



Optimizing an MPI Implementation to Increase CPU Availability

Arthur B. Maccabe, Christopher Wilson
University of New Mexico

Ron Brightwell, Bill Lawry
Sandia National Labs





Outline

- **Objective**
- **COMB benchmark suite**
- **ASCI/Red Hardware**
- **ASCI/Red Software**
- **Results and analysis**
- **Summary**



Objective

- **Demonstrate the utility of the COMB benchmark in helping to more completely analyze MPI performance**
- **Provide more insight than simple ping-pong latency and bandwidth measurements**



Communication Offload MPI Benchmark(COMB)



- **Measures the ability of an MPI implementation to overlap computation and communication**
- **Ability to overlap related to**
 - **Quality of MPI implementation**
 - **Capabilities of the underlying network layers**
- **Provides insight into the relationship between network performance and host CPU performance**
- **Needed to quantify the benefit of “application offload”**



COMB Design Goals

- **Quantify effectiveness of offloading MPI functionality to programmable NICs and Portals hardware**
- **Accurately measure**
 - CPU availability
 - Bandwidth
- **Portable**



Previous Work

- **Overhead**
 - **Netperf**
 - Two processes per node
 - Assumes process driving communication relinquishes the CPU
 - **Portability and accuracy issues**
- **Overlap**
 - **Pallas MPI benchmark's "exploit CPU method"**
 - **Various others, but no definitive metric for measuring overlap with respect to overall performance**



COMB Approach

- **Two nodes**
- **One process per node**
 - **Node 0**
 - **Facilitate communication**
 - **I/O**
 - **Node 1**
 - **Simulate computation**
 - **Communication**
 - **Timing**
- **Use MPI for portability**



COMB Approach (cont'd)

- Time a specified amount of work with no communication
- Time same amount of work with communication
- Ratio is CPU availability (1 – overhead)



COMB Methods

- **Poll**
 - Measures sustained maximum bandwidth
 - Perform communication throughout work
 - Allow for maximum possible overlap
- **Post-Work-Wait (PWW)**
 - Time all MPI calls and do work
 - Tests for overlap under practical restrictions on MPI calls



PWW Method

```
read current time
pre-post asynchronous send(s) & receive(s)
read current time
for ( i = 0; i < work interval; i++ ) {
    /* nothing */
}
read current time
wait for asynchronous send(s) & receive(s)
read current time
```



PWW Method (cont'd)

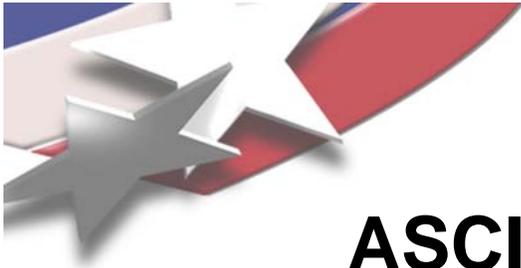
- Time work independent of messaging
- Collects wall clock times for different phases
 - Non-blocking post phase
 - Work phase
 - Wait phase
- Worker process waits for current batch of messages



ASCI/Red Hardware

- 4640 compute nodes
 - Dual 333 MHz Pentium II Xeons
 - 256 MB RAM
- 400 MB/sec bi-directional network links
- 38x32x2 mesh topology
- Red/Black switchable
- First machine to demonstrate 1+ TFLOPS
- 2.38/3.21 TFLOPS
- Deployed in 1997





ASCI/Red Compute Node Software

- **Puma lightweight kernel**
 - **Follow-on to Sandia/UNM Operating System (SUNMOS)**
 - **Developed for 1024-node nCUBE-2 in 1993 by Sandia/UNM**
 - **Ported to 1800-node Intel Paragon in 1995 by Sandia/UNM**
 - **Ported to Intel ASCI/Red in 1996 by Intel/Sandia**
 - **Productized as “Cougar” by Intel**



ASCI/Red Software (cont'd)

- **Puma/Cougar**
 - Space-shared model
 - Exposes all resources to applications
 - Consumes less than 1% of compute node memory
 - Four different execution modes for managing dual processors
 - Portals 2.0
 - High-performance message passing
 - Avoid buffering and memory copies
 - Supports multiple user-level libraries (MPI, Intel N/X, Vertex, etc.)



