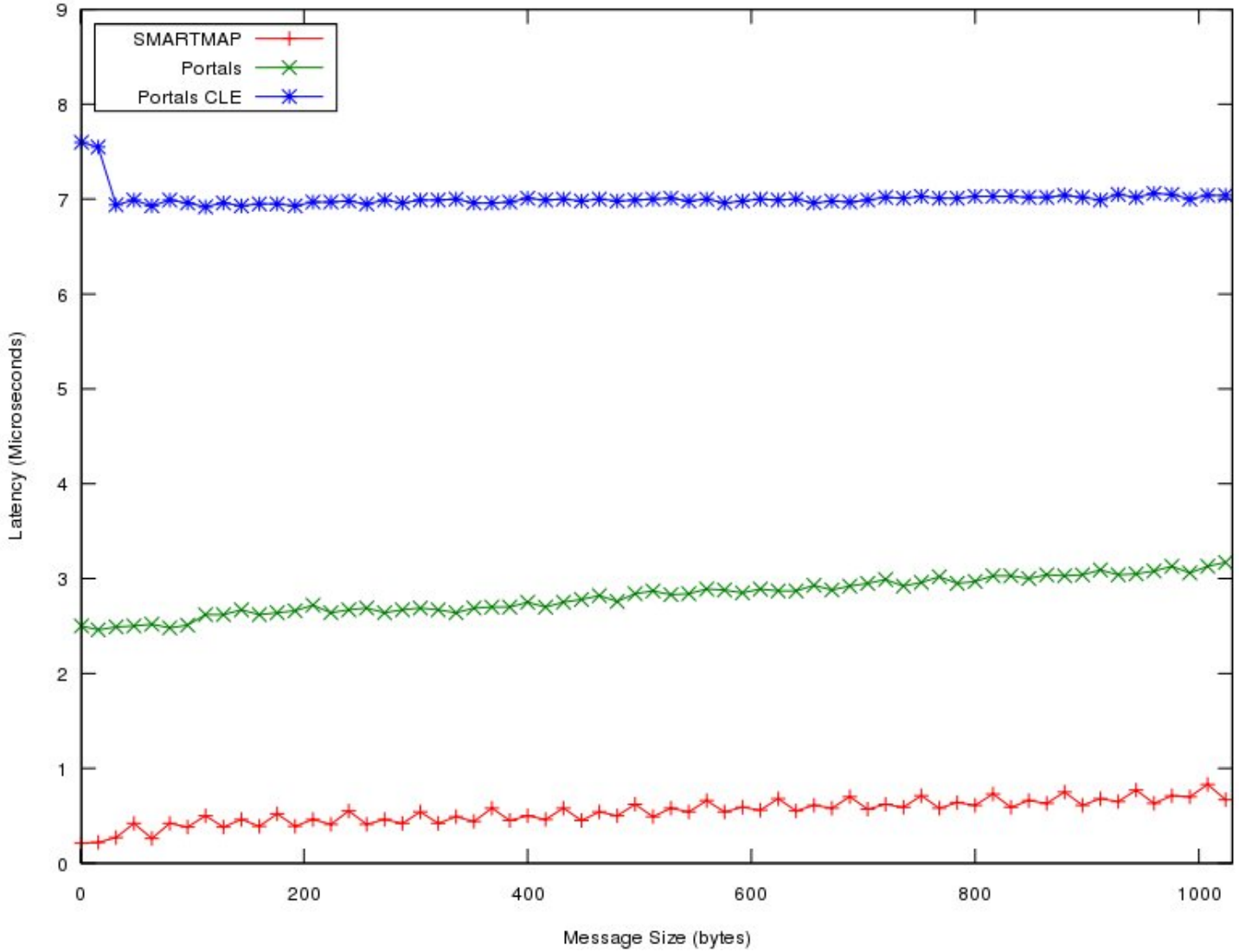
User Code

```
static inline void *
remote_address( unsigned core, volatile void * vaddr)
{
    uintptr_t addr = (uintptr_t) vaddr;
    addr |= ((uintptr_t) (core+1)) << 39;
    return (void*) addr;
}
```

Implementing Cray SHMEM

```
void shmem_ putmem( void *target, void *source, size_t length, int pe )
{
        int  core;

        if   ( (core = smap_pe_is_local( pe )) != -1 ) {

             void targetr = remote_address( core , target );

             memcpy( targetr, source, length );

        } else {

             pshmem_putmem( target, source, length, pe );

        }

}
```

Cray SHMEM Put Latency



Open MPI

- Modular Component Architecture
- Point-to-point modules
  - Point-to-Point Management Layer (PML)
    - Matching in the MPI library
    - Multiplexes over multiple transport layers (BTL)
      - Sockets, IB Verbs, shared memory, MX, Portals
  - Matching Transport Layer (MTL)
    - Matching in the transport layer
    - Only a single transport can be used
      - MX, Qlogic PSM, Portals
- Collective modules
  - Layered on MPI point-to-point
    - Basic, tuned, hierarchical
  - Directly on underlying transport

SMARTMAP MPI Point-to-Point

- Portals MTL
  - Each process has a
    - Receive queue for each core
    - Send queue
  - To send a message
    - Write request to the end of the destination receive queue
    - Wait for send request to be marked complete
  - To receive a message
    - Traverse send queues looking for a match
    - Copy message once match is found
    - Mark send request as complete
- Shared Memory BTL
  - Emulate shared memory with SMARTMAP
    - One process allocates memory from its heap and publishes this address
    - Other processes read address and convert it to a “remote” address

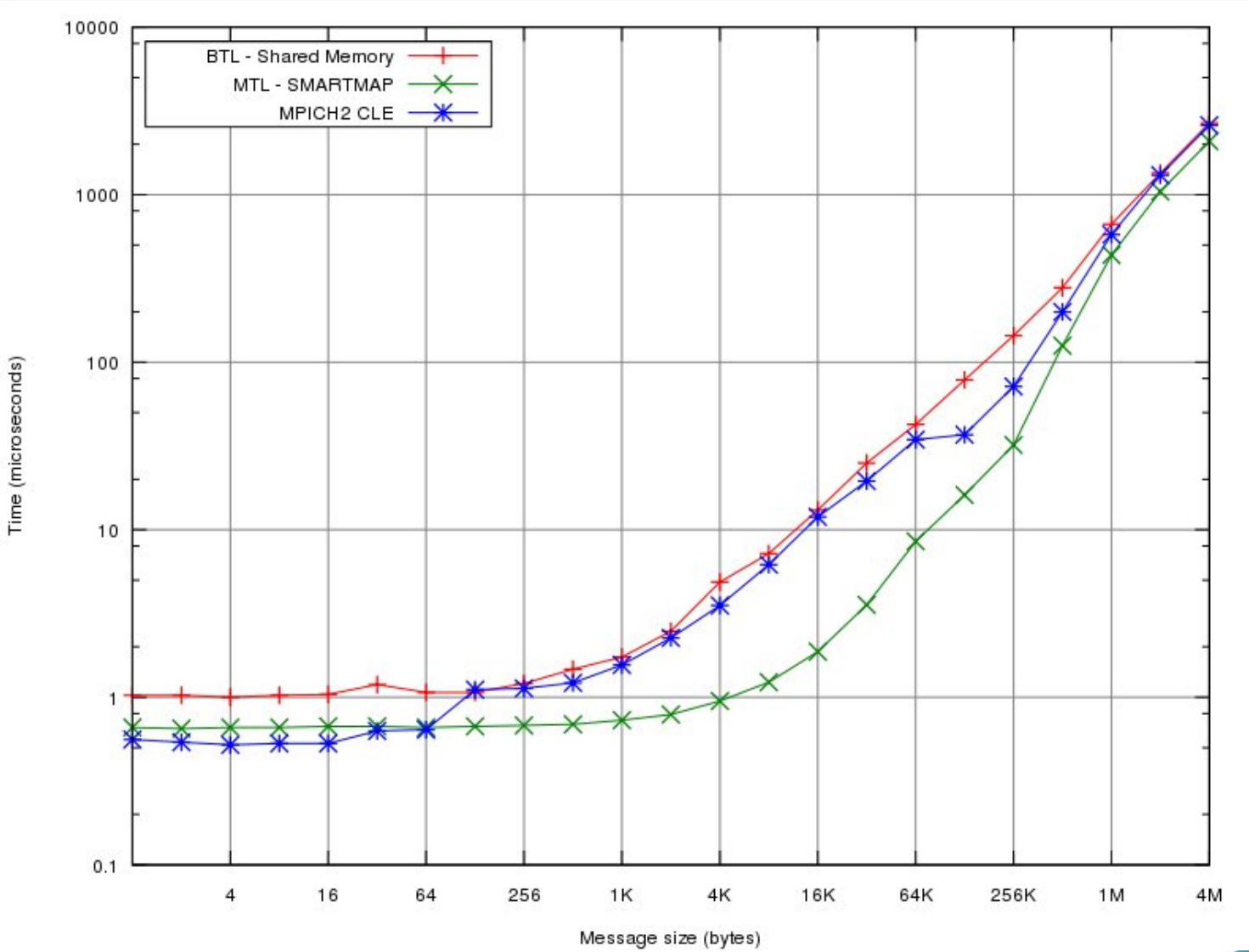
Portals MTL Limitations

- Messages are synchronous
  - Data is not copied until receiver posts matching receive
  - Send-side copy defeats the purpose
- Two posted receive queues
  - One inside Portals for inter-node messages
  - One in shared memory for intra-node messages
- Handling MPI_ANY_SOURCE receives
  - Search unexpected messages
  - See if communicator is all on-node or all off-node
  - Otherwise
    - Post Portals receive and shared memory receive
    - Only use shared memory receive if Portals receive hasn't been used

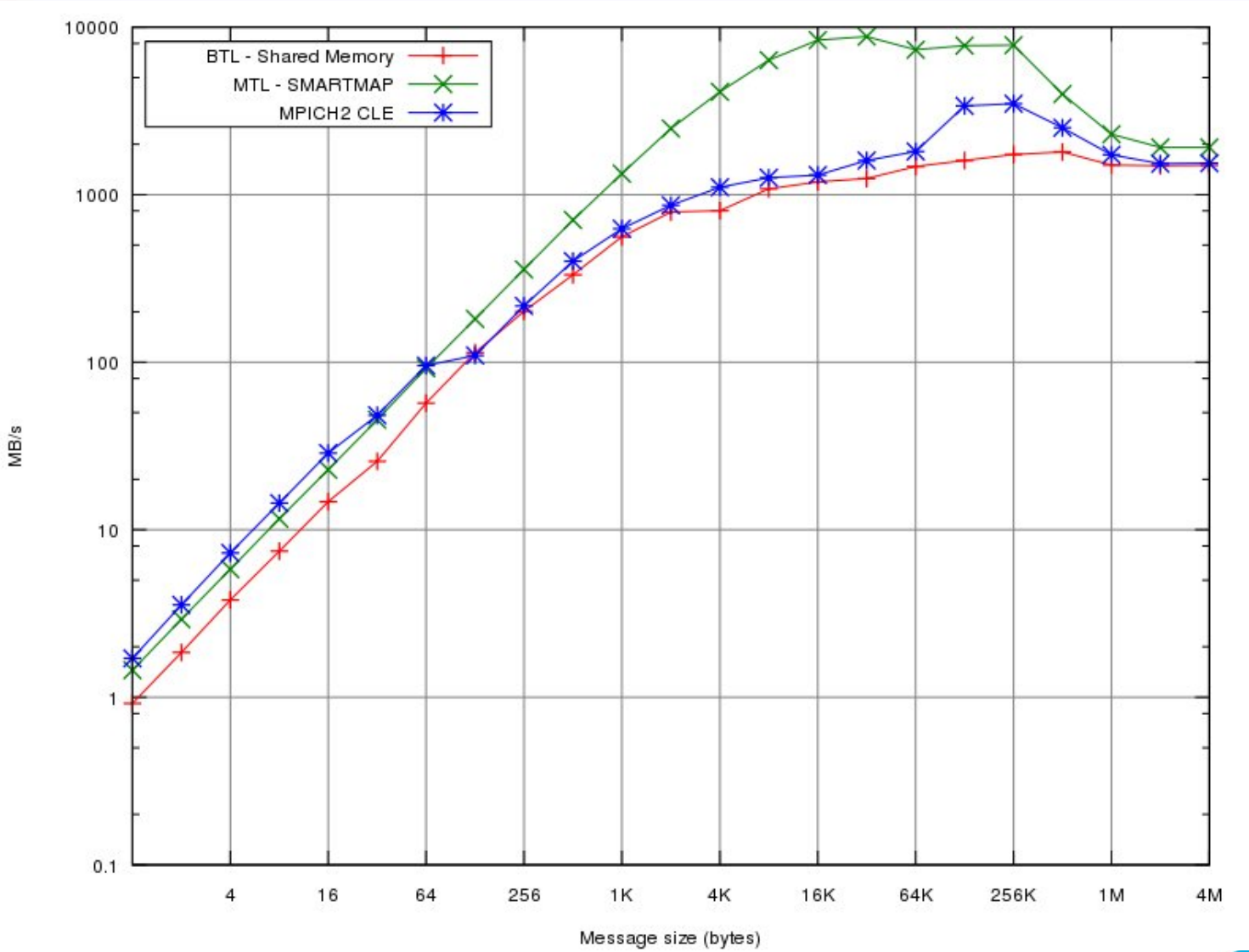
Test Environment

- Cray XT hardware
  - 2.3 GHz dual-socket quad-core AMD Barcelona
- Software
  - Catamount N-Way 2.1.41
  - Open MPI r17917 (February 2008)
- Benchmarks
  - Intel MPI Benchmarks (IMB) 2.3
  - MPI Message rate
    - PathScale modified OSU bandwidth benchmark
- Single node results

