



---

# Panel: MPI for Multi-Core and Many-Core

**Ron Brightwell**

**Scalable Computing Systems Department  
Center for Computation, Computers, Information and Math  
Sandia National Laboratories**

**[rbrigh@sandia.gov](mailto:rbrigh@sandia.gov)**

**<http://www.sandia.gov/~rbrigh>**

**Next-Generation Scalable Applications: When MPI-Only Is Not Enough  
June 4, 2008**



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.





## **OS Limitations and MPI Exacerbate Memory Bandwidth Problem**

---

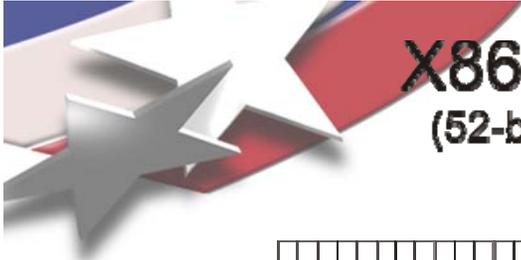
- **Multi-core processors stress memory bandwidth**
- **MPI compounds the problem**
  - **Intra-node MPI uses memory-to-memory copies**
  - **Most implementations use POSIX shared memory**
    - **Sender copies data in**
    - **Receiver copies data out**
  - **Alternative strategies use OS to re-map memory pages and copy between address spaces**
    - **Trapping and mapping takes time**
    - **Serialization through the OS creates bottleneck**



# Current Approach for Intra-Node MPI on Catamount Has Several Limitations

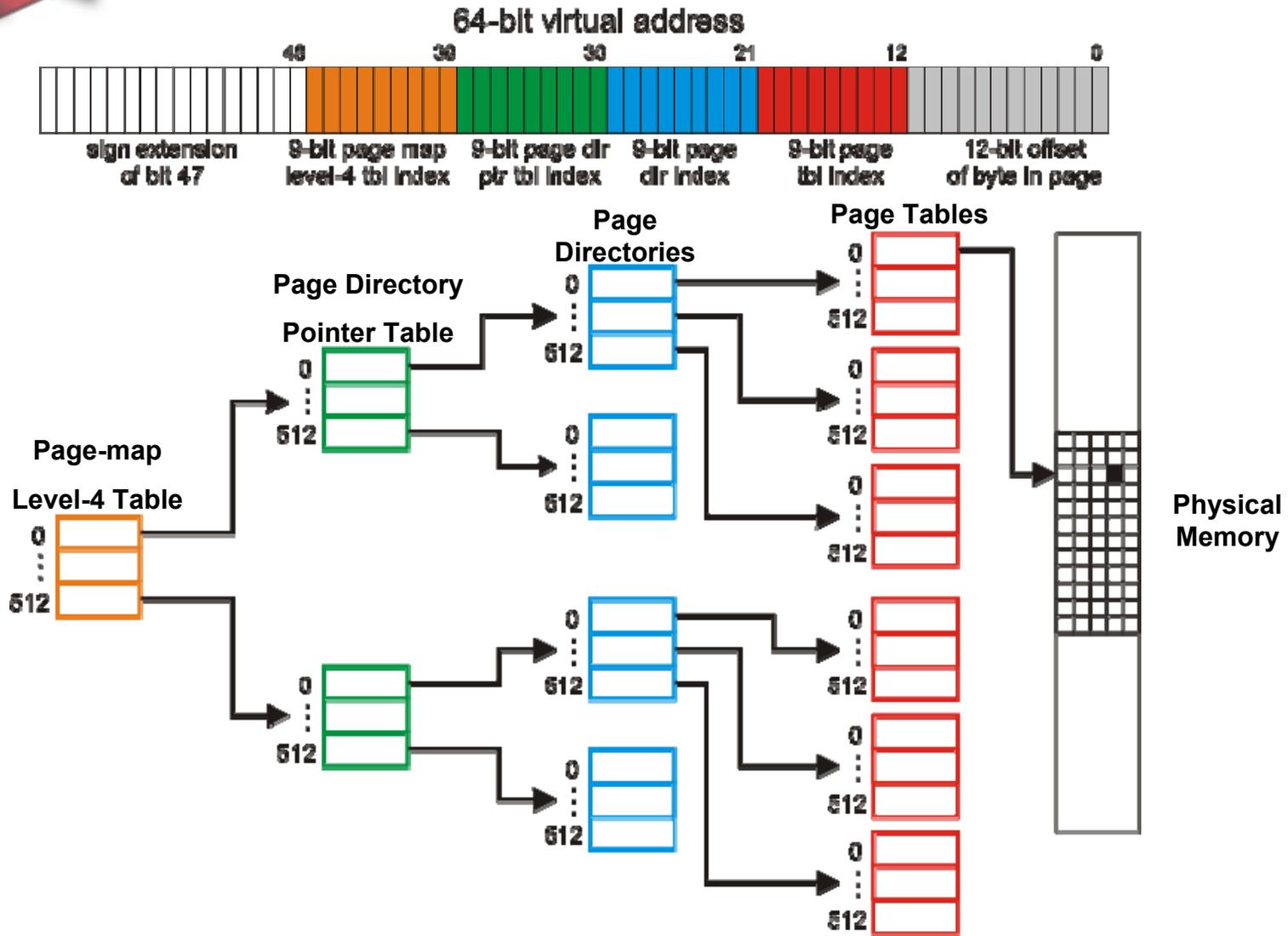
---

- **Uses Portals for all inter-process communication**
- **Interrupt driven implementation (aka GP)**
  - OS does memory copy between processes ( $\leq 512$  KB)
  - OS uses SeaStar DMA ( $> 512$  KB)
  - Serialization through OS
- **NIC-based implementation (aka AP)**
  - SeaStar does DMA between processes
  - Still need OS trap to initiate send
  - Serialization through OS and SeaStar
- **Both approaches create load imbalance**
  
- **Can the OS help?**



# X86-64 Long PAE Page Table Lookup

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

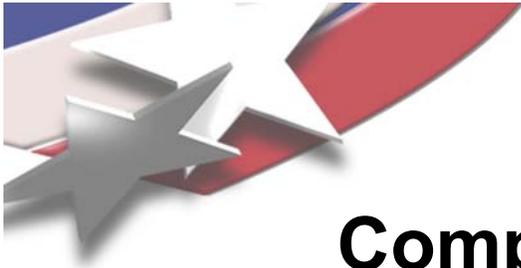




## SMARTMAP: Simple Mapping of Address Region Tables for Multi-core Aware Programming

---

- **Allows all of the processes on a multi-core processor to access each others' memory directly**
  - **User-space to user-space**
  - **No serialization through the OS or NIC**
  - **Access to remote address by flipping a few bits**
- **Each process still has a separate virtual address space**
  - **Everything is private and everything can be shared**
- **Leverages Catamount's physically contiguous memory model**
- **Allows MPI to eliminate *all* extraneous memory-to-memory copies on node**
  - **No copying for non-contiguous MPI datatypes**
  - **More efficient collective operations**
    - **Reductions can operate directly on user buffer**
- **Not just for MPI**