ASCI/Red Software (cont'd)

- **Puma/Cougar processor modes**
 - **Proc 0 (Heater mode)**
 - OS and application only use main CPU
 - **Proc 1 (Communication co-processor mode)**
 - OS dedicated to main CPU
 - Application dedicated to second CPU
 - **Proc 2 (Application co-processor mode)**
 - OS and application on main CPU
 - Application can spawn co-routines on second CPU
 - **Proc 3 (Virtual node mode)**
 - OS and one application process on main CPU
 - Separate application process on second CPU

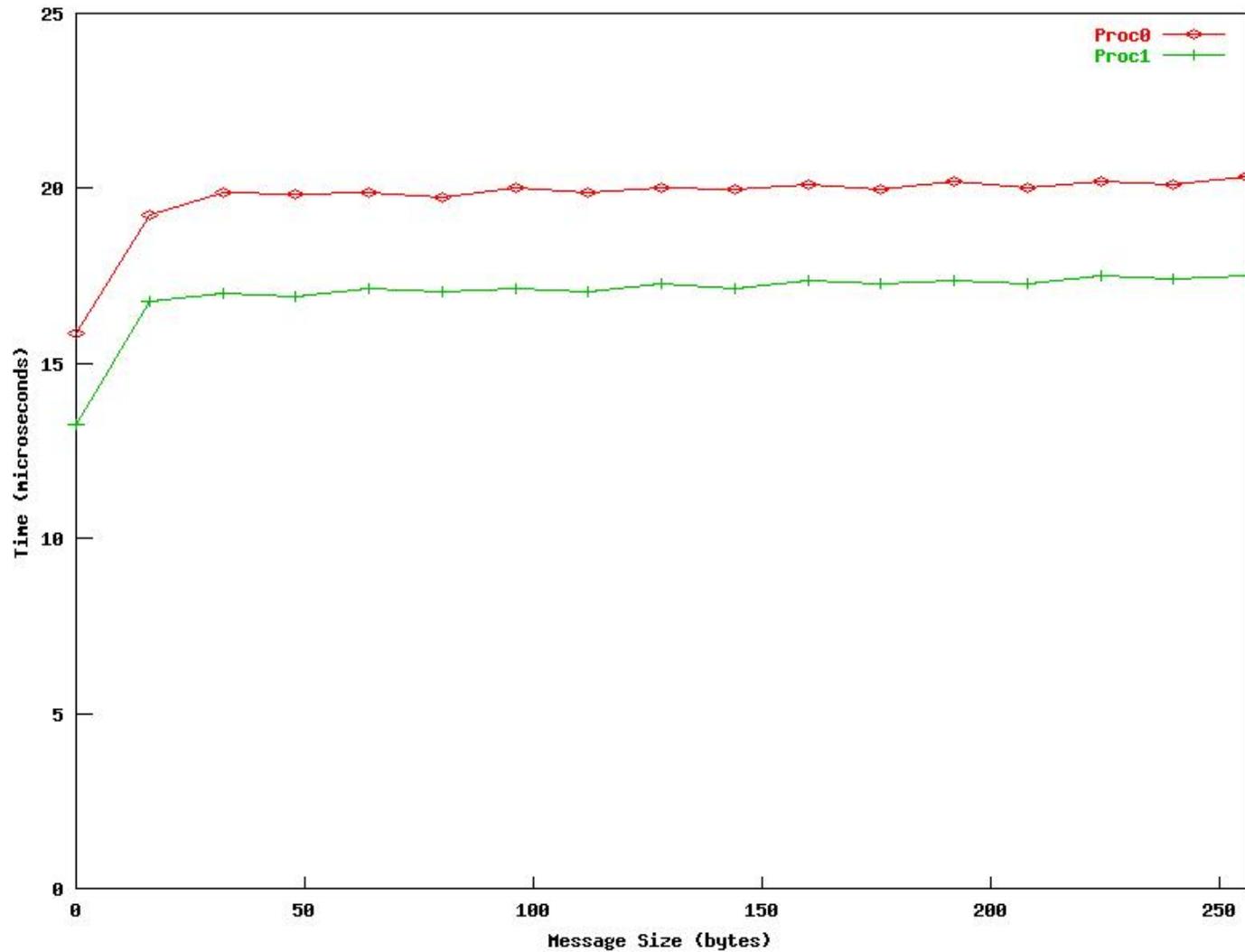


ASCI/Red MPI Implementation

- **MPICH 1.0.12 (1997)**
- **Direct ADI-1 device on Portals 2.0**
- **Validated as a product by Intel and never upgraded**