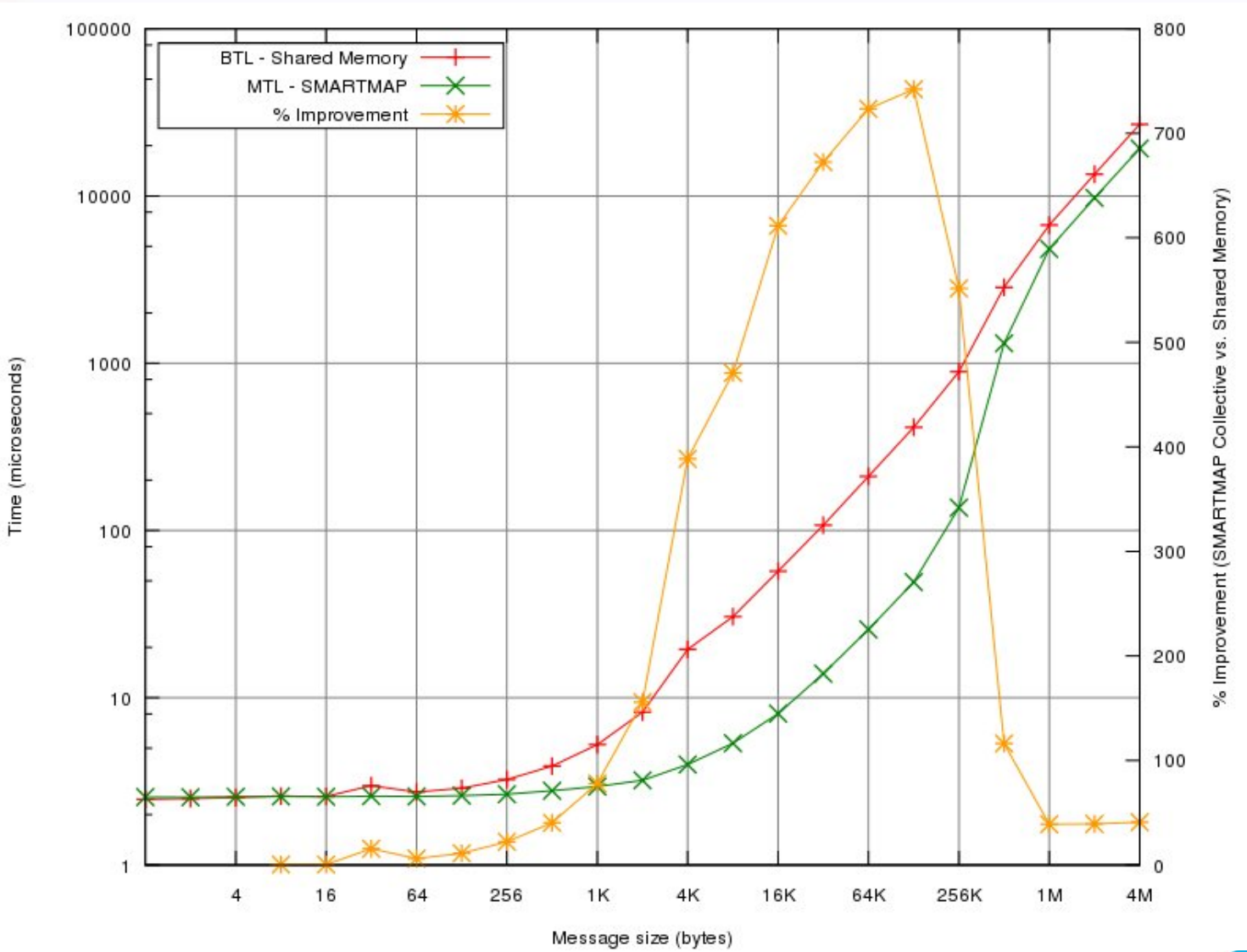
MPI Ping-Pong Latency



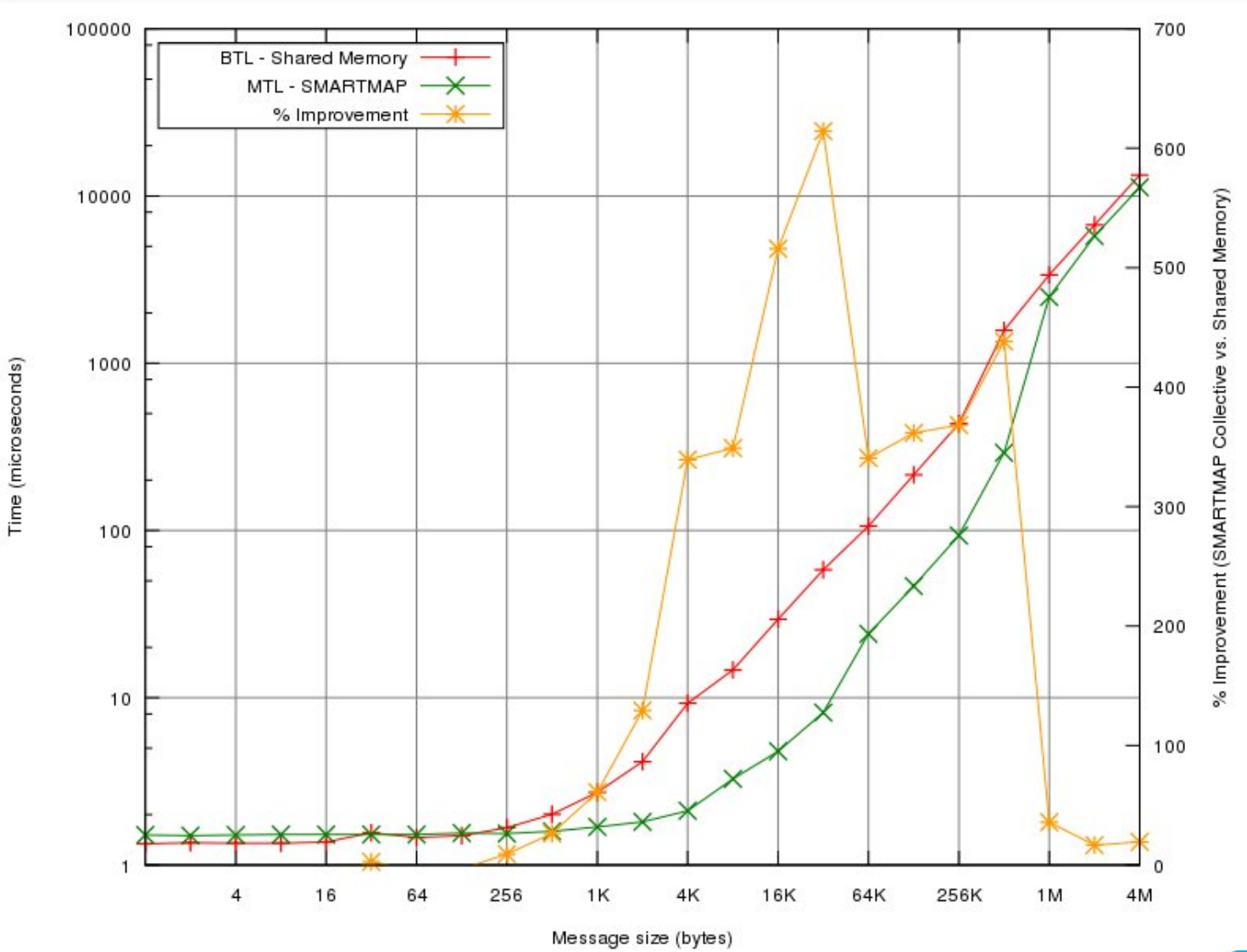
MPI Ping-Pong Bandwidth



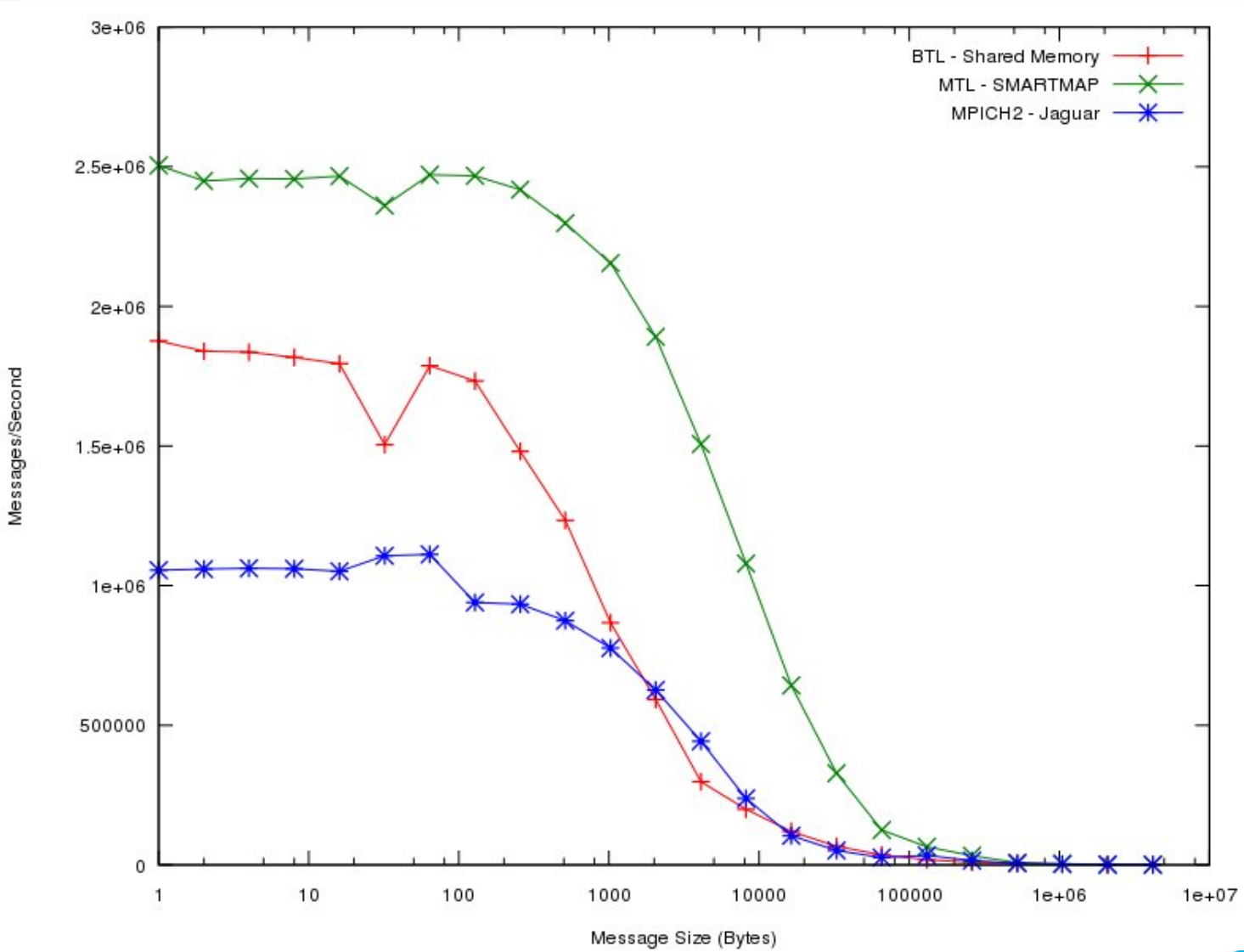
MPI Exchange – 8 cores



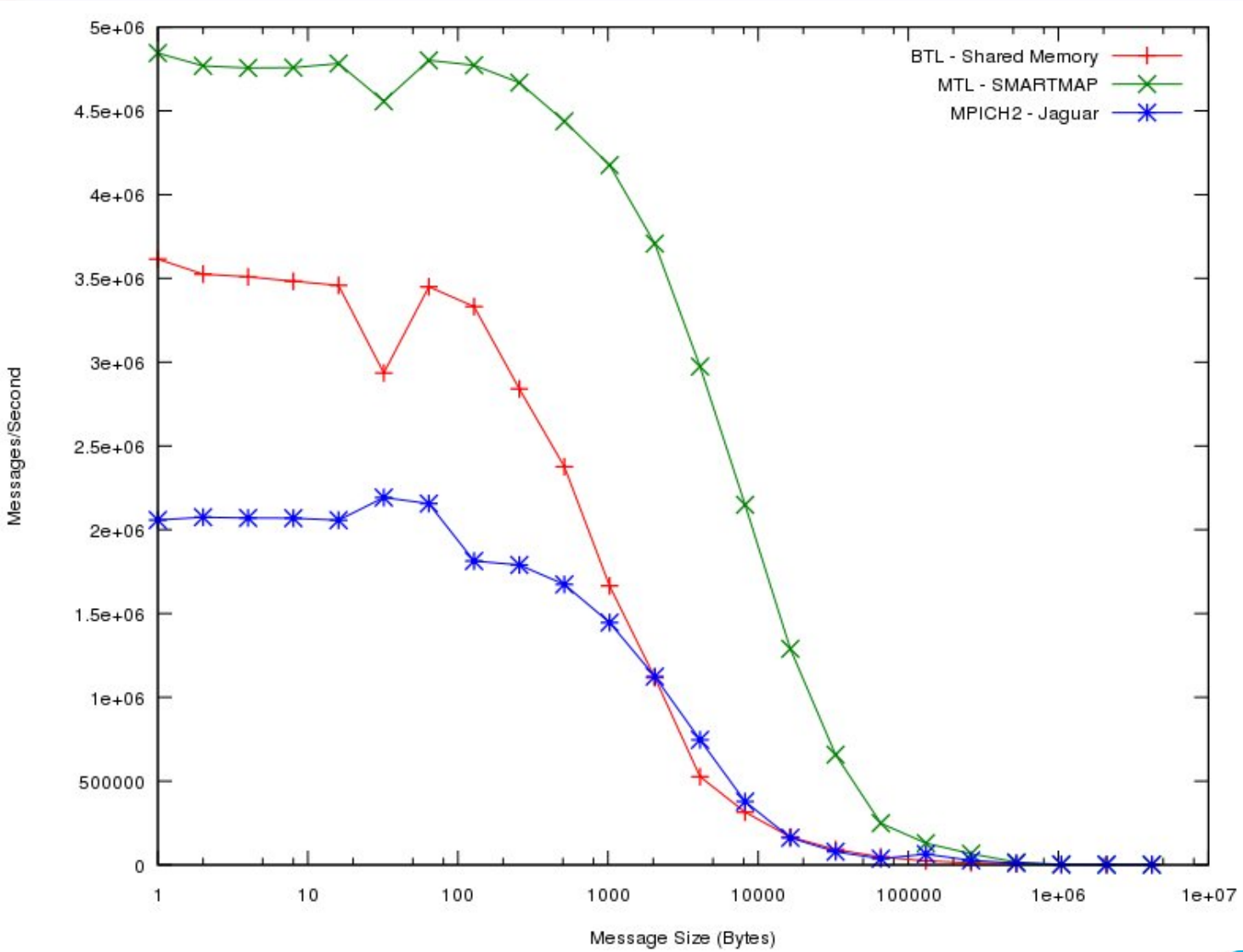
MPI Sendrecv – 8 cores



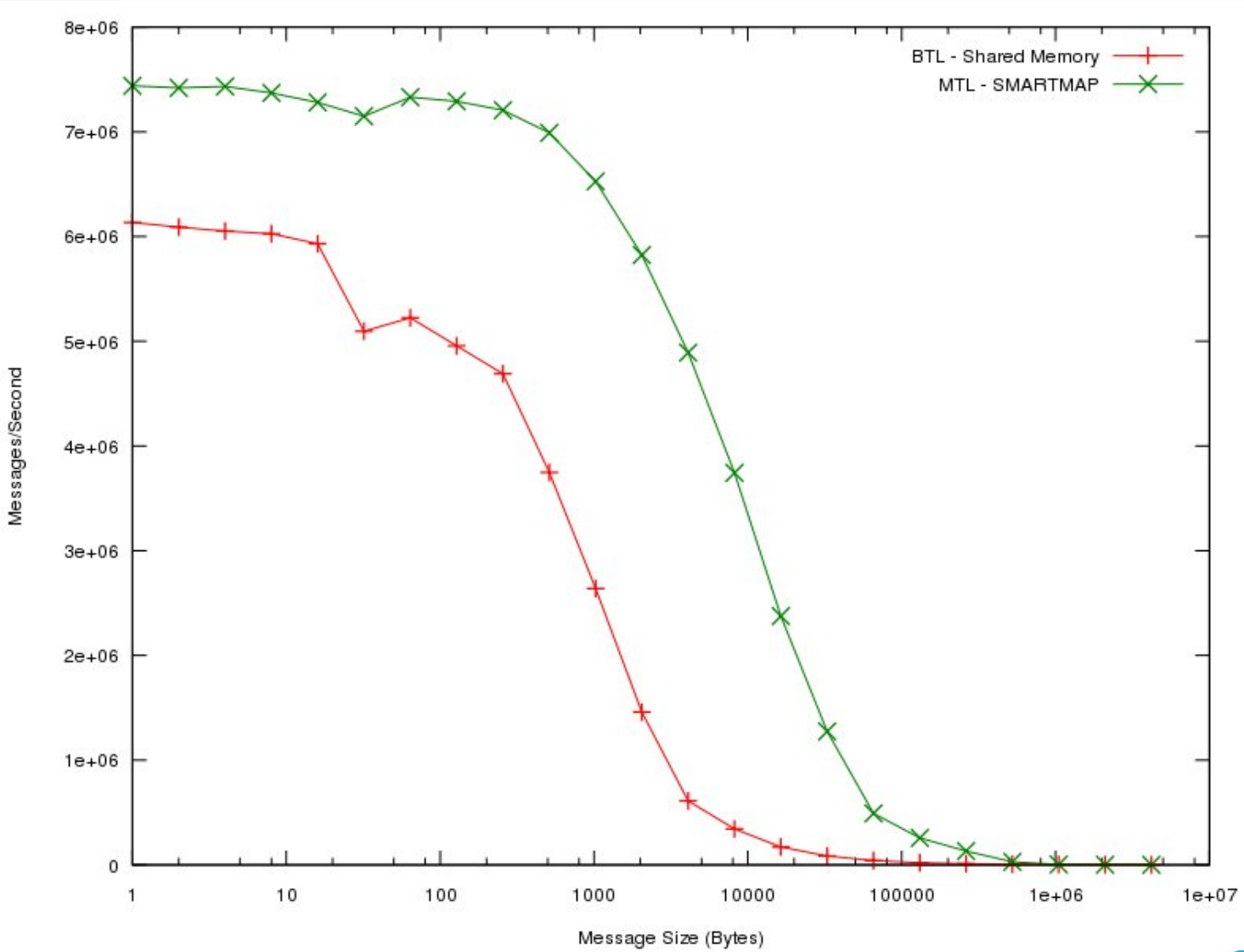
MPI Message Rate – 2 cores



MPI Message Rate – 4 cores



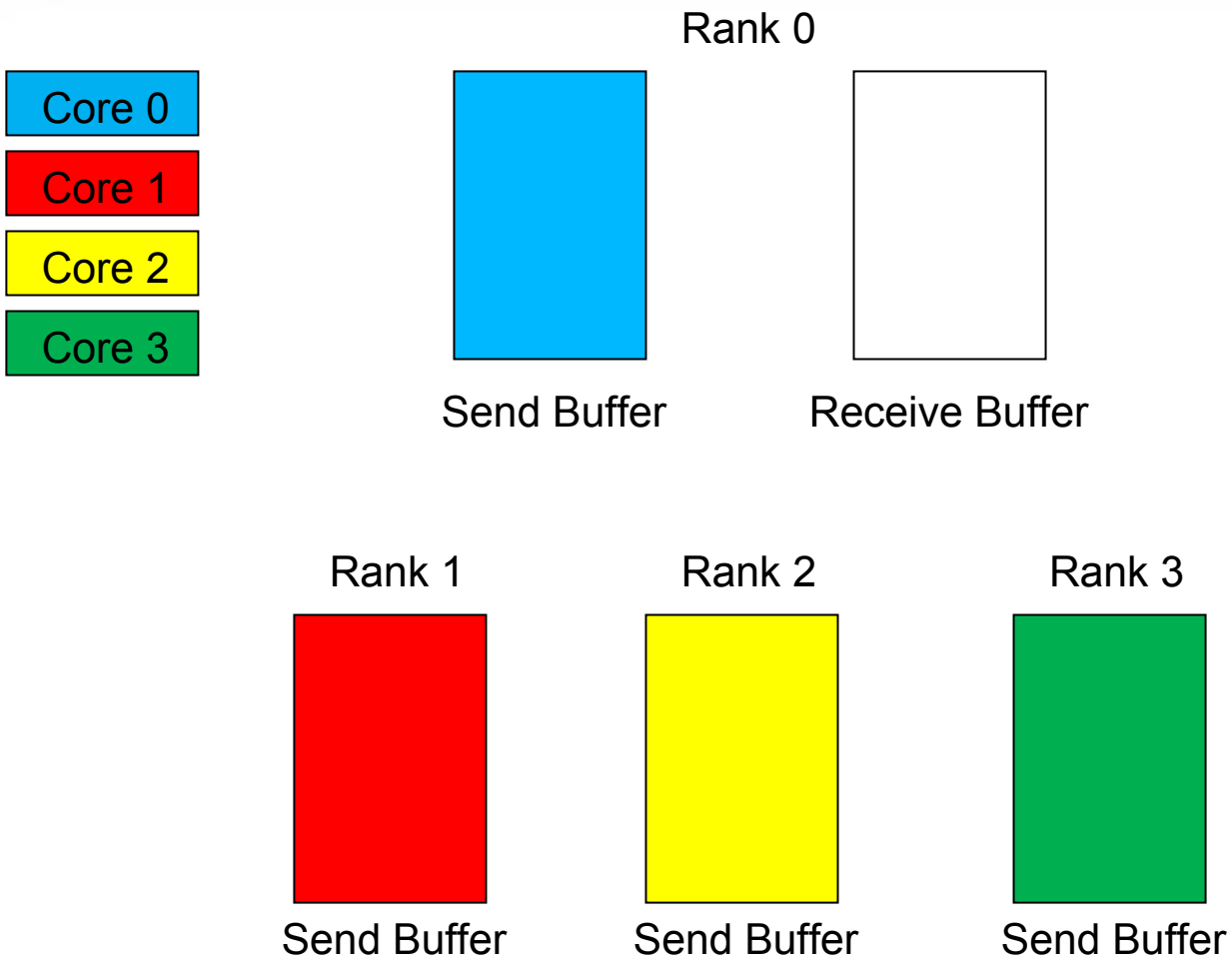
MPI Message Rate – 8 cores



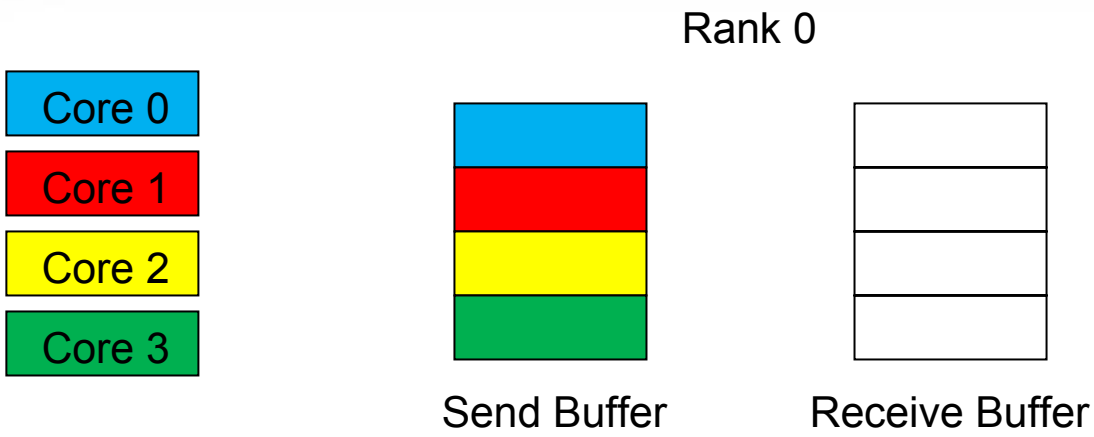
SMARTMAP MPI Collectives

- Broadcast
  - Each process copies from the root
- Reduce
  - Serial algorithm
    - Each process operates on root's buffer in rank order
  - Parallel algorithm
    - Each process takes a piece of the buffer
- Gather
  - Each process writes their piece to the root
- Scatter
  - Each process reads their piece from the root
- Alltoall
  - Every process copies its piece to the other processes
- Barrier
  - Each process atomically increments a counter