# Complexity 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;
}
```



# Making SHMEM Work with SMARTMAP

---

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

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

        void targetr = (void)remote_address(core, target);

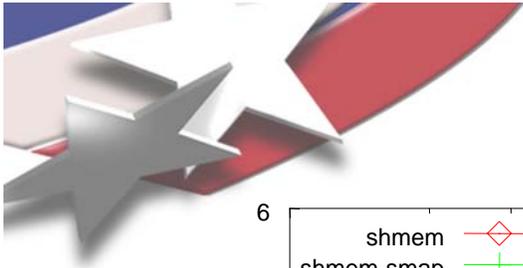
        memcpy(targetr, source, length);

    } else {

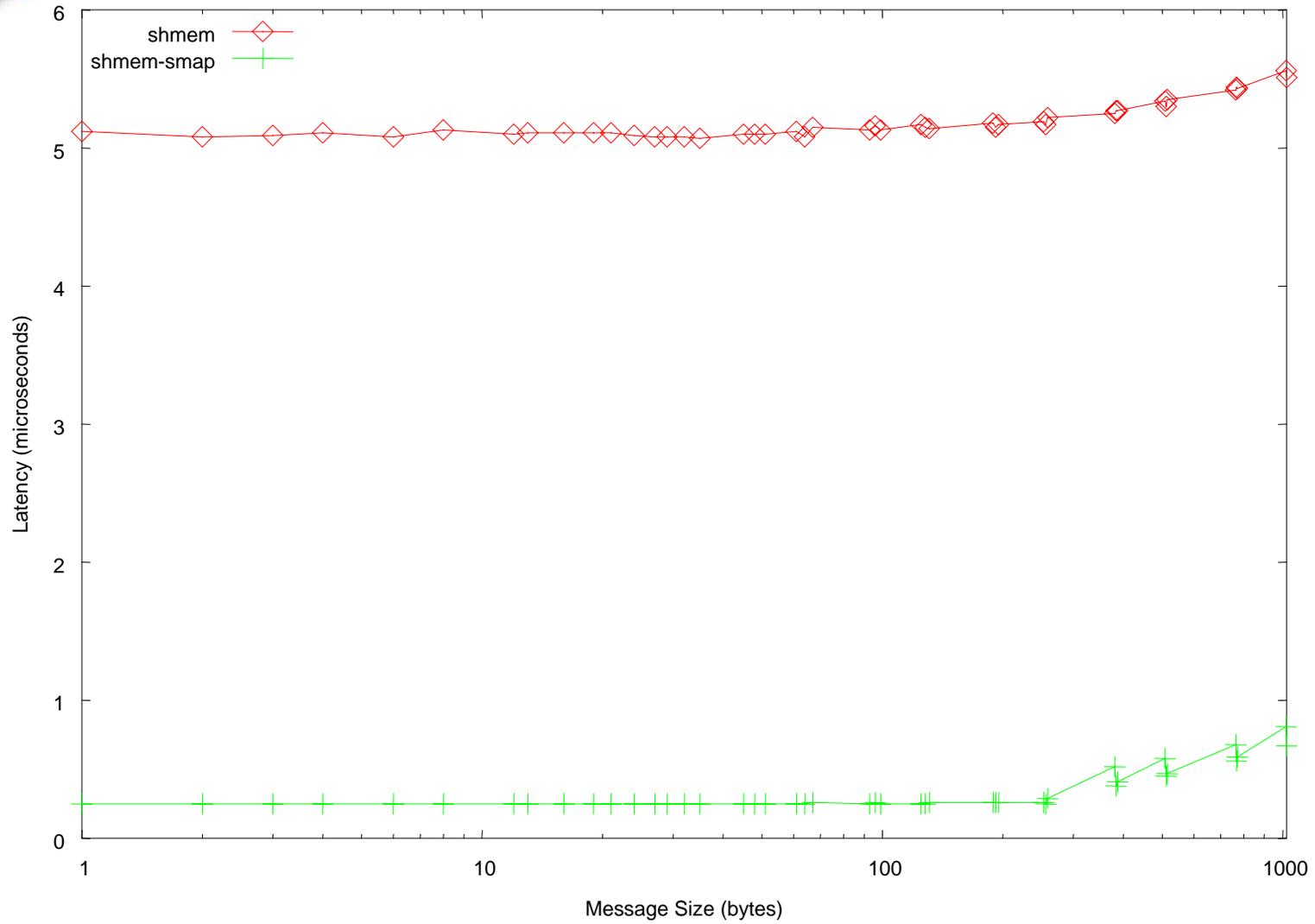
        pshmem_putmem(target, source, length, pe);

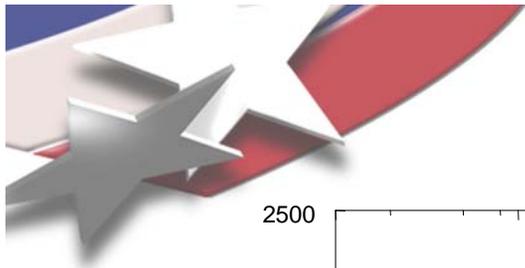
    }

}
```