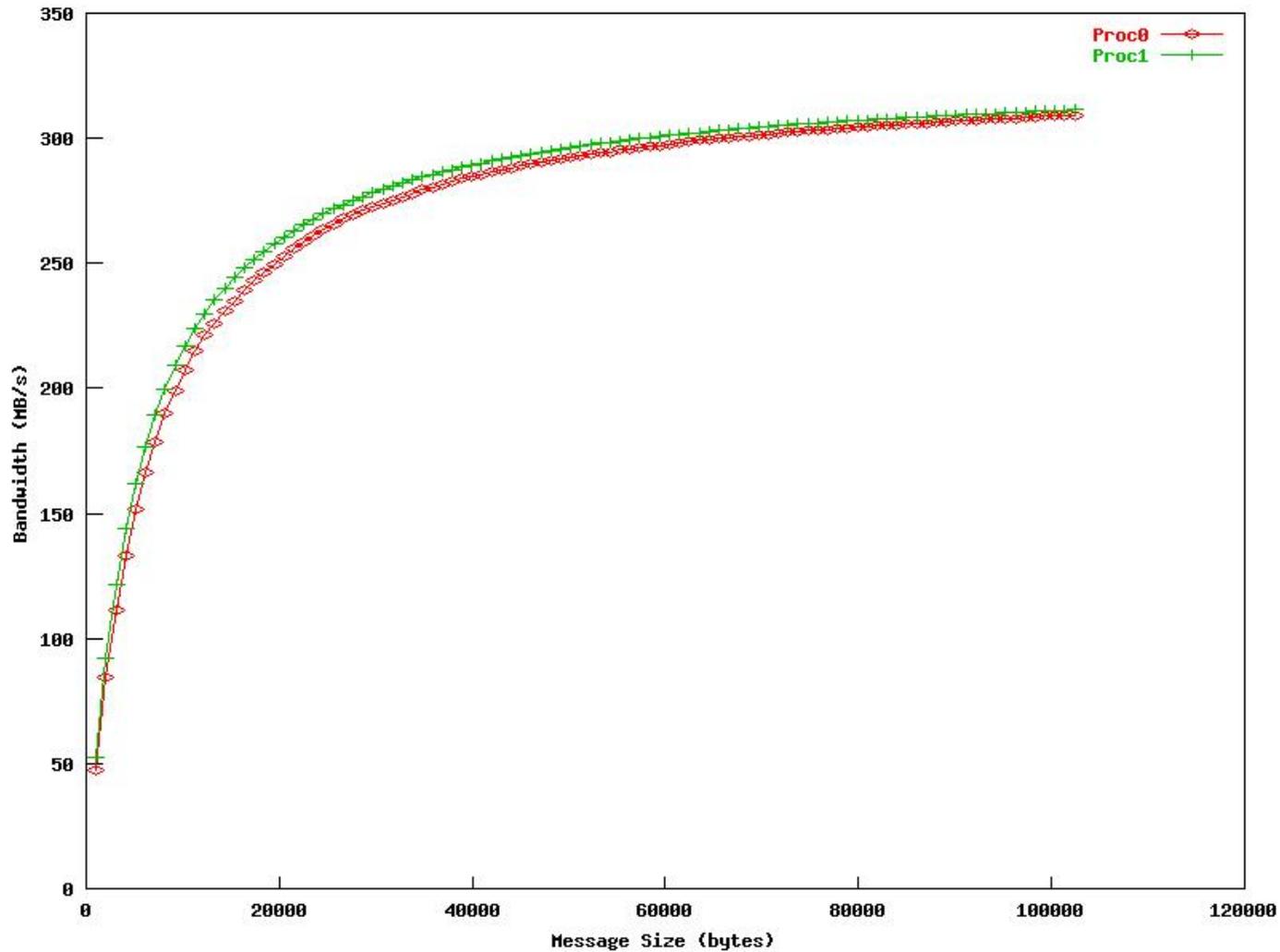


Half Round-Trip Latency