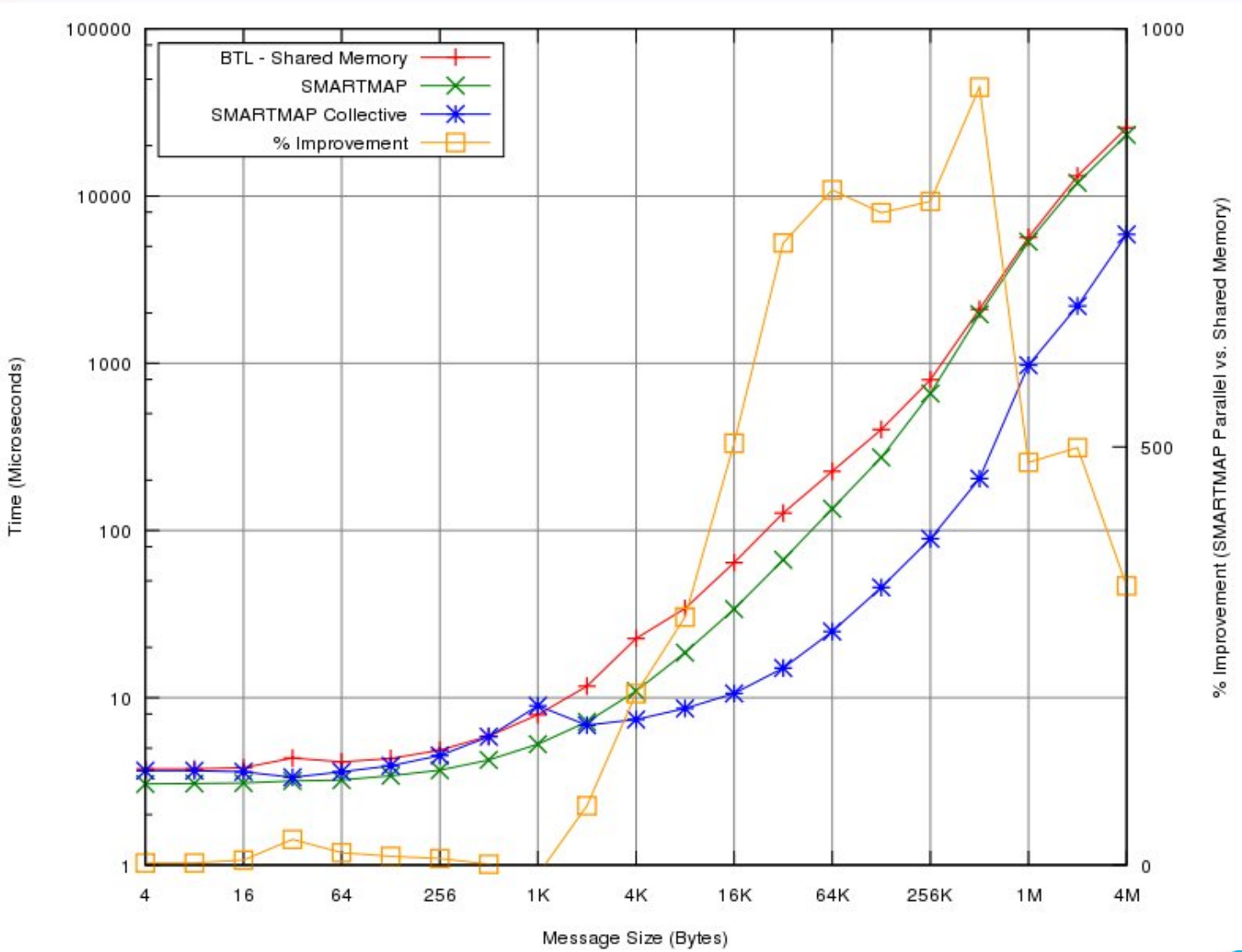
MPI Reduce - Serial



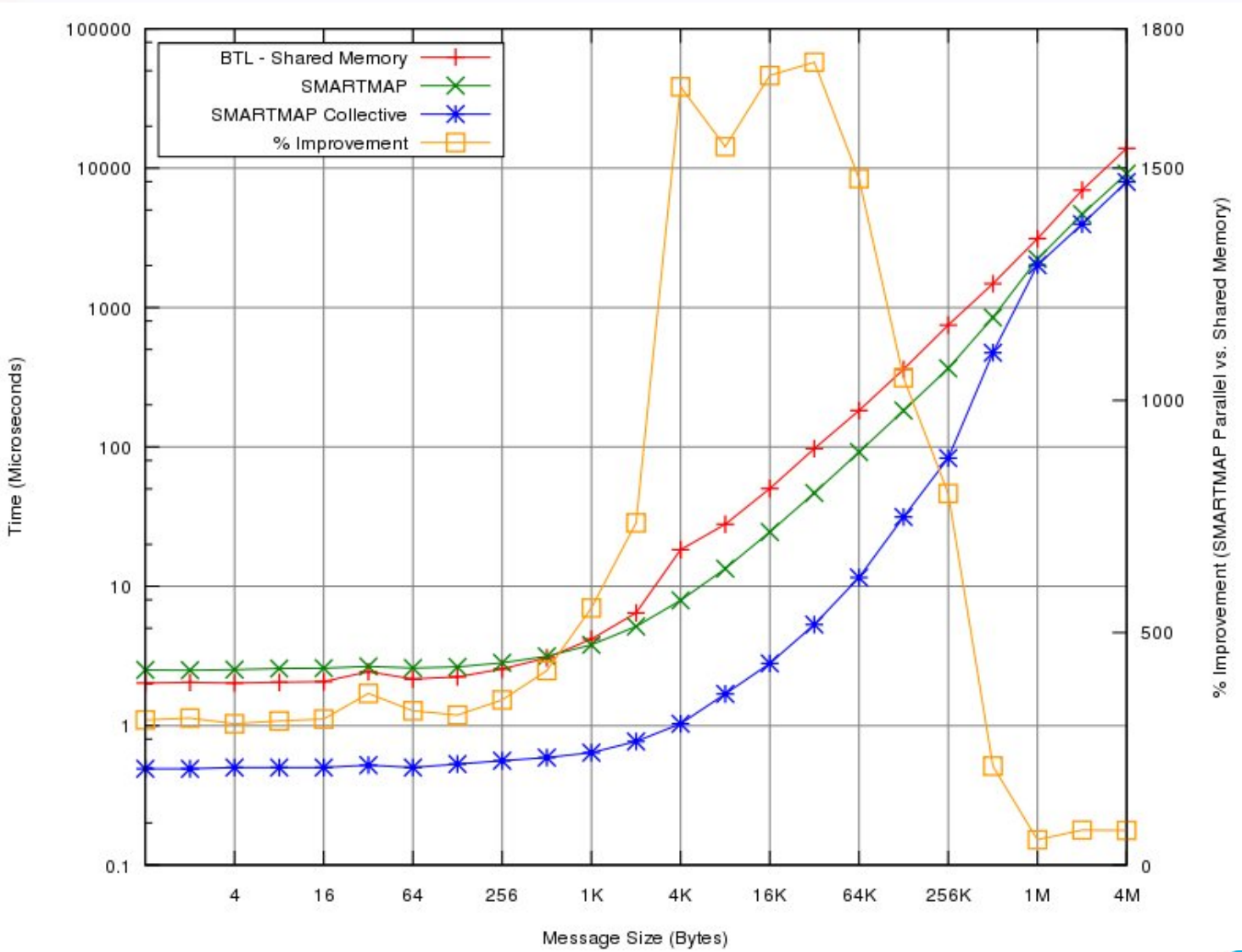
MPI Reduce – Parallel



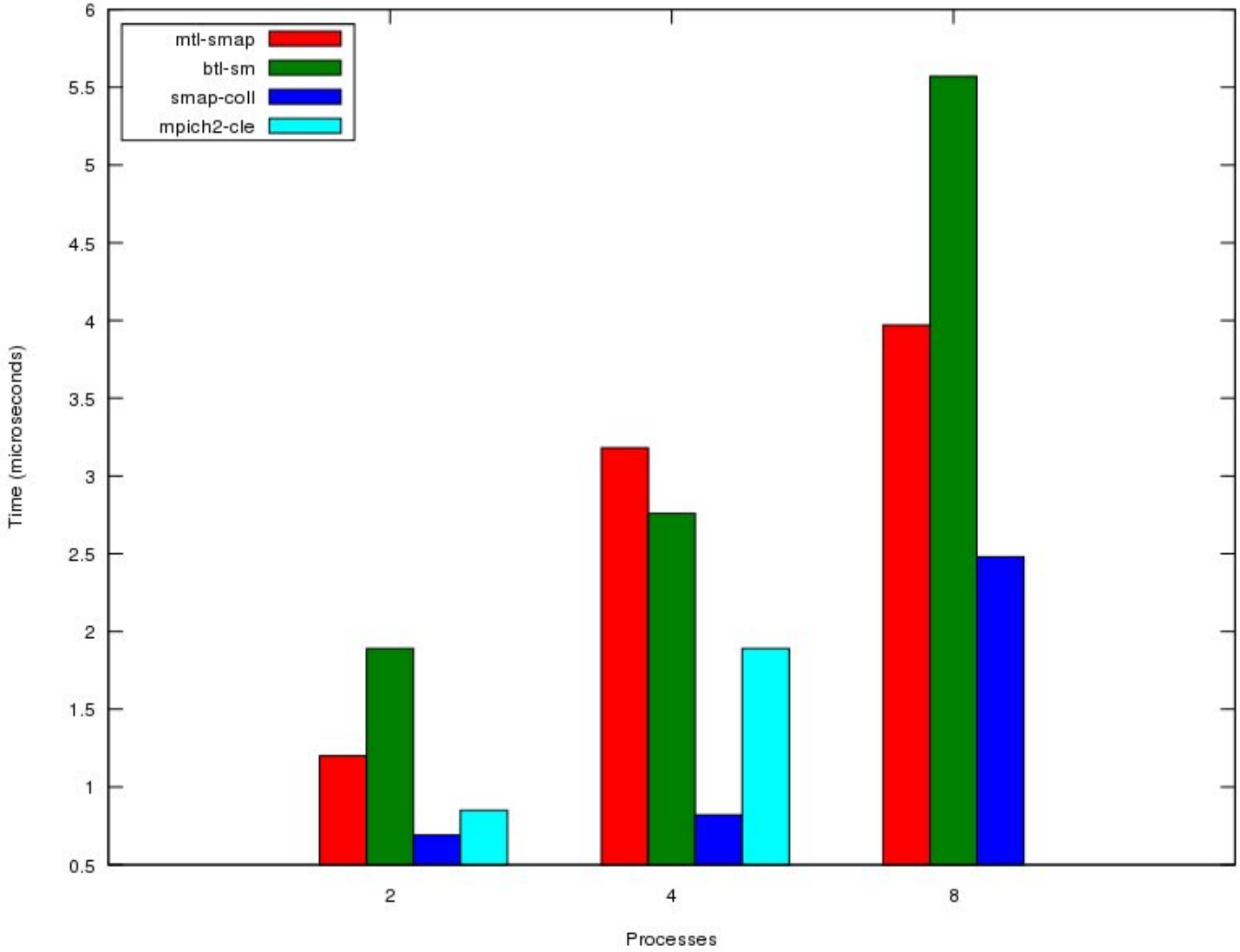
MPI Reduce – 8 cores



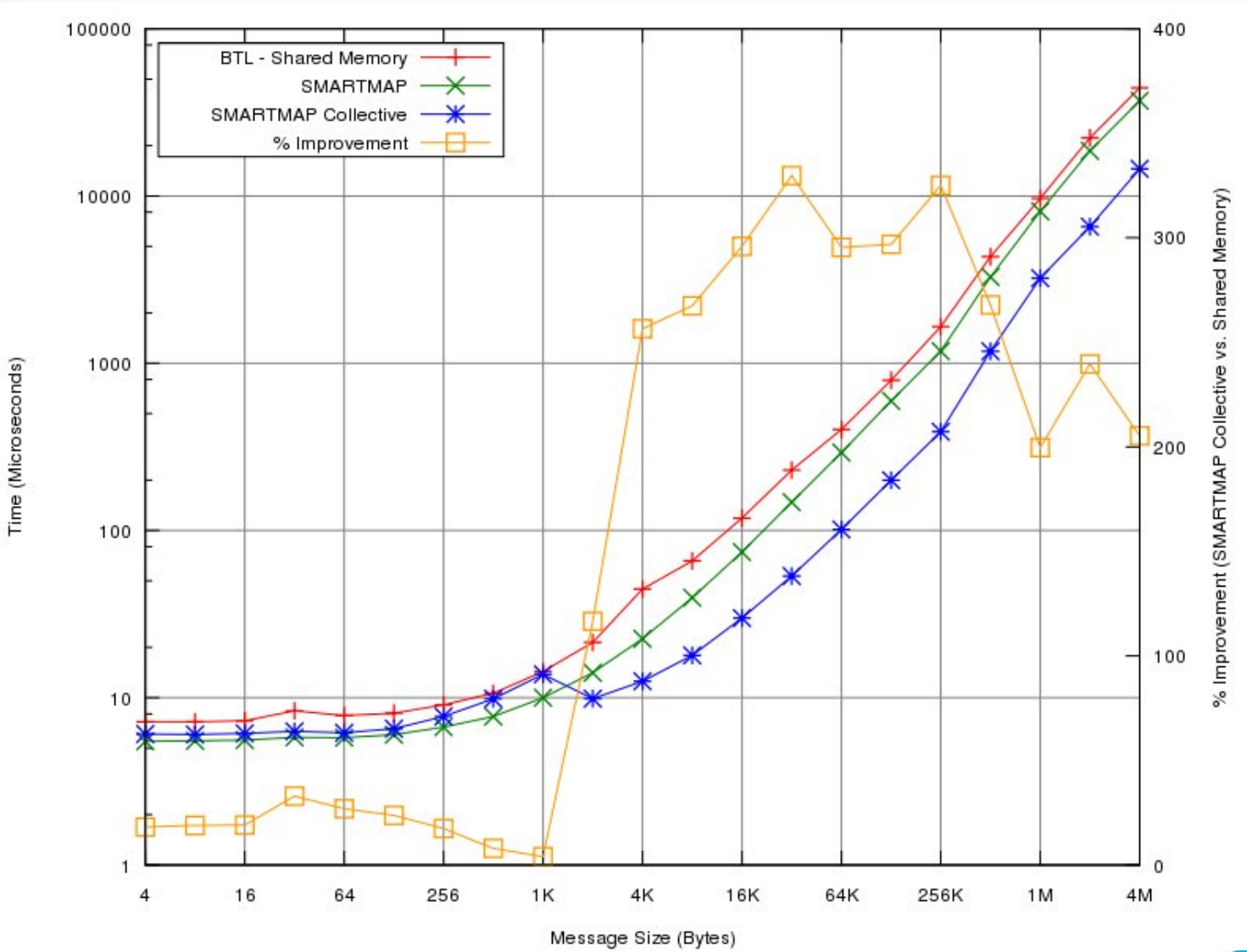
MPI Broadcast – 8 cores



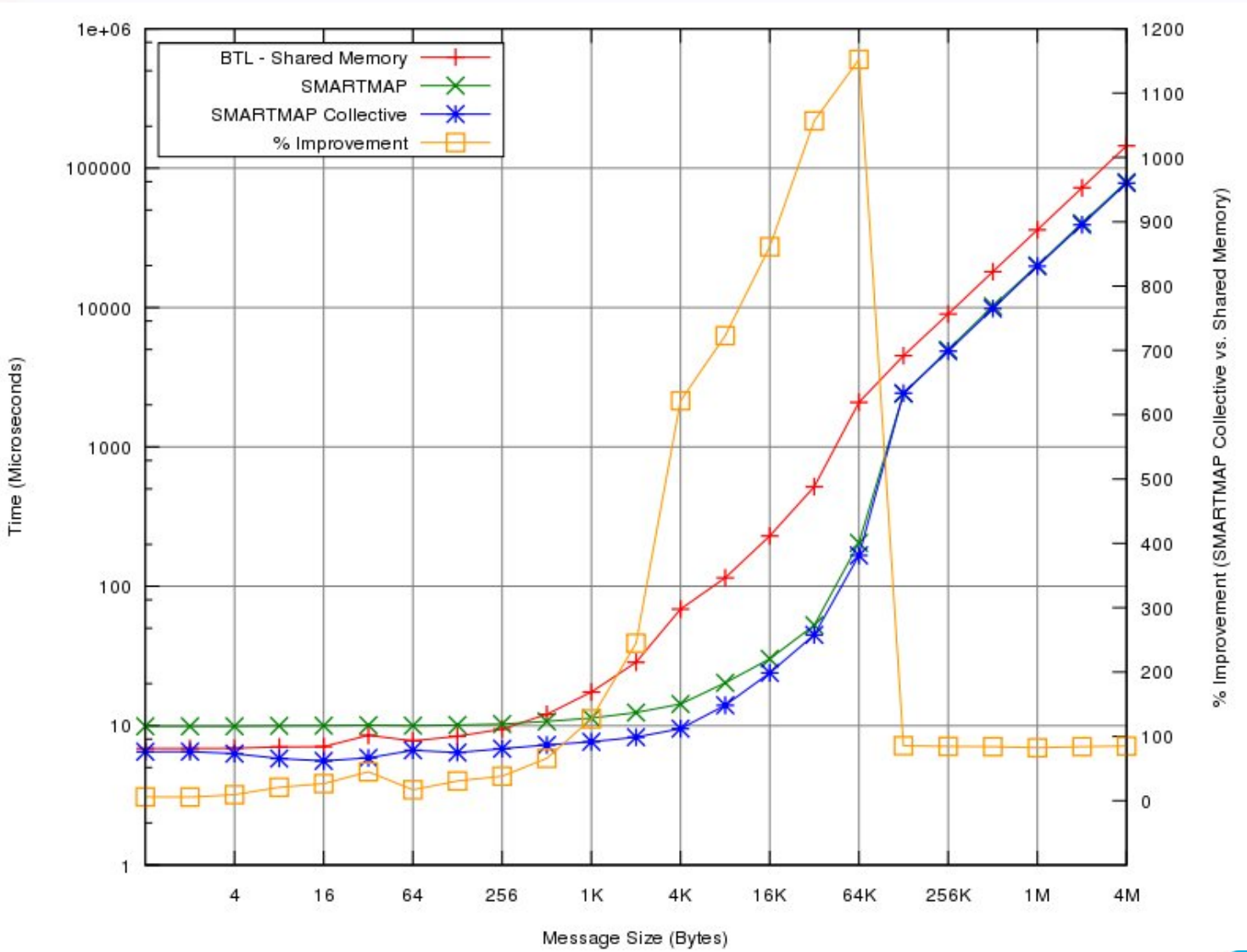
MPI Barrier


mtl-smap
btl-sm
smap-coll
mpich2-cle
Time (microseconds)
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
2
4
8
Processes