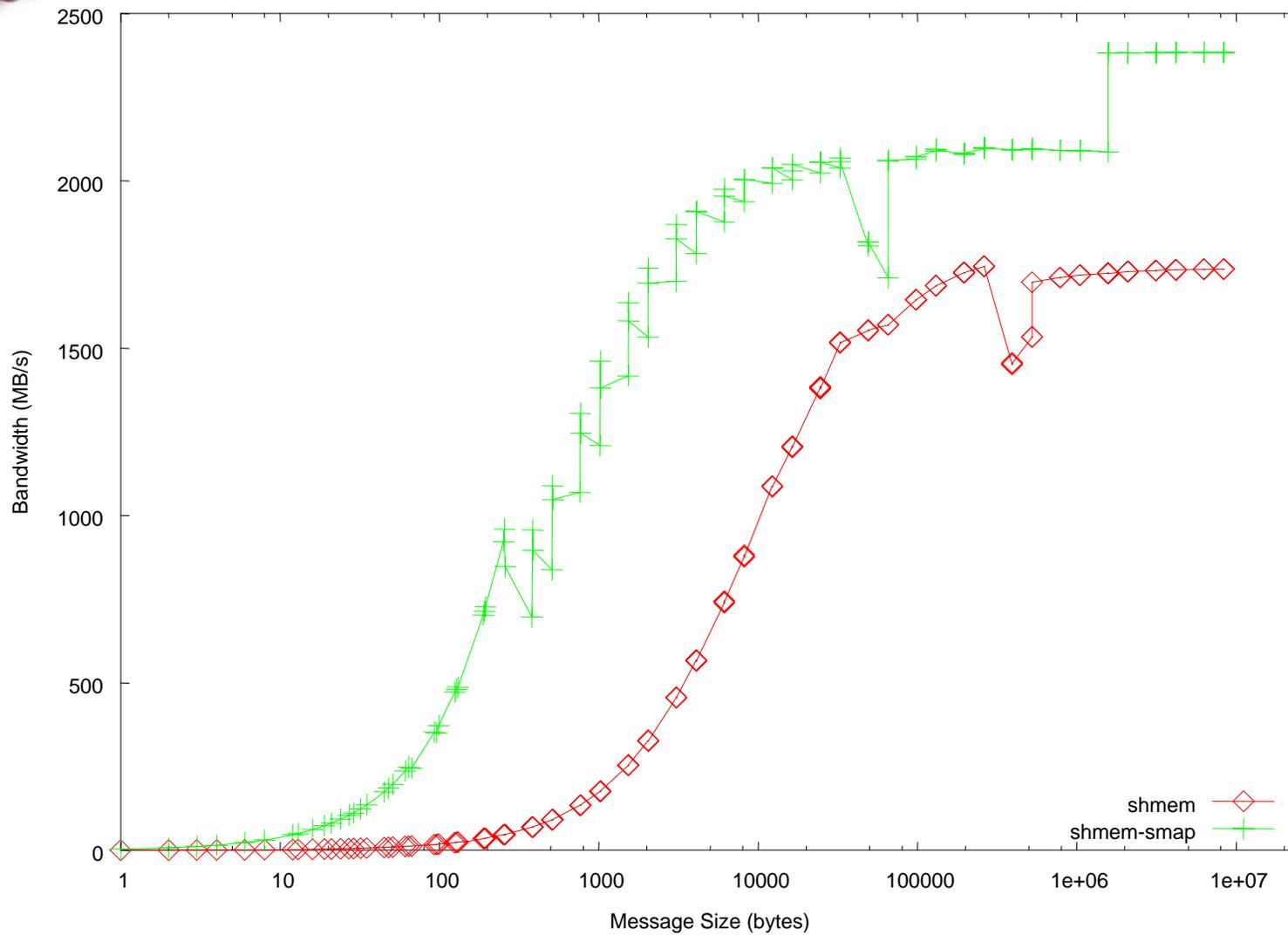


Ping-Pong Latency





Ping-Pong Bandwidth





# Open MPI for Portals

---

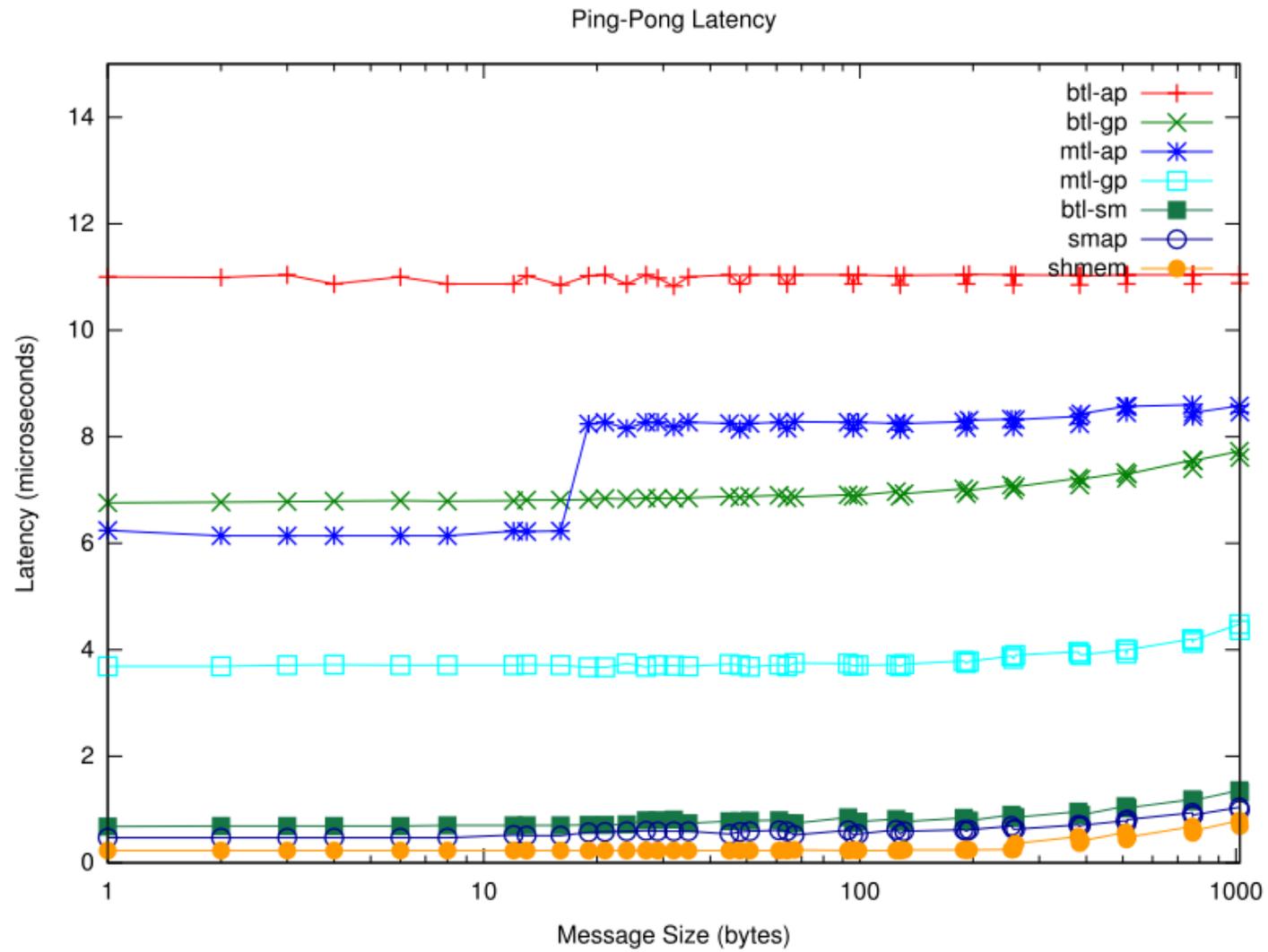
MPI	
Point-to-Point Management Layer (PML)	
Byte Transfer Layer (BTL)	Matching Transport Layer (MTL)
Portals	
Shared Memory	SeaStar