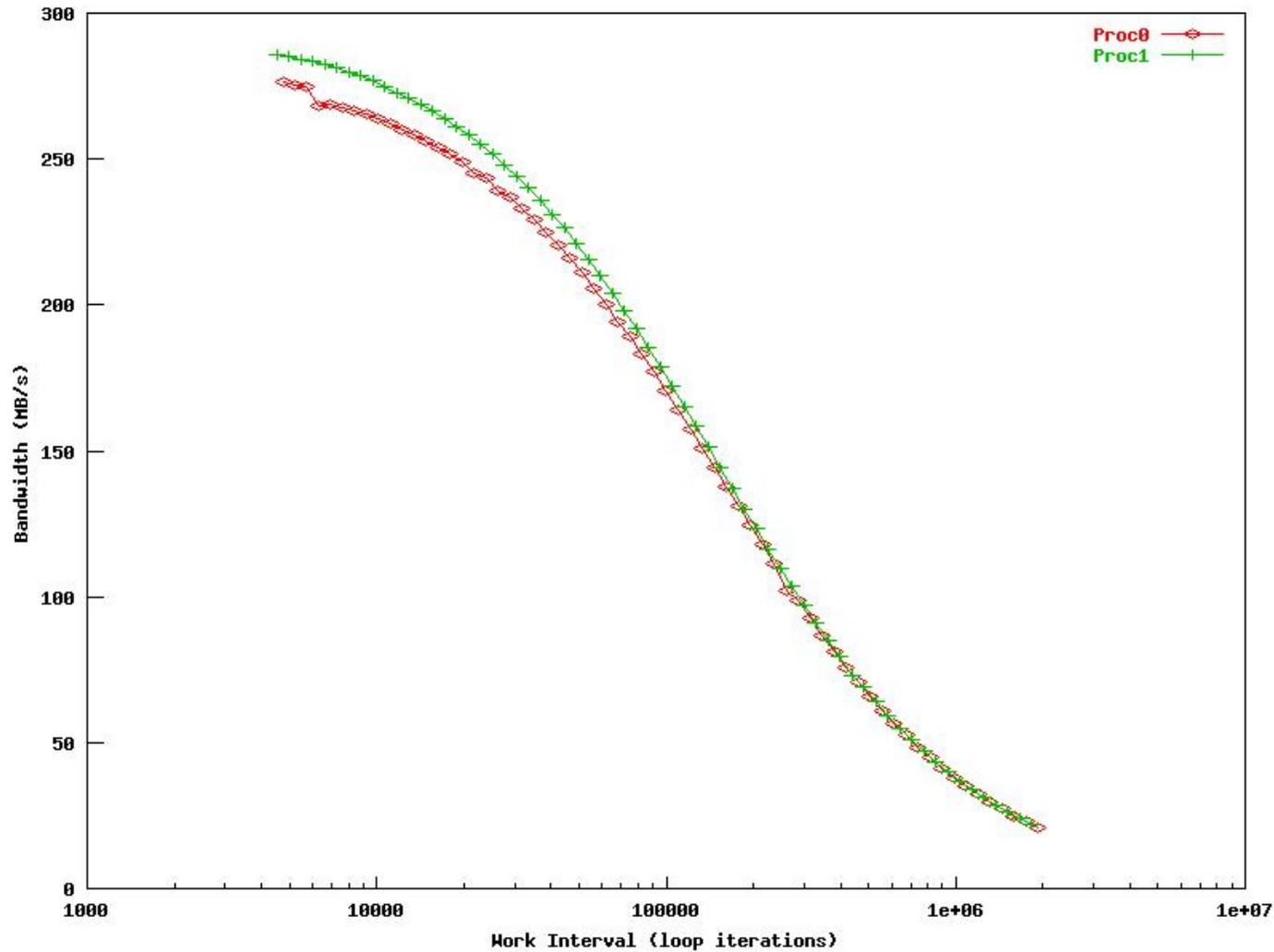


Ping-Pong Bandwidth



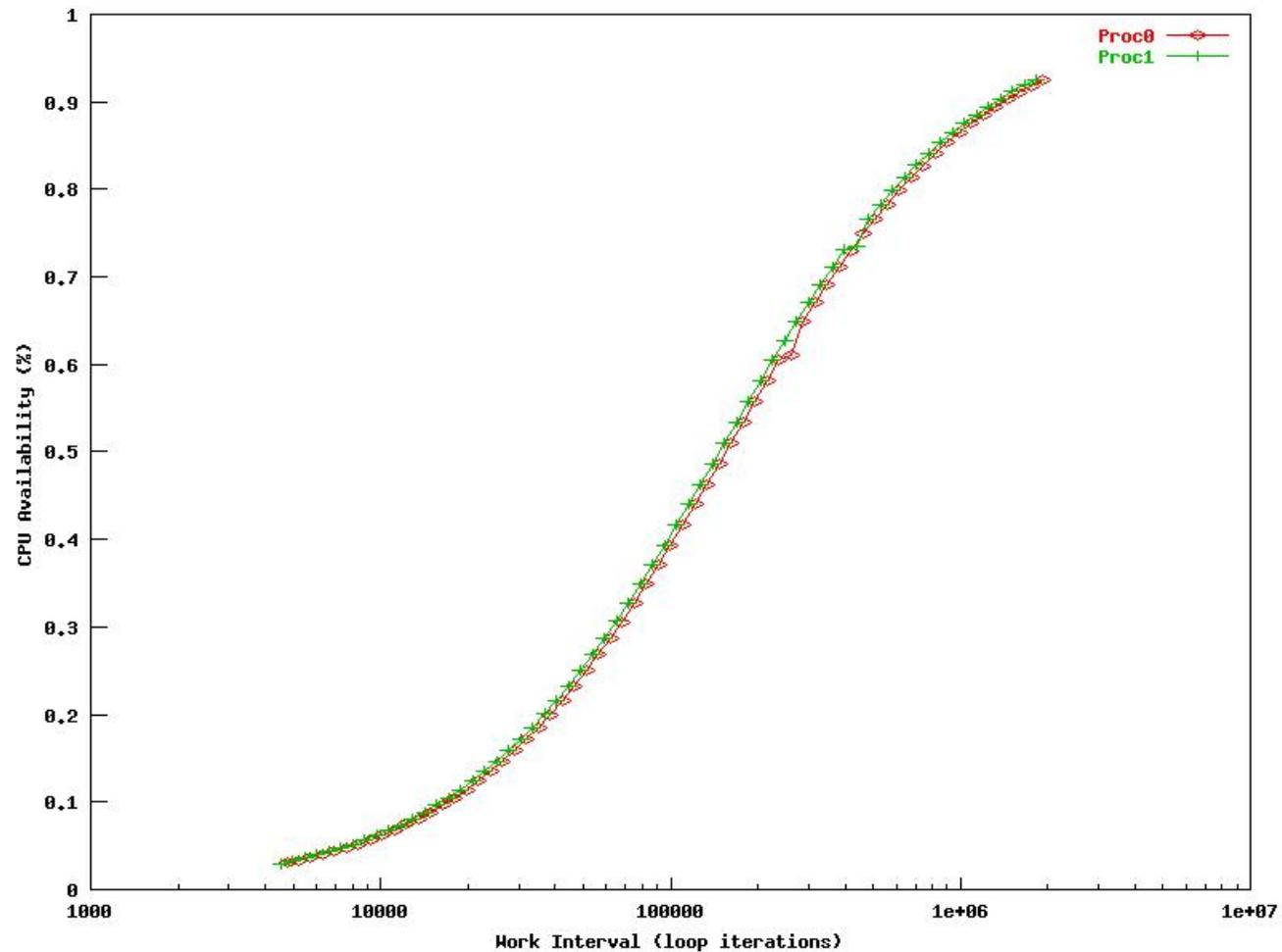


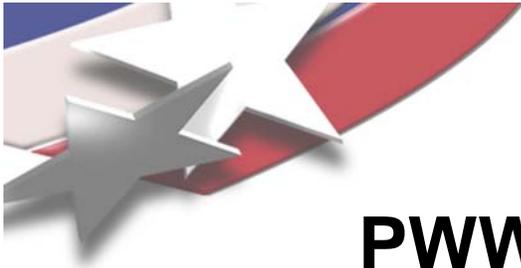
PWW: Bandwidth (100 KB)