MPI Allreduce – 8 cores

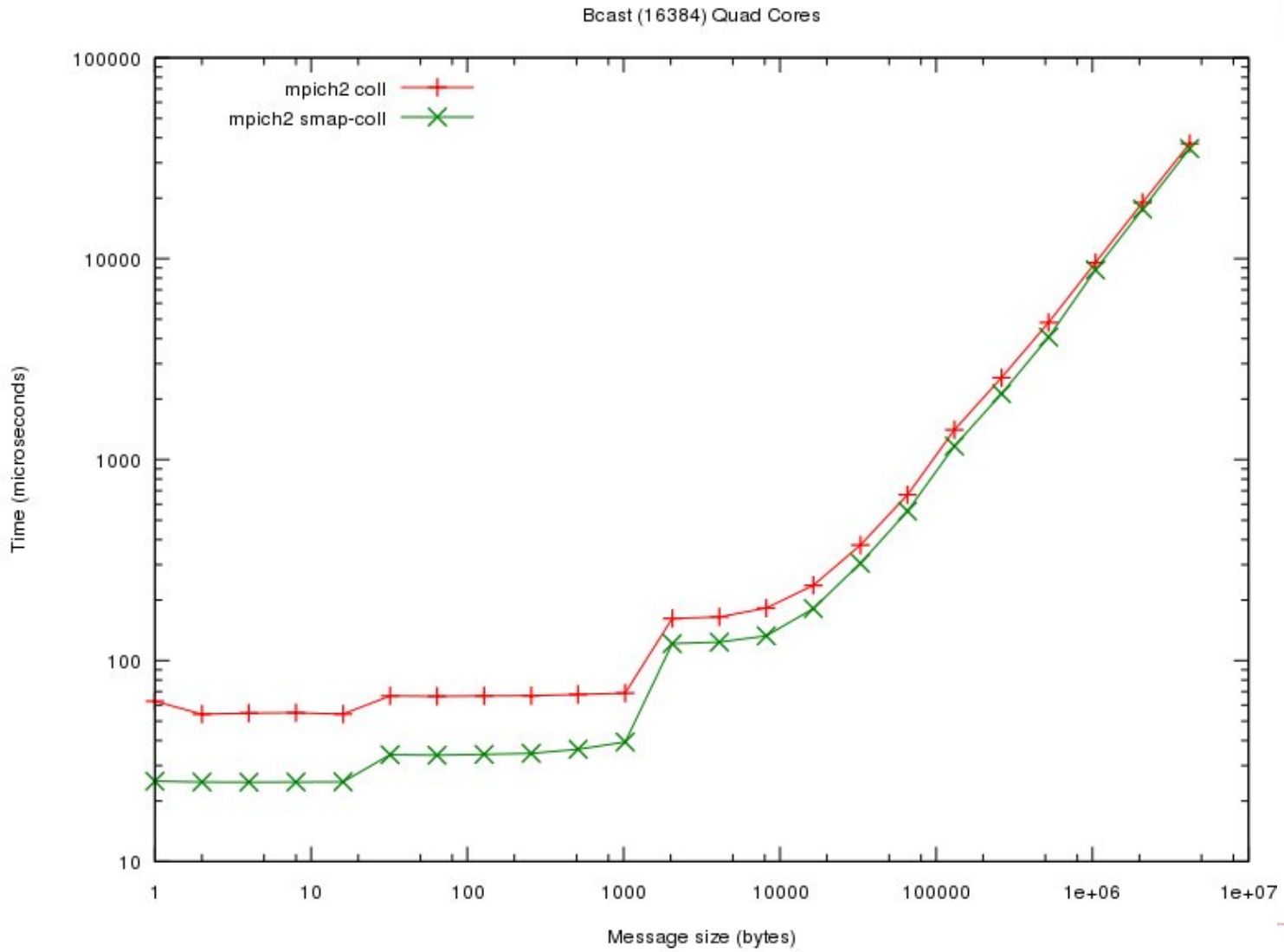


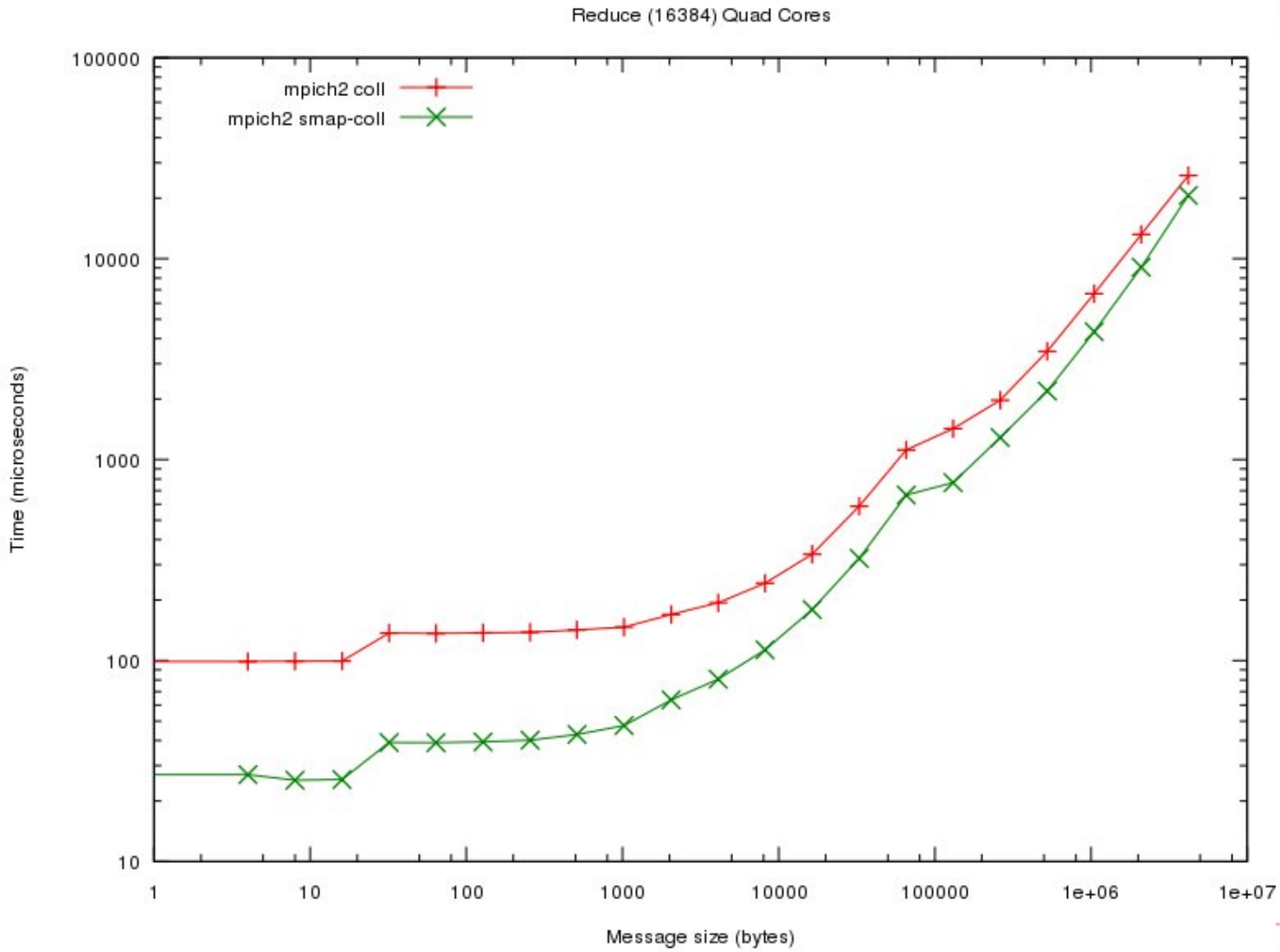
MPI Alltoall – 8 cores

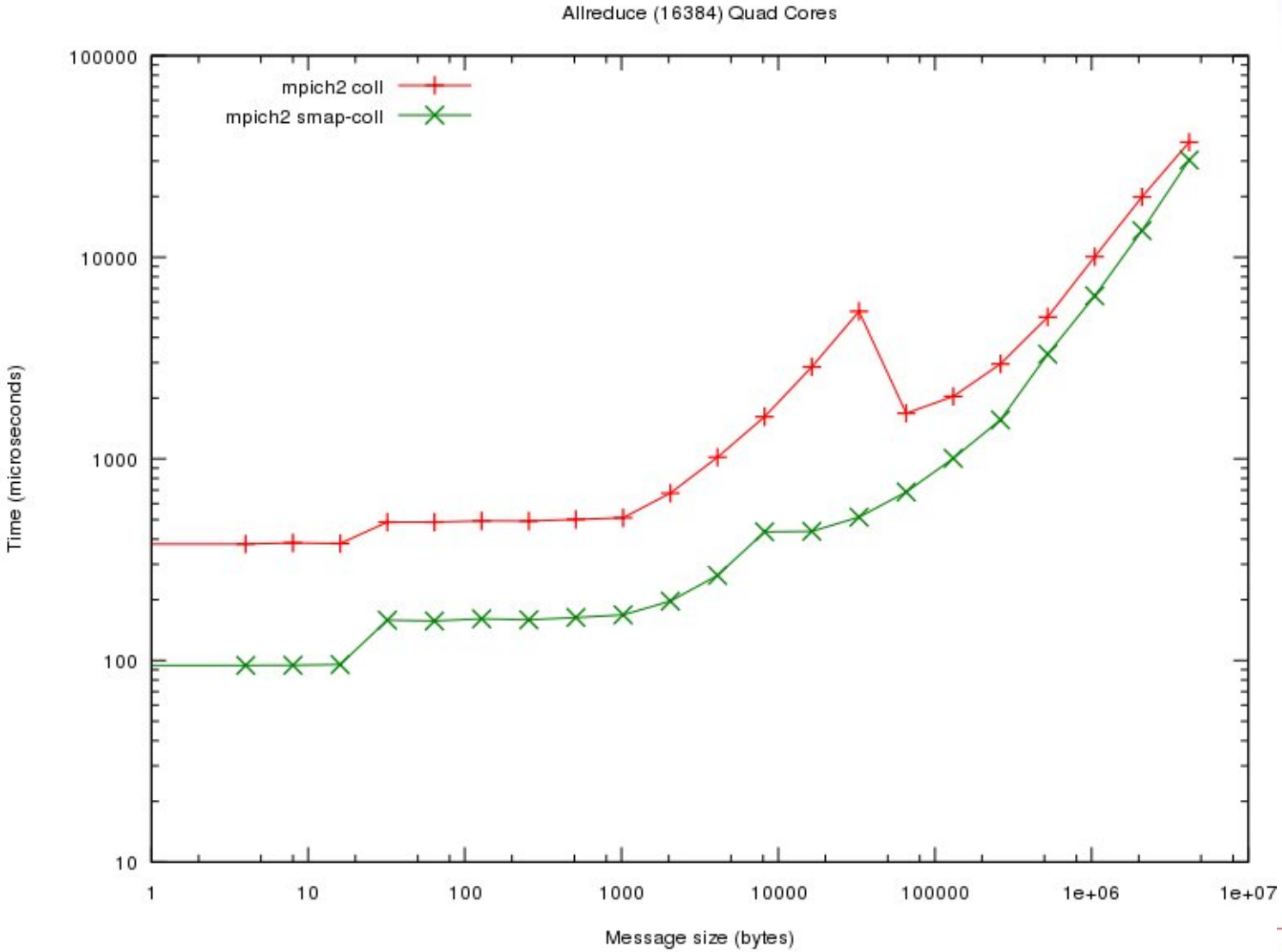


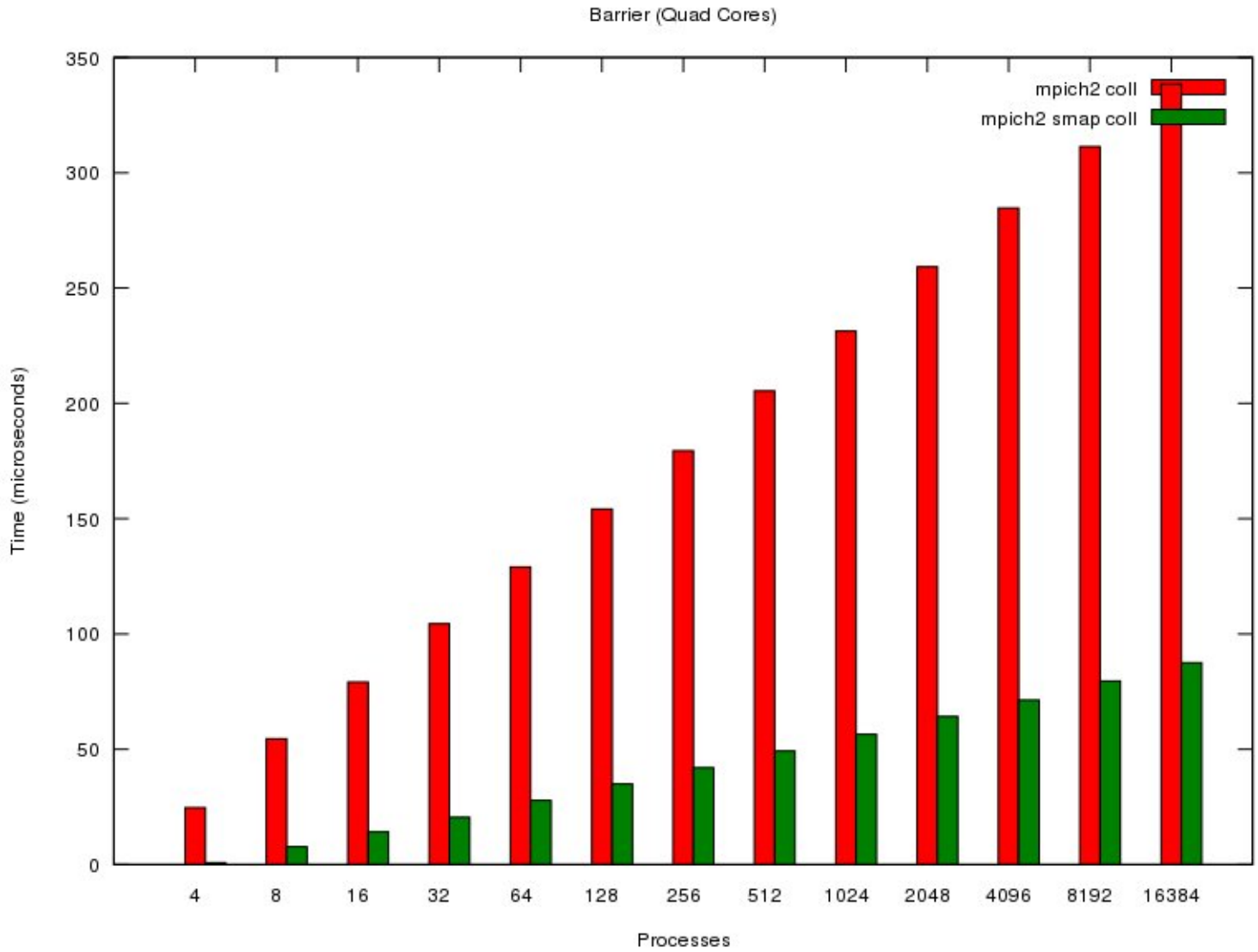
SMARTMAP for Cray MPICH2

- Cray's MPICH2 is the production MPI for Red Storm
  - Really old version of MPICH2
  - Cray added support for hierarchical Barrier, Bcast, Reduce, Allreduce
- Initial approach is to use SMARTMAP for these collectives
  - Reducing point-to-point latency with SMARTMAP unlikely to impact performance
    - Most codes dominated by longest latency
  - Optimizing collectives likely to have the most impact
- Results show hierarchical using SMAP versus non-hierarchical


Bcast (16384) Quad Cores
mpich2 coll
mpich2 smap-coll
Time (microseconds)
Message size (bytes)
100000
10000
1000
100
10
1
10
100
1000
10000
100000
1e+06
1e+07