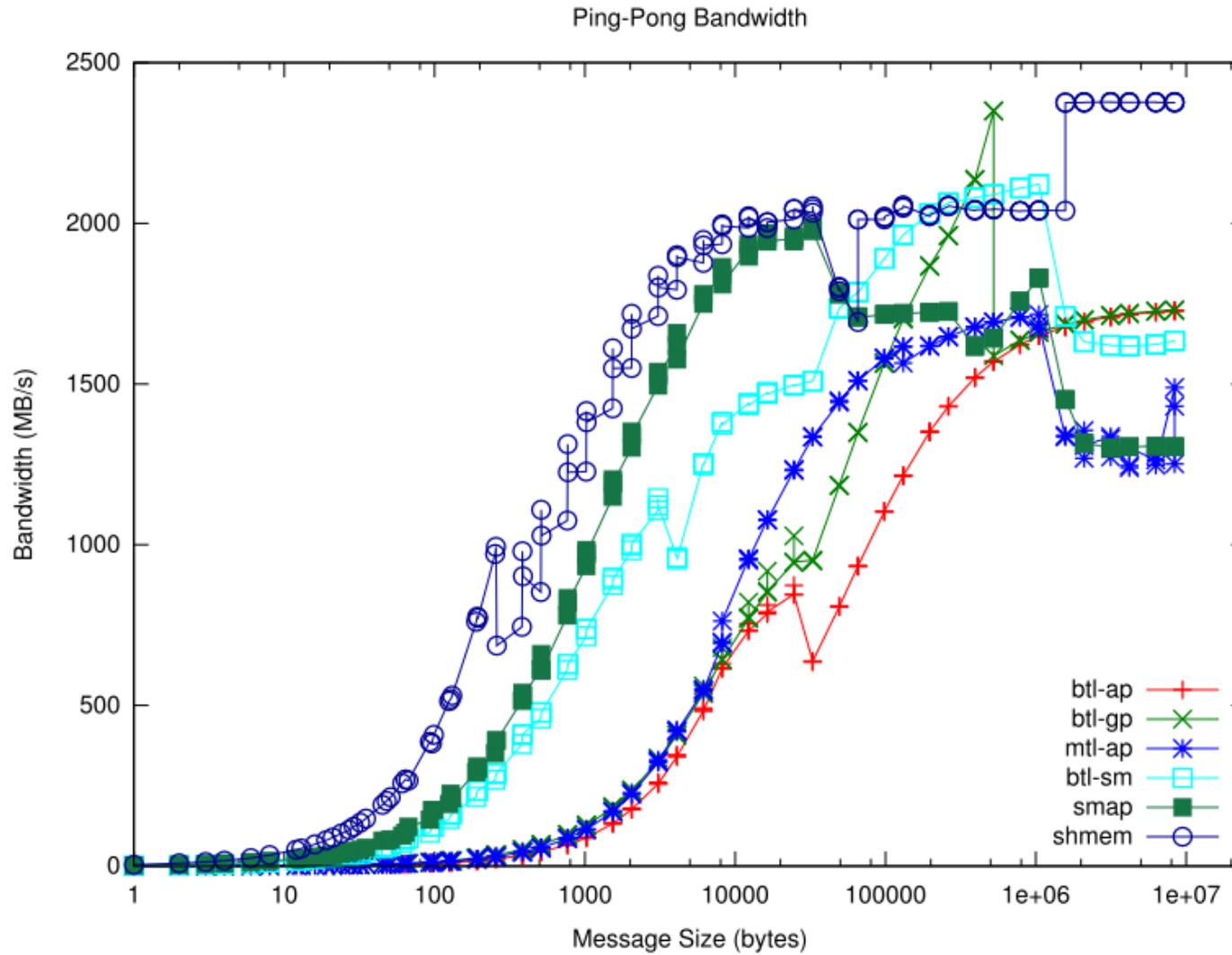


# Open MPI with SMARTMAP

---

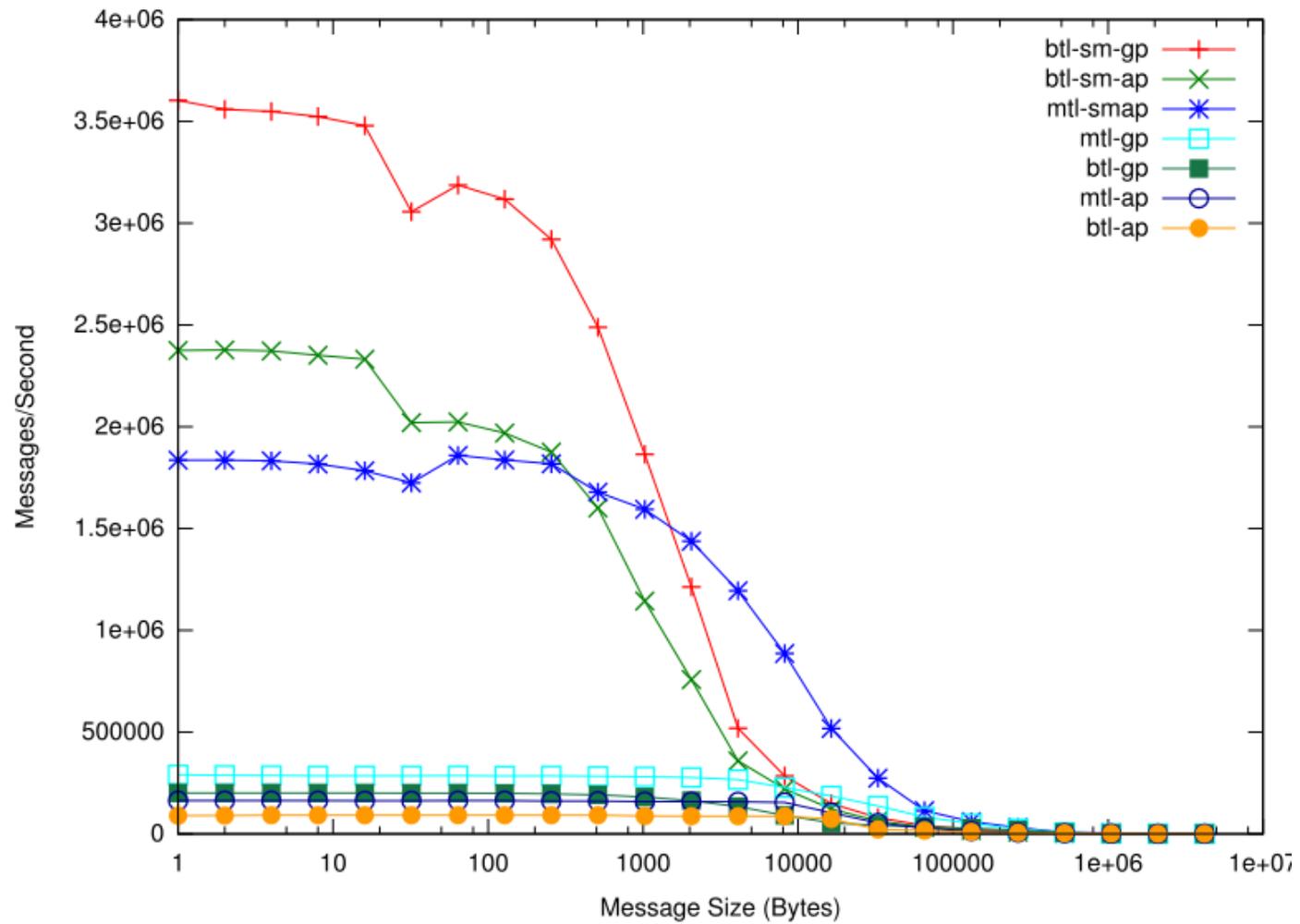
MPI			
Point-to-Point Management Layer (PML)			