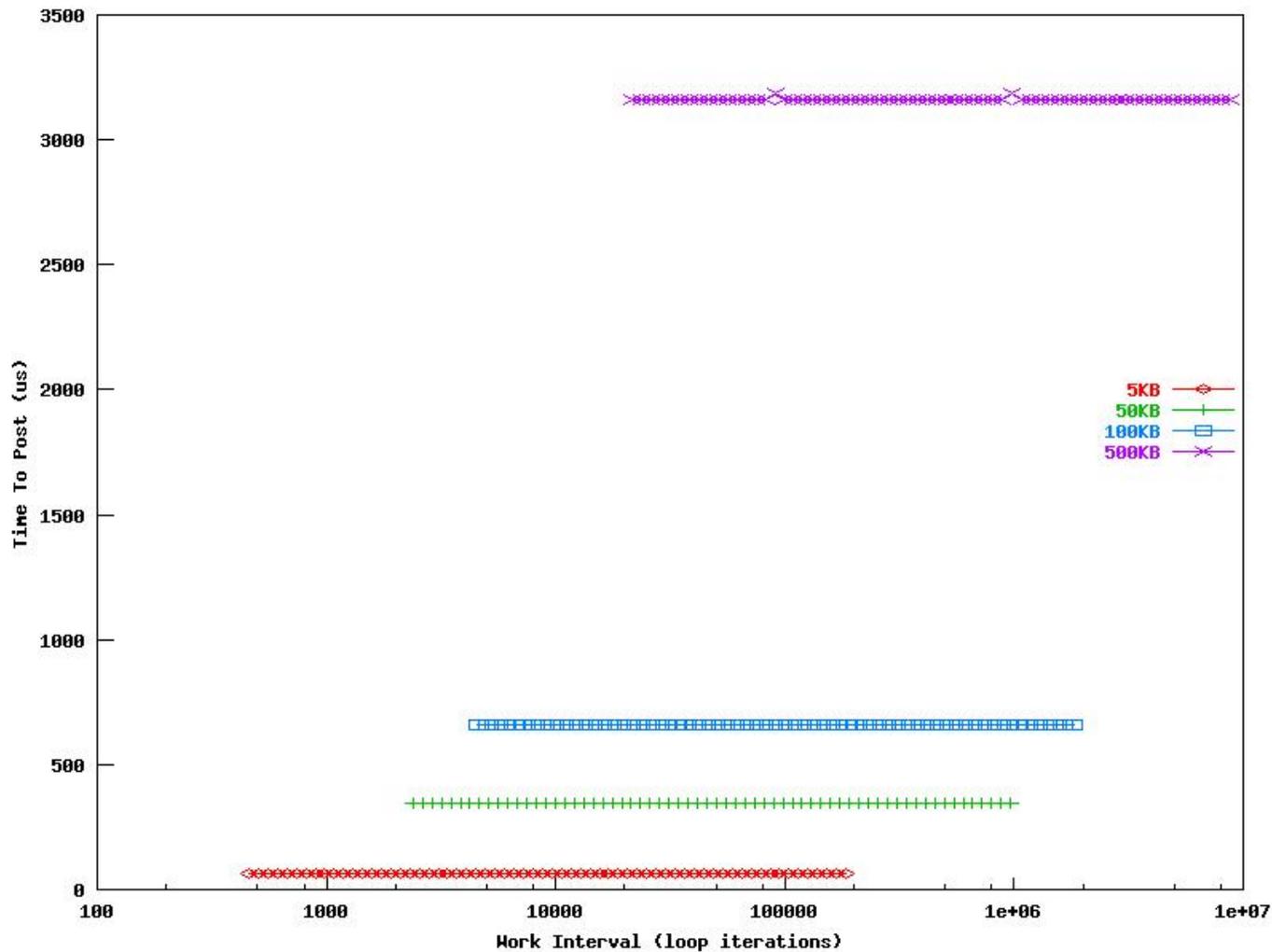


PWW: CPU Availability (100 KB)