Reduce (16384) Quad Cores
mpich2 coll
mpich2 smap-coll
Time (microseconds)
Message size (bytes)
100000
10000
1000
100
10
1
10
100
1000
10000
100000
1e+06
1e+07


Allreduce (16384) Quad Cores
mpich2 coll
mpich2 smap-coll
Time (microseconds)
Message size (bytes)
100000
10000
1000
100
10
1
10
100
1000
10000
100000
1e+06
1e+07

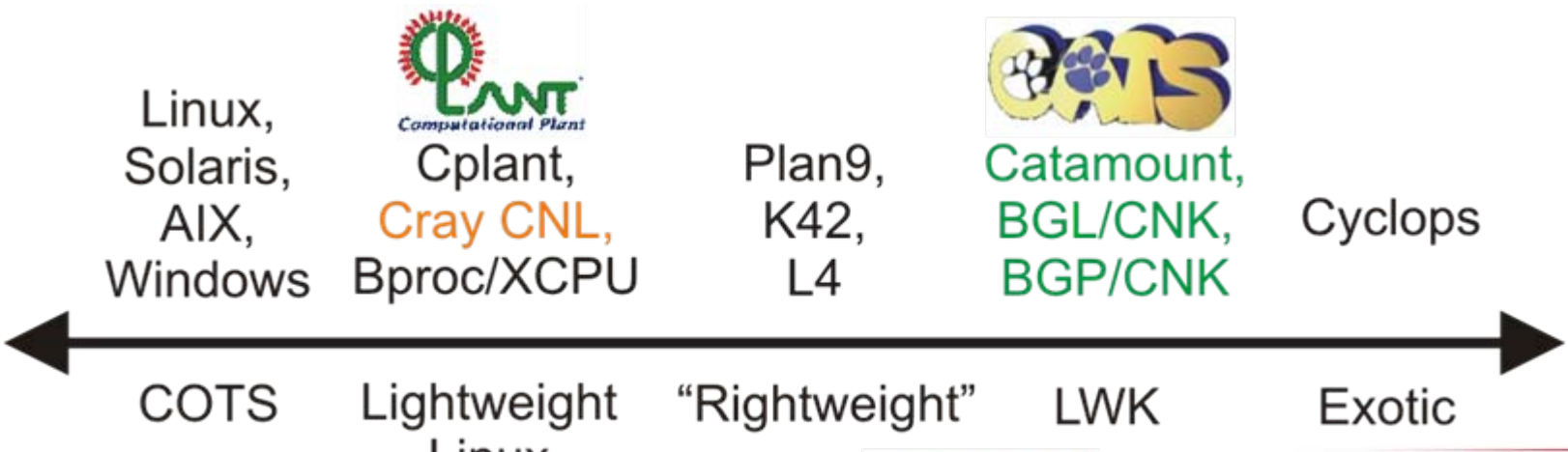


SMARTMAP Summary

- SMARTMAP provides significant performance improvements for intra-node MPI
  - Single-copy point-to-point messages
  - In-place collective operations
  - “Threaded” reduction operations
  - No serialization through OS or NIC
  - Simplified resource allocation
- Supports one-sided get/put semantics
- Can emulate POSIX-style shared memory regions

Project Kitten

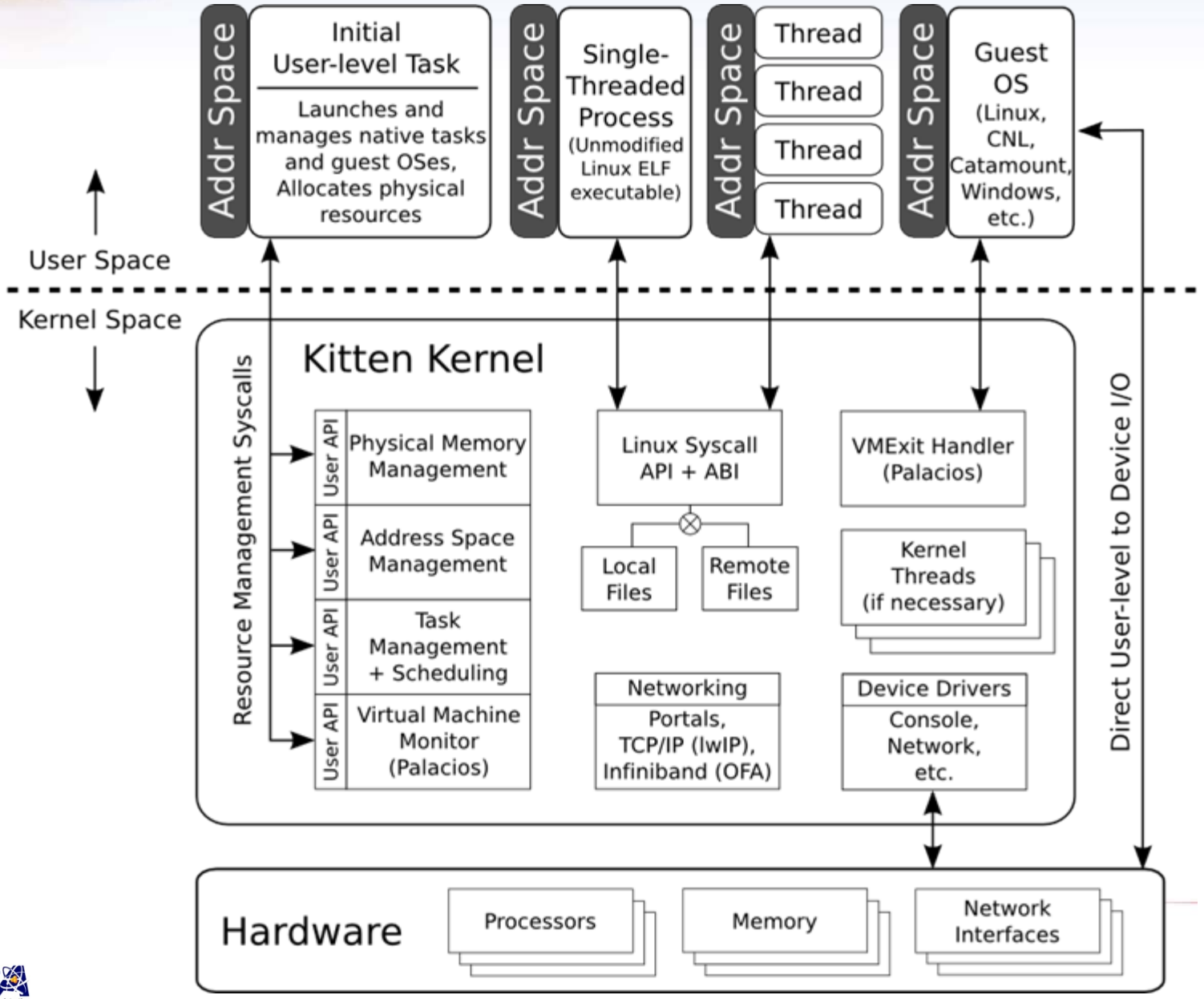
- **Creating modern open-source LWK platform**
  - Multi-core becoming MPP on a chip, requires innovation
  - Leverage hardware virtualization for flexibility
- **Retain scalability and determinism of Catamount**
- **Better match user and vendor expectations**
- **Available from http://software.sandia.gov/trac/kitten**



Leverage Linux and Open Source

- **Repurpose basic functionality from Linux Kernel**
  - Hardware bootstrap
  - Basic OS kernel primitives
- **Innovate in key areas**
  - Memory management (Catamount-like)
  - Network stack
  - SMARTMAP
  - Fully tick-less operation, but short duration OS work
- **Aim for drop-in replacement for CNL**
- **Open platform more attractive to collaborators**
  - Collaborating with Northwestern Univ. and Univ. New Mexico on lightweight virtualization for HPC, http://v3vee.org/
  - Potential for wider impact

Kitten Architecture



Current Status

- Initial release (December 2008)
  - Single node, multi-core
  - Available from http://software.sandia.gov/trac/kitten
- Development trunk
  - Support for Glibc NPTL and GCC OpenMP via Linux ABI compatible clone(), futex(), ...
  - Palacios virtual machine monitor support (planning parallel Kitten and Palacios releases for May 1)
  - Kernel threads and local files for device drivers
- Private development trees
  - Catamount user-level for multi-node (yod, PCT, Catamount Glibc port, Libsysio, etc.)
  - Ported Open Fabrics Alliance IB stack

Virtualization Support

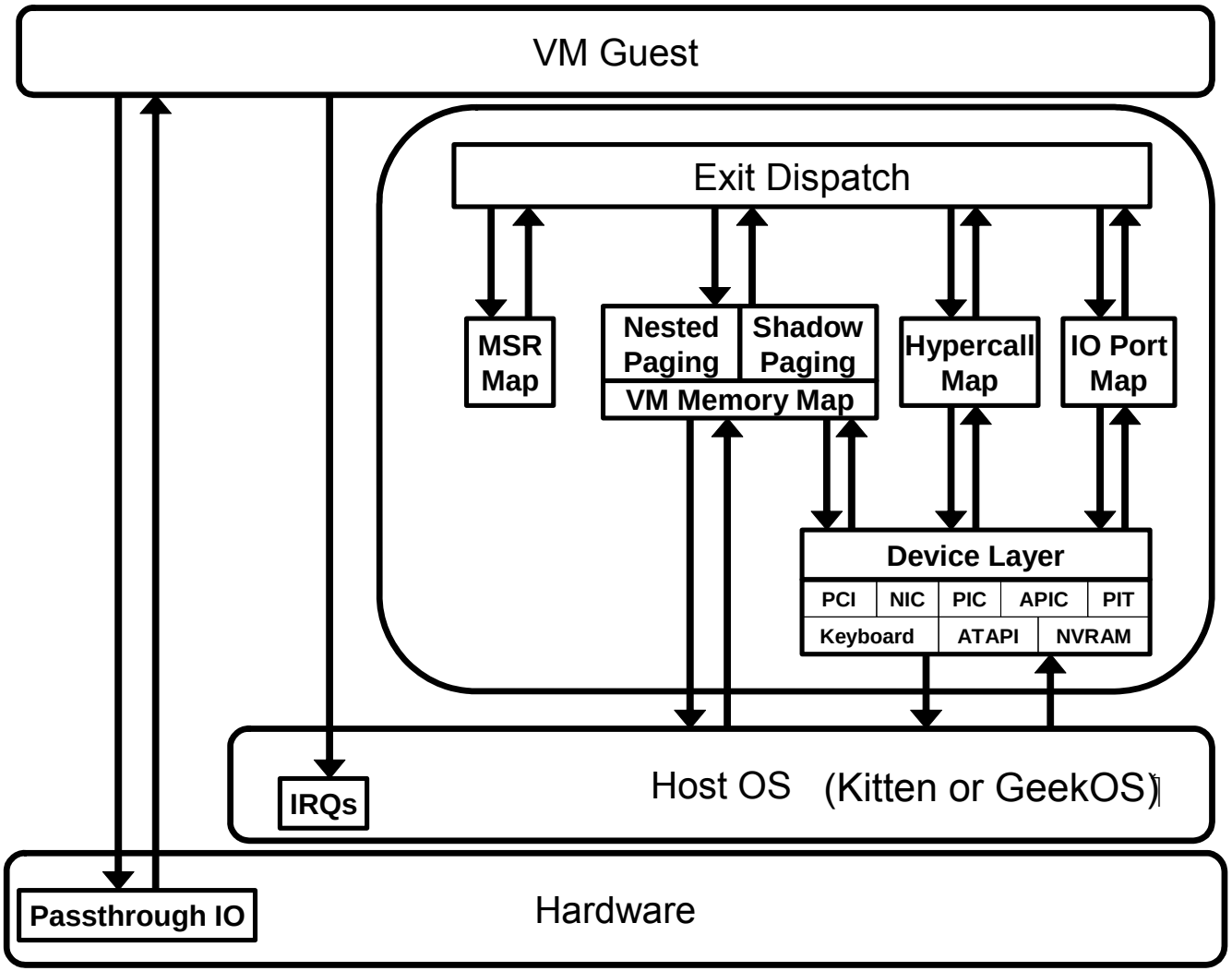
- Kitten optionally links with Palacios
  - Palacios developed by Jack Lange and Peter Dinda at Northwestern
  - Allows user-level Kitten applications to launch unmodified guest ISO images or disk images
  - Standard PC environment exposed to guest, even on Cray XT
  - Guests booted: Puppy Linux 3.0 (32-bit), Finnix 92.0 (64-bit), Compute Node Linux, Catamount
- "Lightweight Virtualization"
  - Physically contiguous memory allocated to guest
  - Pass-through devices (memory + interrupts)
  - Low noise, no timers or deferred work
  - Space-sharing rather than time-sharing

Motivations for Virtualization in HPC

- Provide full-featured OS functionality in a lightweight kernel
  - Custom tailor OS to application (ConfigOS, JeOS)
  - Possibly augment guest OS's capabilities
- Improve resiliency
  - Node migration, full-system checkpointing
  - Enhanced debug capabilities
- Dynamic assignment of compute node roles
  - Individual jobs determine I/O node to compute node balance
  - No rebooting required
- Run-time system replacement
  - Capability run-time poor match for high-throughput serial workloads

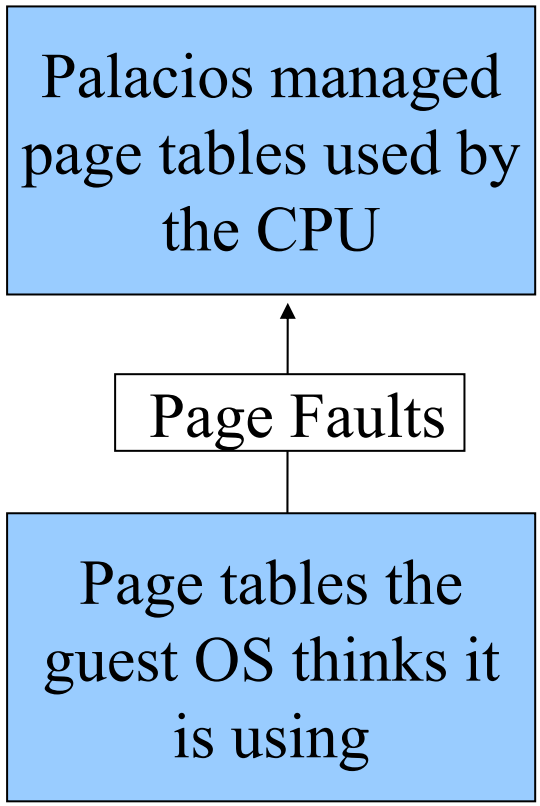
Palacios Architecture

(credit: Jack Lange, Northwestern University)