Byte Transfer Layer (BTL)		Matching Transport Layer (MTL)	
Shared Memory	Portals		SMARTMAP
SMARTMAP	Shared Memory	SeaStar	





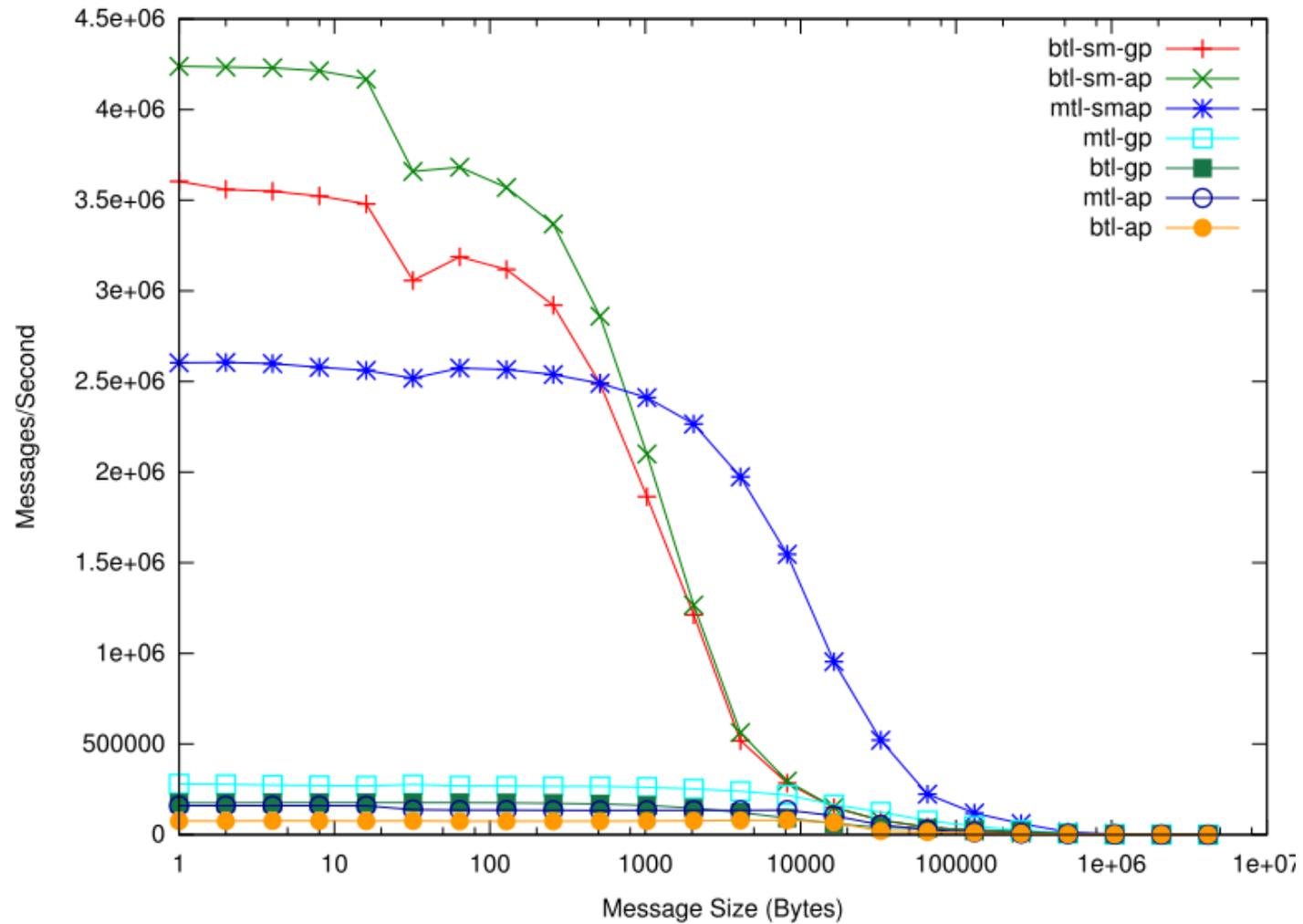


MPI Message Rate (2 processes)