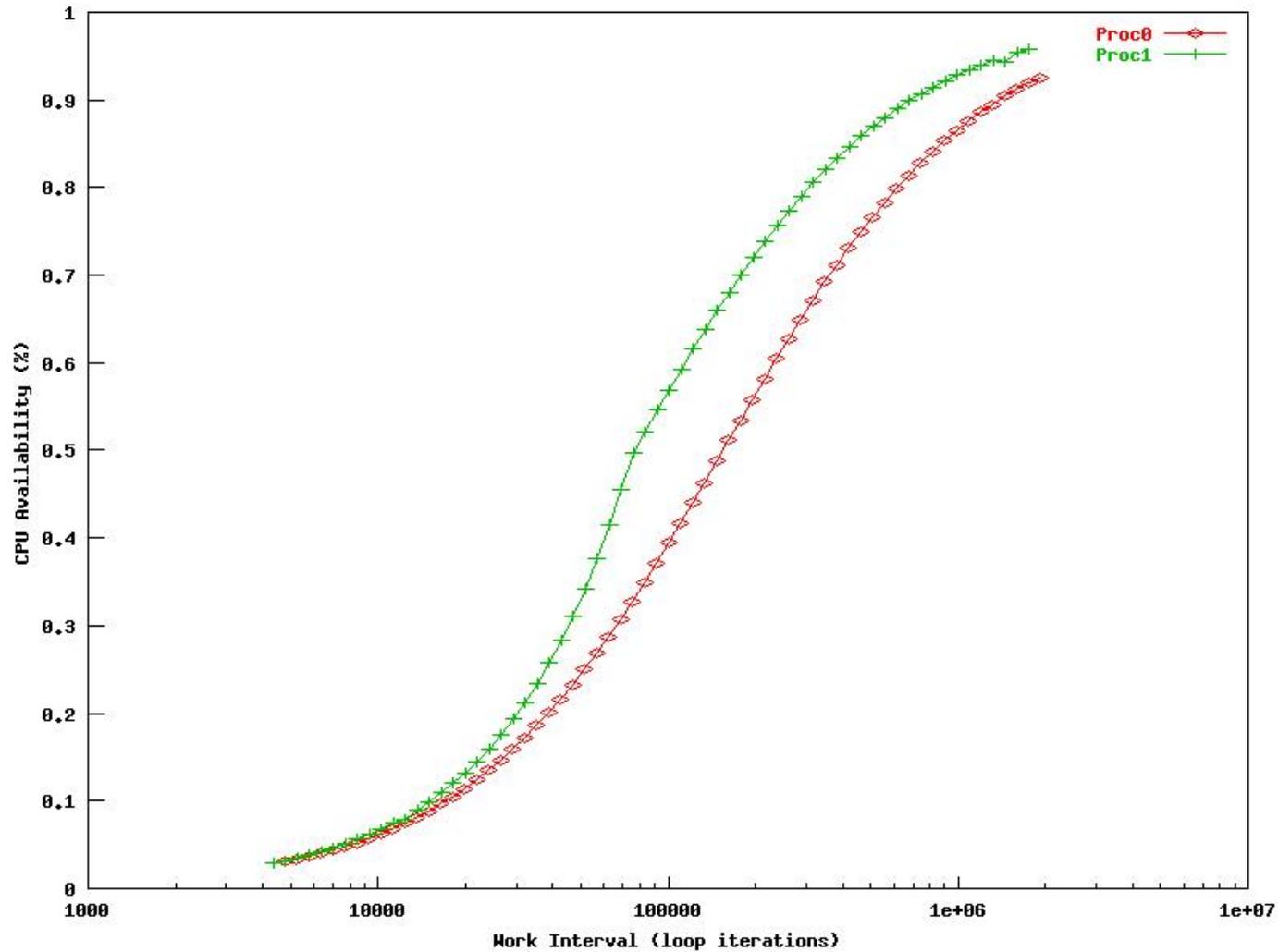


PWW: Time To Post Send (Proc 1)