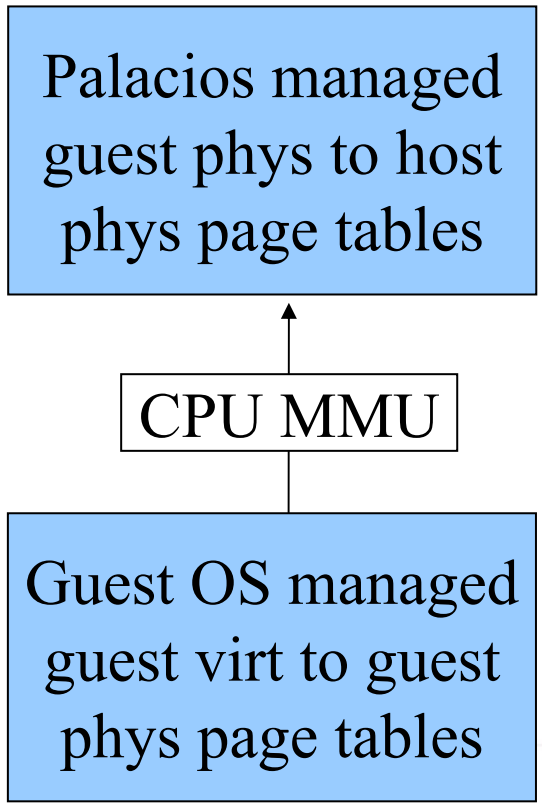


Shadow vs. Nested Paging: No Clear Winner

Shadow Paging, O(N) mem accesses per TLB miss



Nested Paging, O(N^2) mem accesses per TLB miss



Lines of Code in Kitten and Palacios

Component	Lines of Code	
	sloccount .	wc *.c *.h *.s
Kitten		
Kitten Core (C)	17,995	29,540
Kitten x86_64 Arch Code (C+Assembly)	14,604	22,190
Misc. Contrib Code (Kbuild + lwIP)	27,973	39,593
Palacios Glue Module (C)	286	455
Total	60,858	91,778
Palacios		
Palacios Core (C+Assembly)	15,084	24,710
Palacios Virtual Devices (C)	8,708	13,406
XED Interface (C+Assembly)	4,320	7,712
Total	28,112	45,828
Grand Total	88,970	137,606

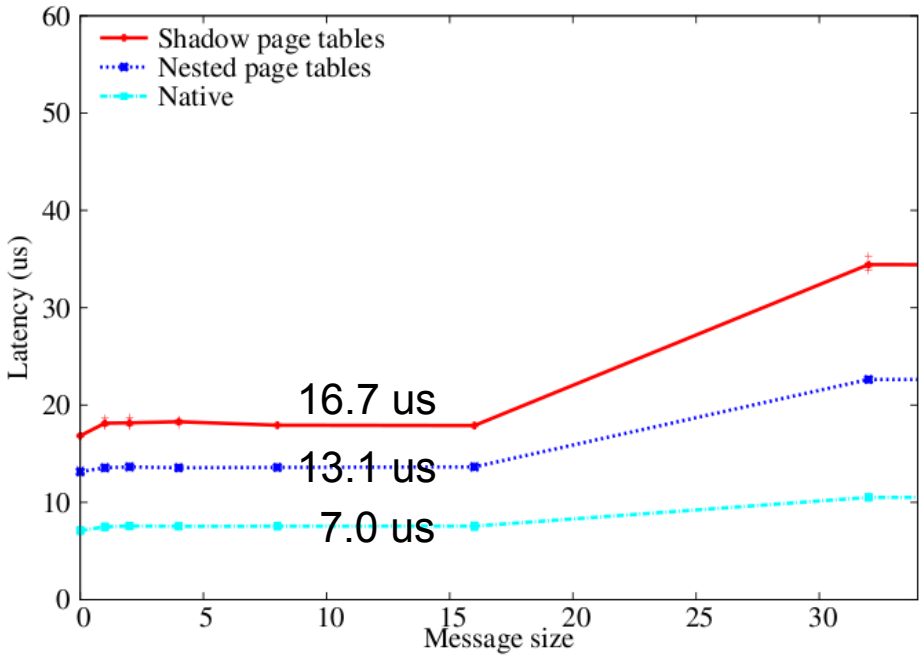
Kitten+Palacios on Cray XT

- Kitten boots as drop-in replacement for CNL
  - Kitten kernel vmlwk.bin -> vmlinux
  - Kitten initial task ELF binary -> initramfs
  - Kernel command-line args passed via parameters file
- Guest OS ISO image embedded in Kitten initial task
  - Kitten boots, starts user-level initial task, initial task “boots” the embedded guest OS
  - Both CNL and Catamount ported to the standard PC environment that Palacios exposes
- SeaStar direct-mapped through to guest
  - SeaStar 2 MB device window direct mapped to guest physical memory
  - SeaStar interrupts delivered to Kitten, Kitten forwards to Palacios, Palacios injects into guest

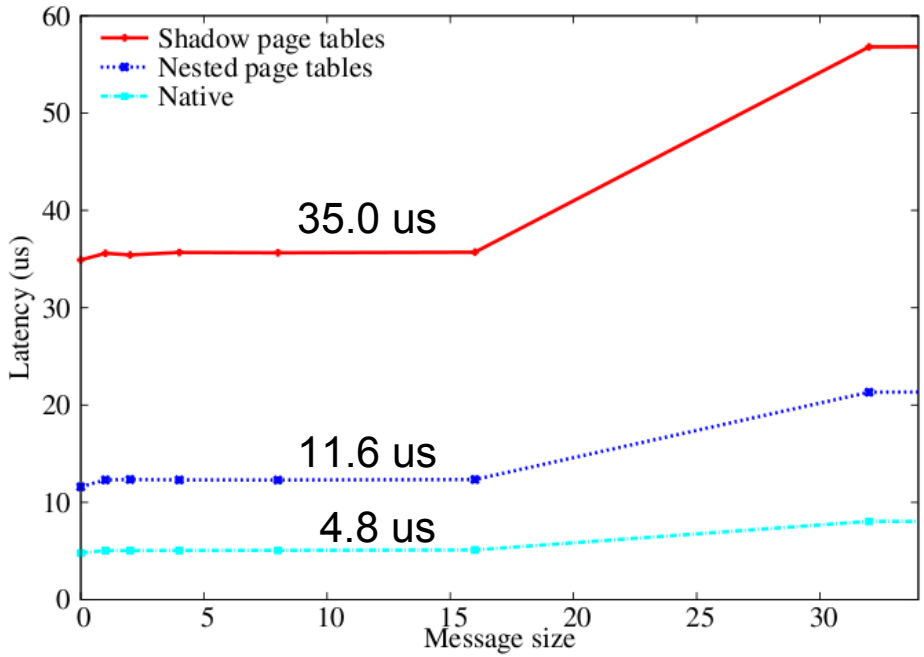
Native vs. Guest
CNL and Catamount Tests

- Testing performed on rsqual XT4 system at Sandia
  - Single cabinet, 48 2.2 GHz quad-core nodes
  - Developers have reboot capability
- Benchmarks:
  - Intel Messaging Benchmarks (IMB, formerly Pallas)
  - HPCCG “Mini-application”
    - Sparse CG solver
    - 100 x 100 x 100 problem, ~400 MB per node
  - CTH Application
    - Shock physics, important Sandia application
    - Shaped charge test problem (no AMR)
    - Weakly scaled

IMB PingPong Latency: Nested Paging has Lowest Overhead



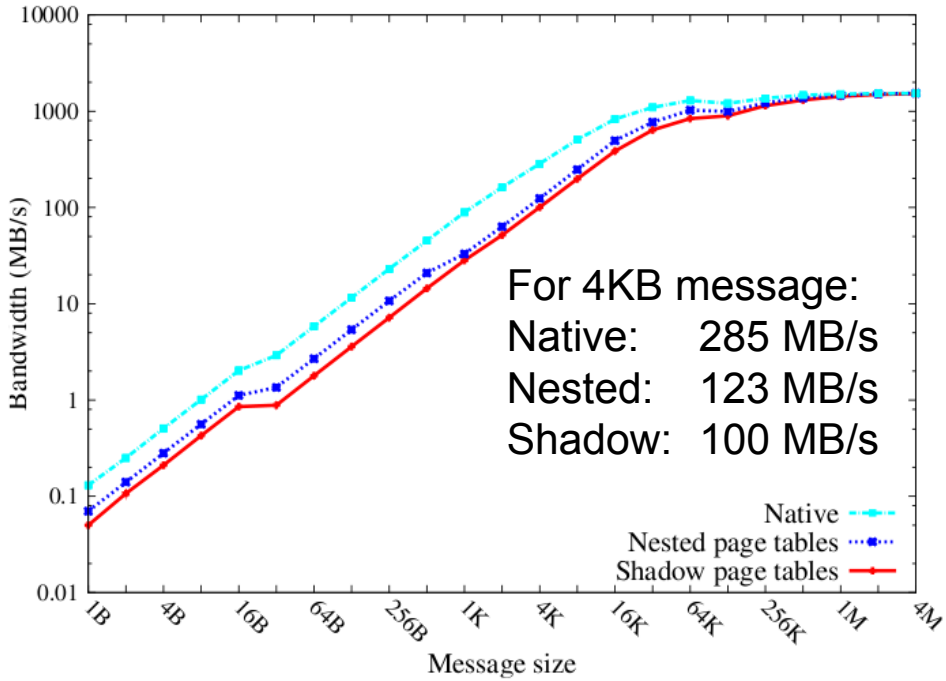
Compute Node Linux



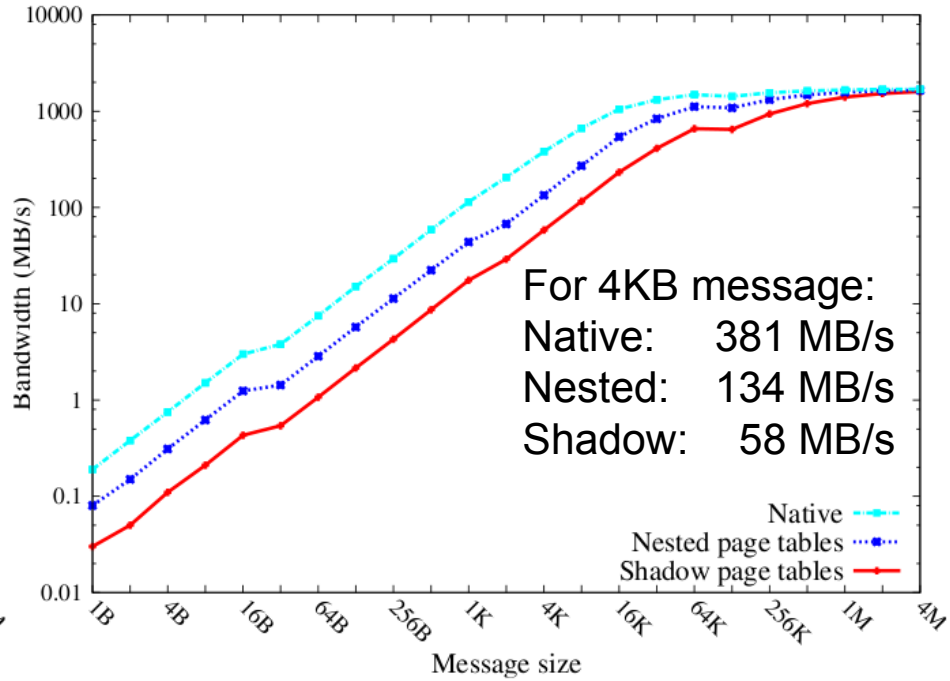
Catamount

Still investigating cause of poor performance of shadow paging on Catamount. Likely due to overhead/bug in emulating guest 2 MB pages for pass-through memory-mapped devices.

IMB PingPong Bandwidth: All Cases Converge to Same Peak Bandwidth

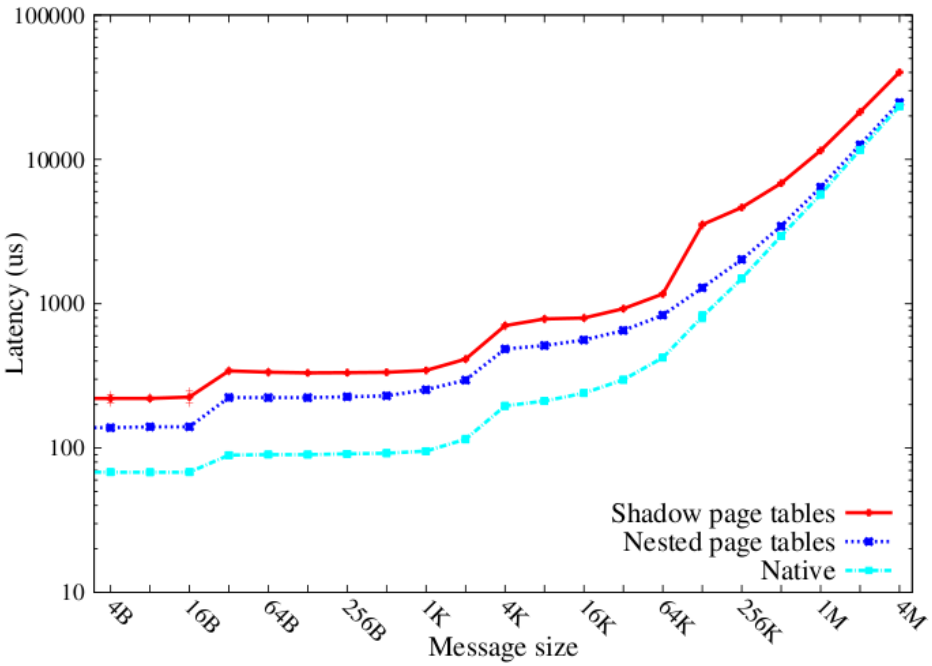


Compute Node Linux

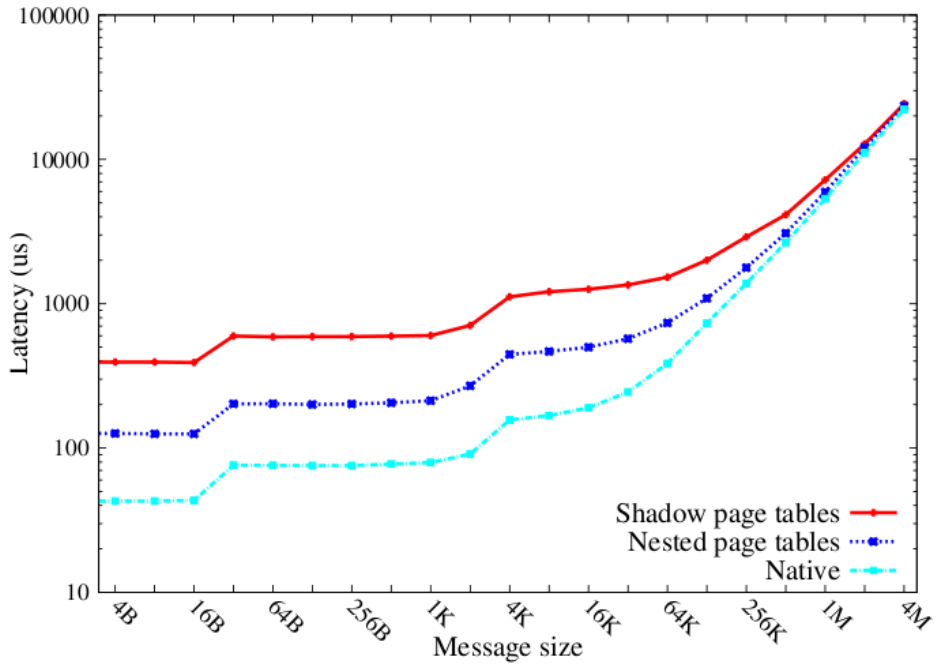


Catamount

48-Node IMB Allreduce Latency: Nested Paging Wins, Most Converge at Large Message Sizes