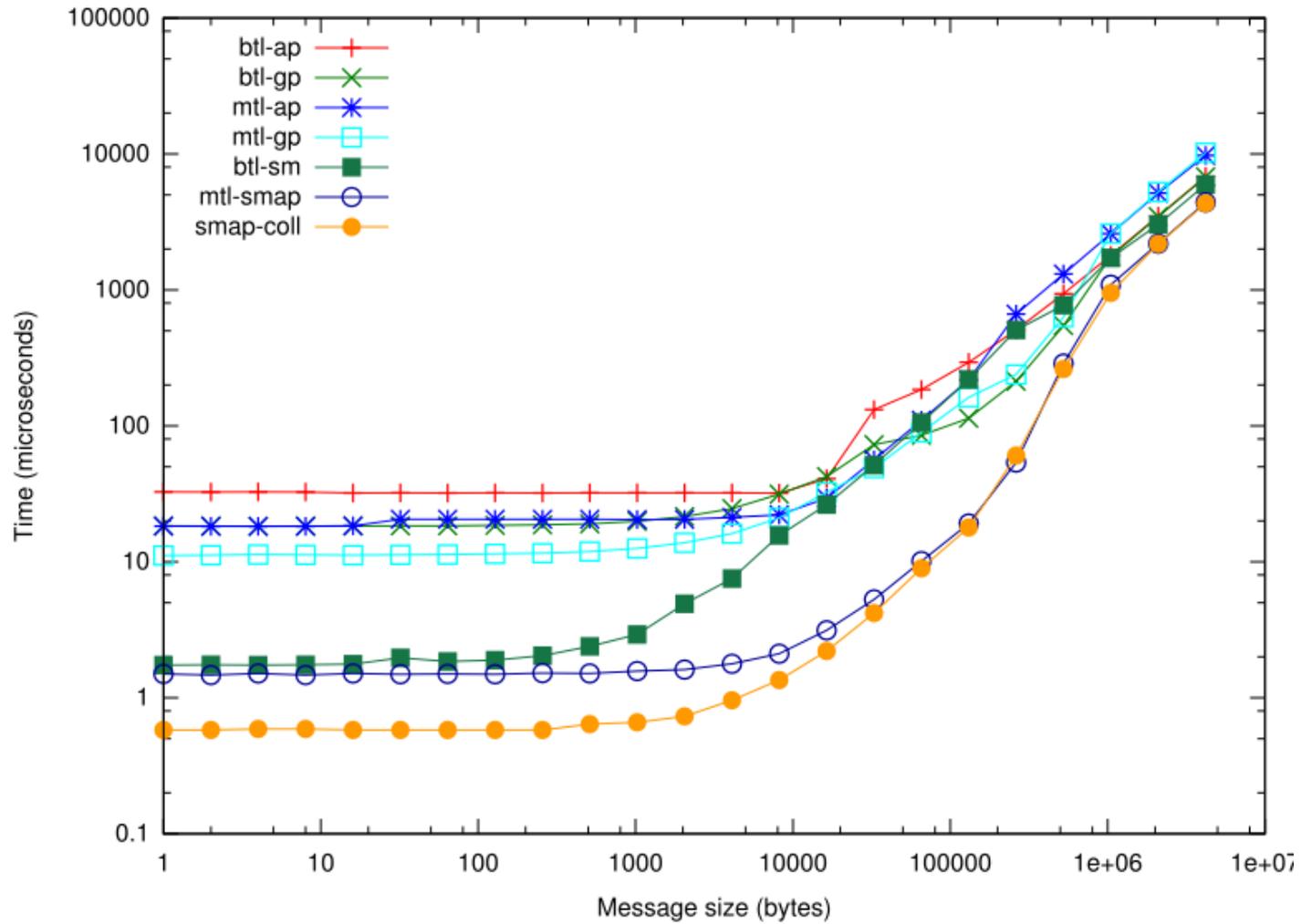


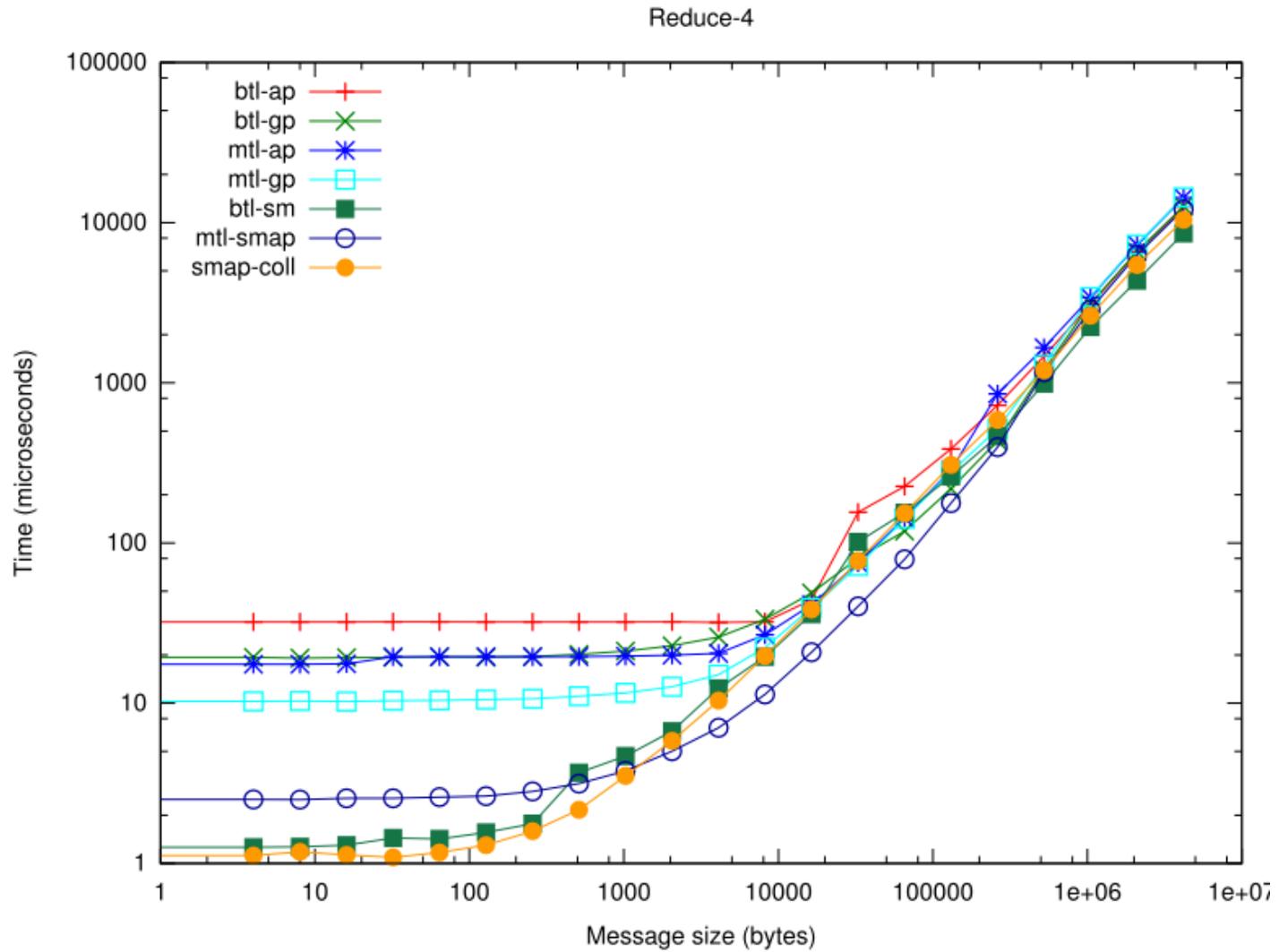
MPI Message Rate (4 Processes)





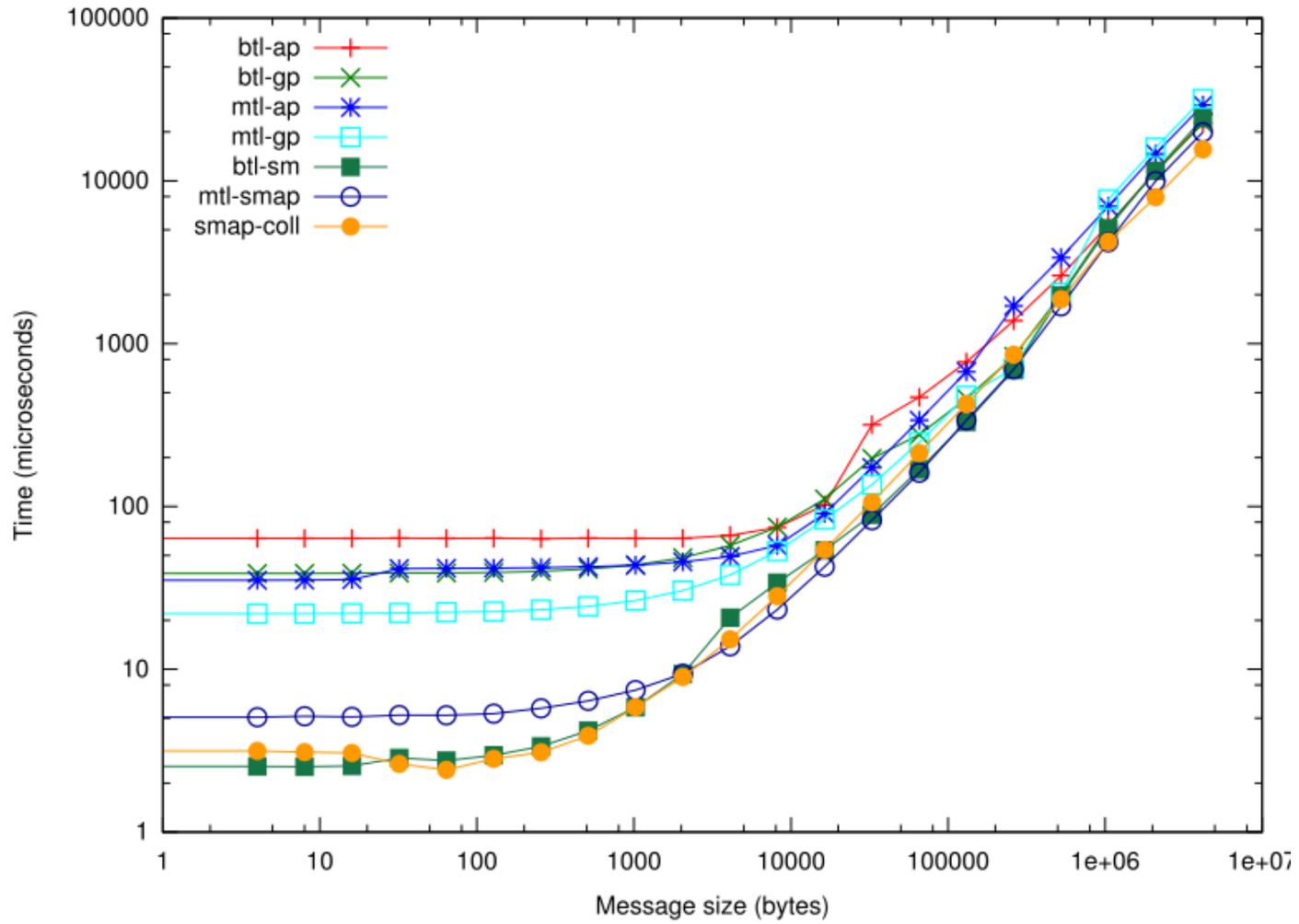
Bcast-4





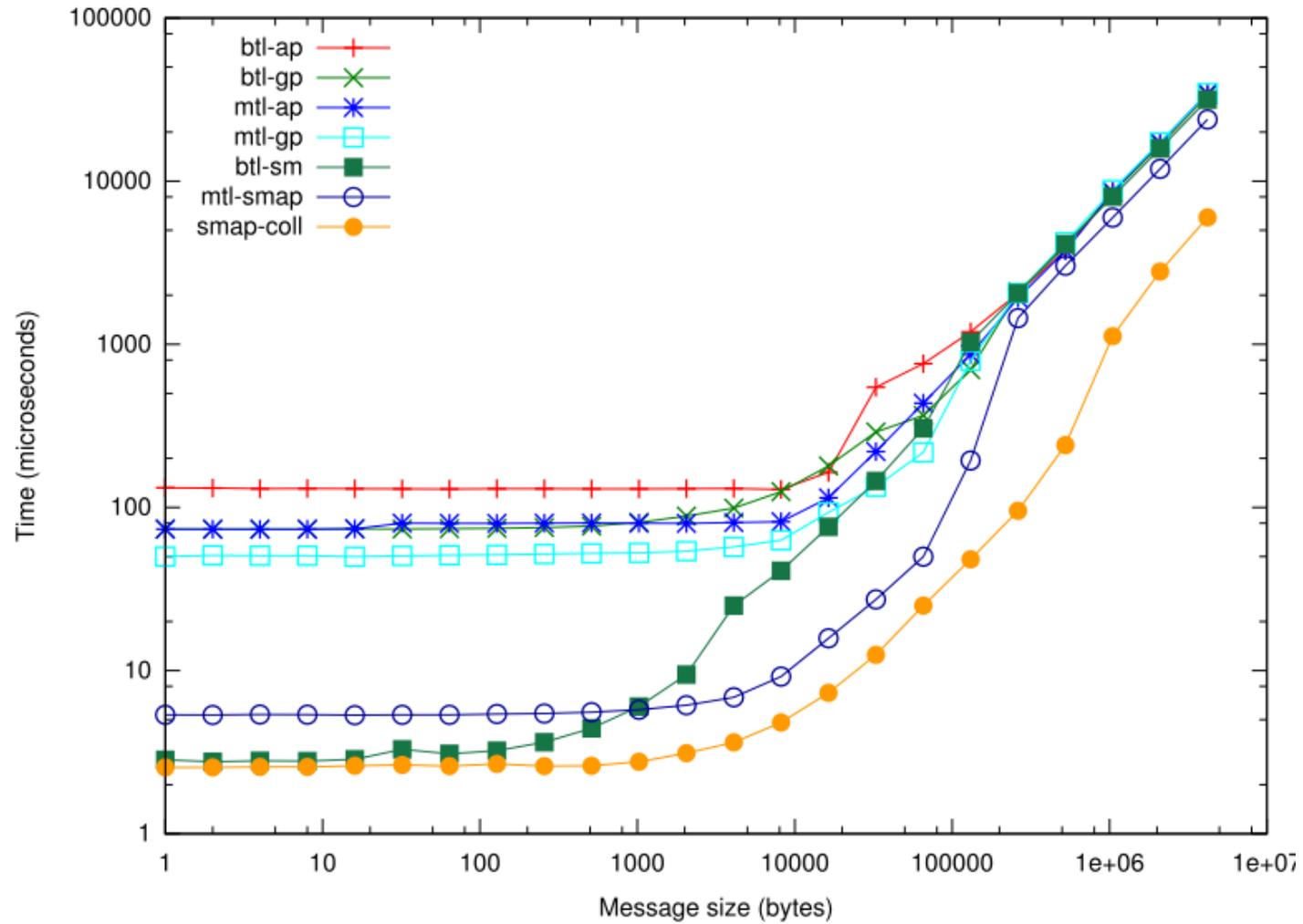


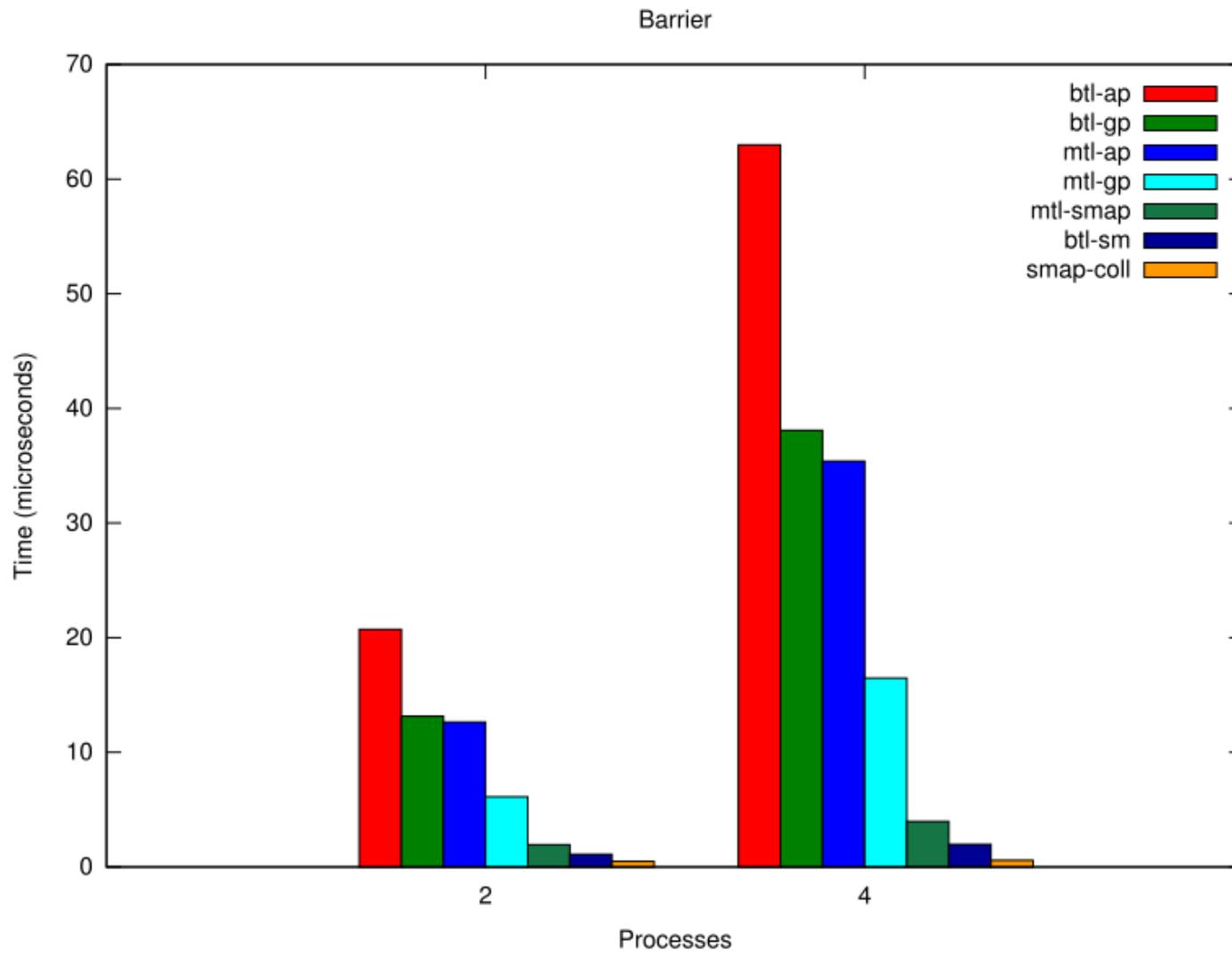
Allreduce-4





Alltoall-4







## Other Strategies

---

- Use communicators to expose hierarchy
  - MPI\_COMM\_NODE, MPI\_COMM\_NET
    - Collectives already do this
  - Can relieve pressure on network interface
  - MPI\_COMM\_SOCKET, MPI\_COMM\_CACHE??
- Partitioned *Nodal* Address Space model
  - Shared data spans processes on a node
  - MPI\_ALLOC\_MEM\_SHARED()
    - Use POSIX shared memory
  - Library and/or compiler support for using SMARTMAP directly
  - Avoids copying and/or data replication