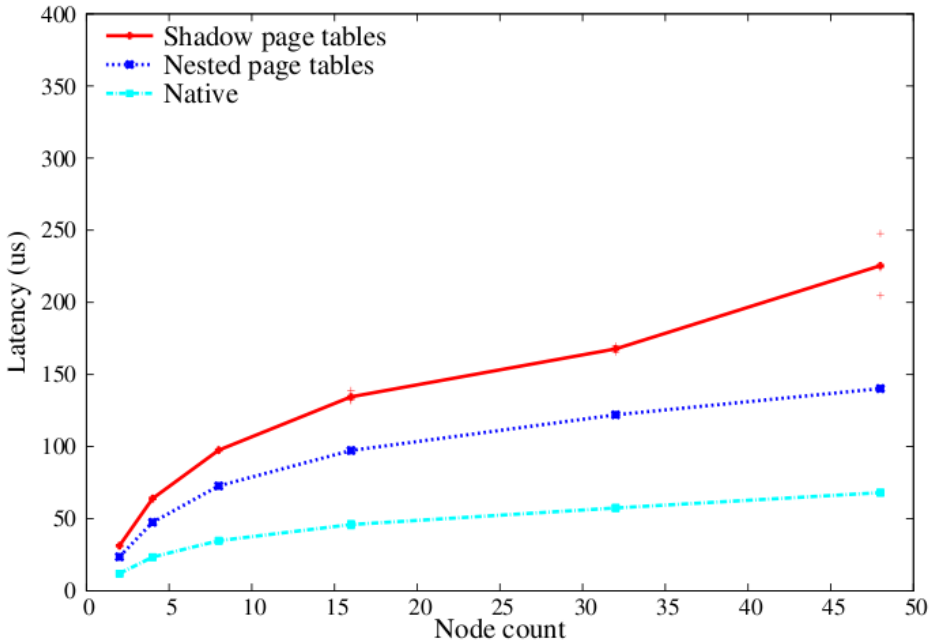


Compute Node Linux

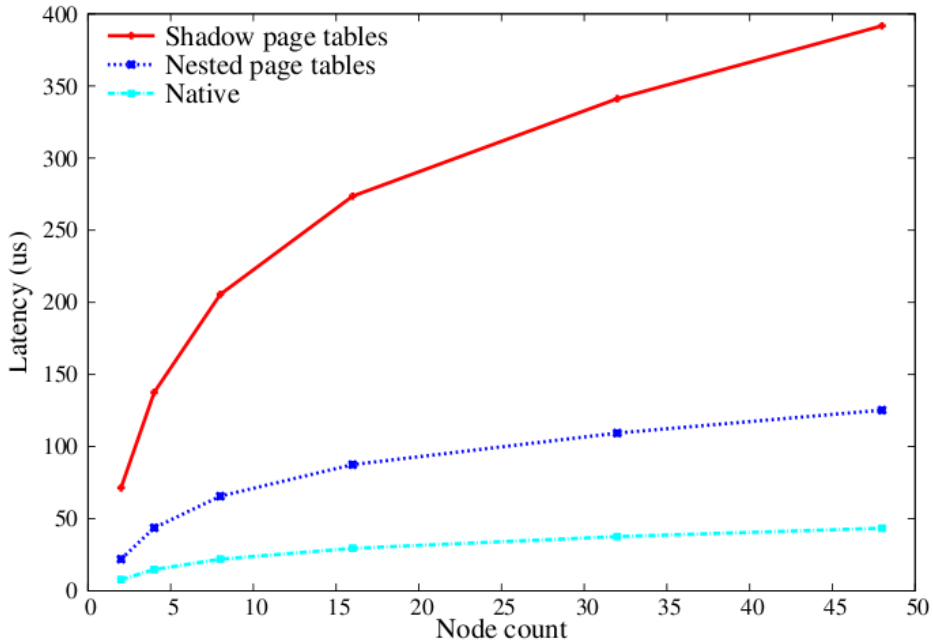


Catamount

16-byte IMB Allreduce Scaling: Native and Nested Paging Scale Similarly



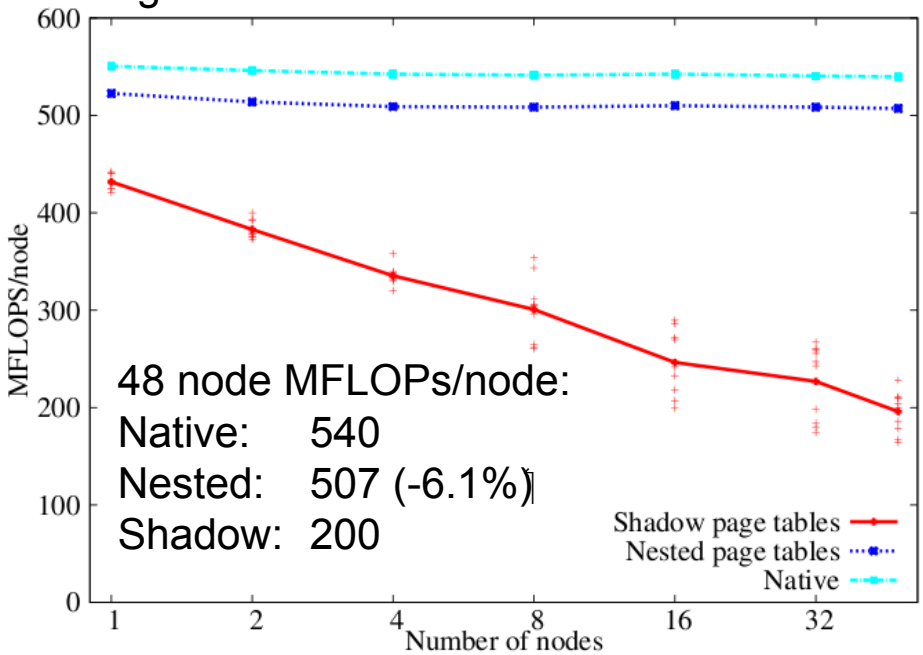
Compute Node Linux

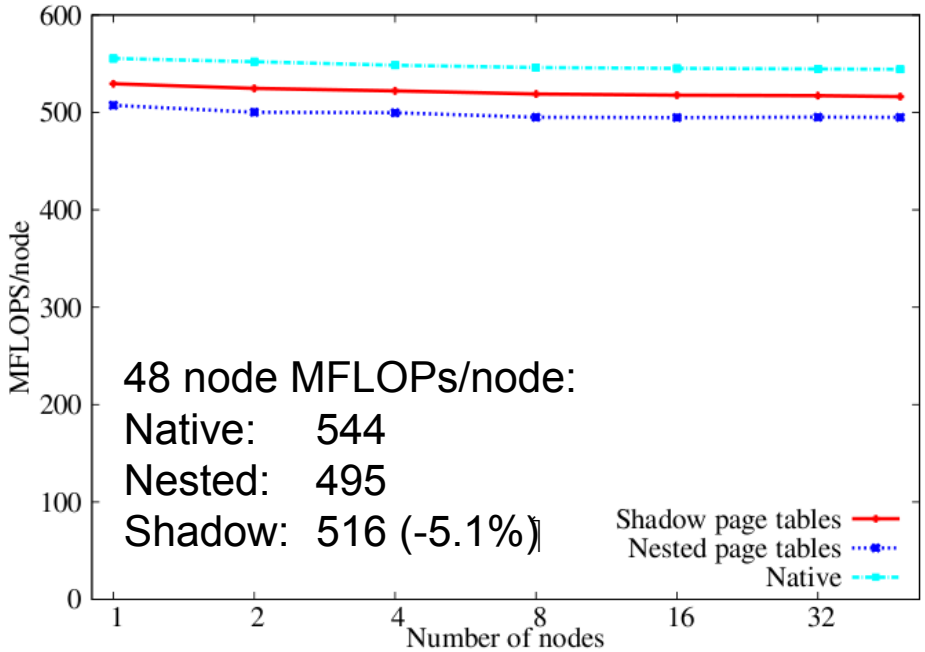


Catamount

HPCCG Scaling:
5-6% Virtualization Overhead
Shadow faster than Nested on Catamount

Higher is Better





Compute Node Linux

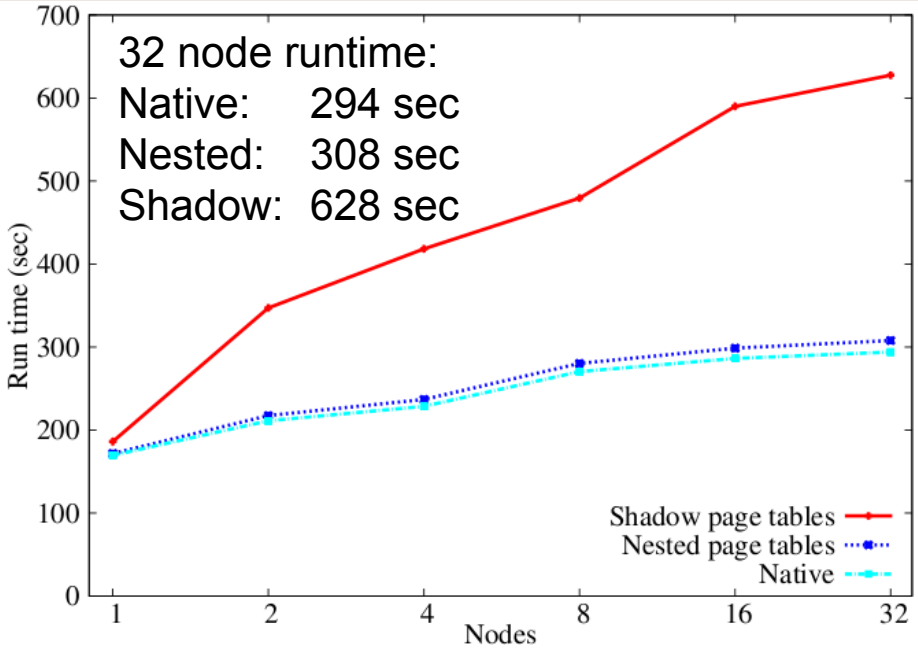
Catamount

Poor performance of shadow paging on CNL due to context switching. Could be avoided by adding page table caching to Palacios.

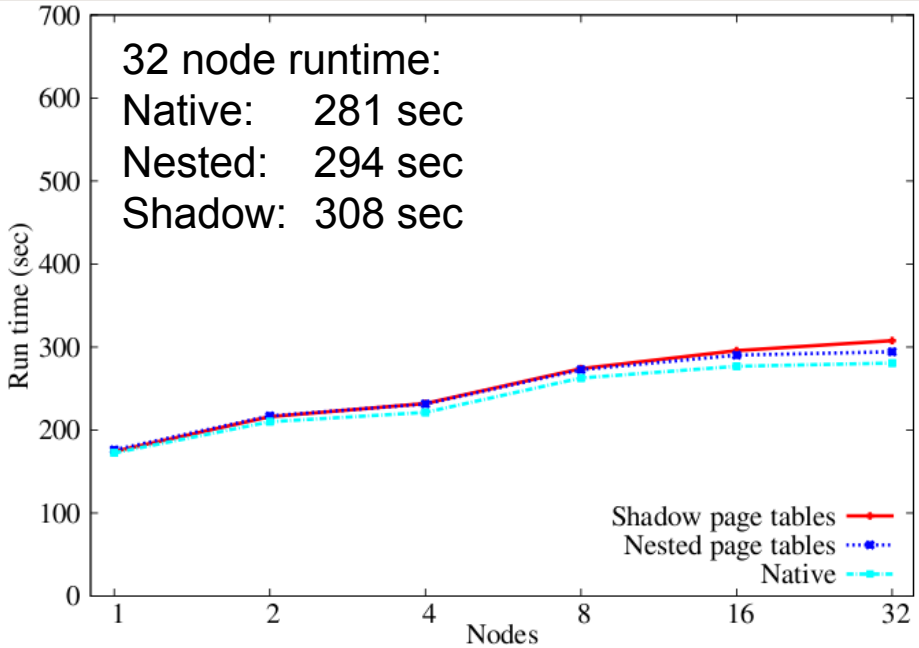
Catamount is essentially doing no context switching, benefiting shadow paging (2n vs. n^2 page table depth issue discussed earlier)

CTH Scaling:
< 5% Virtualization Overhead
Nested faster than Shadow on Catamount

Lower is Better



Compute Node Linux



Catamount

Poor performance of shadow paging on CNL due to context switching.
Could be avoided by adding page table caching to Palacios.

Kitten Summary

- Kitten LWK is in active development
  - Runs on Cray XT and standard PC hardware
  - Guest OS support when combined with Palacios
  - Available now, open-source
- Virtualization experiments on Cray XT indicate ~5% performance overhead for CTH application
  - Would like to do larger scale testing
  - Accelerated portals may further reduce overhead

Acknowledgments

- Catamount/SMARTMAP
  - John VanDyke, SNL
  - Tramm Hudson, OS Research
  - Kevin Pedretti, SNL
  - Kurt Ferreira, SNL
  - Sue Kelly, SNL
  - Jeff Crow, HP
- Kitten
  - Kevin Pedretti, SNL
  - Tramm Hudson, OS Research
  - Mike Levenhagen, SNL
  - Peter Dinda, Northwestern U.
  - Jack Lange, Northwestern U.
  - Patrick Bridges, U. New Mexico

Questions?