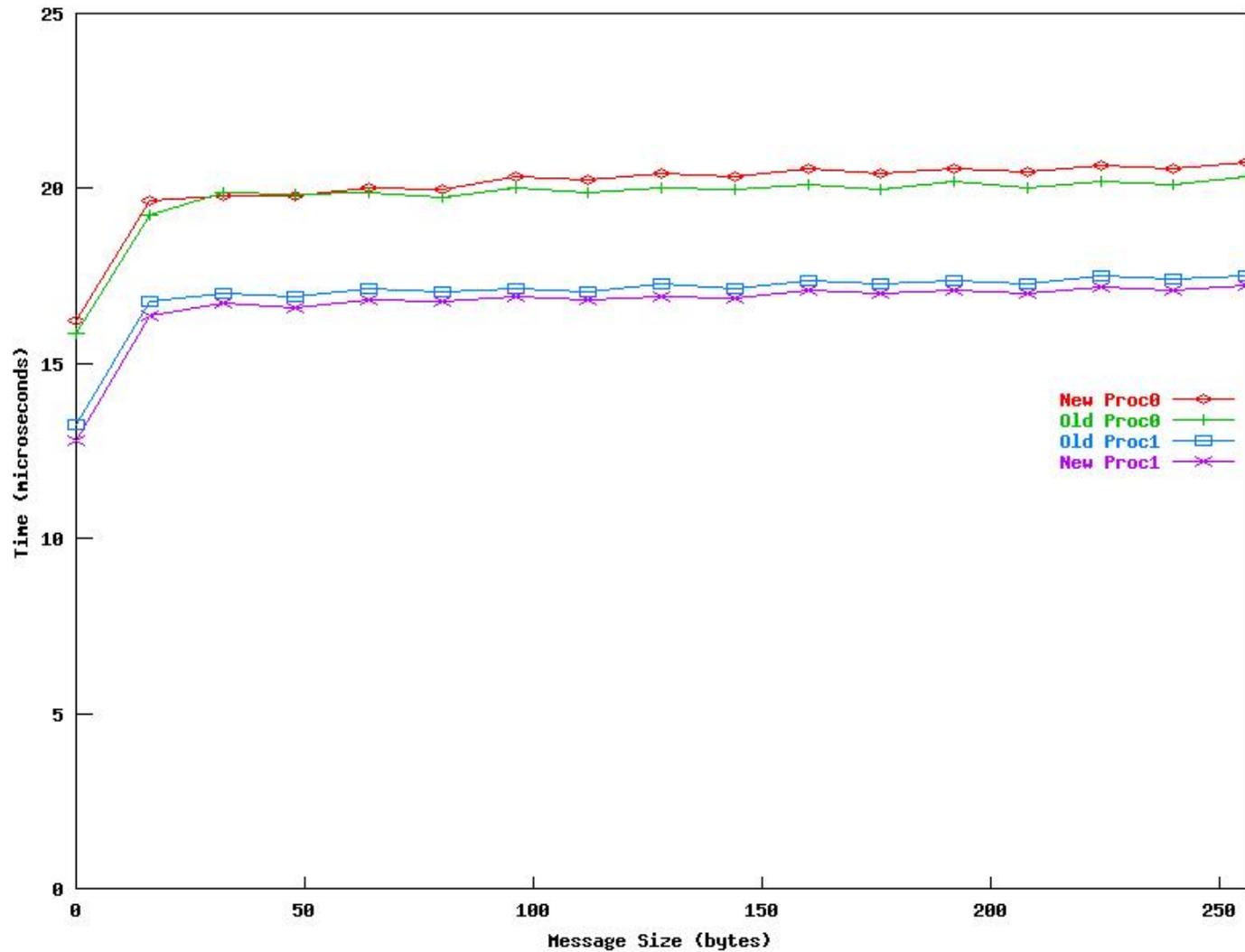


PWW: New CPU Availability (100 KB)



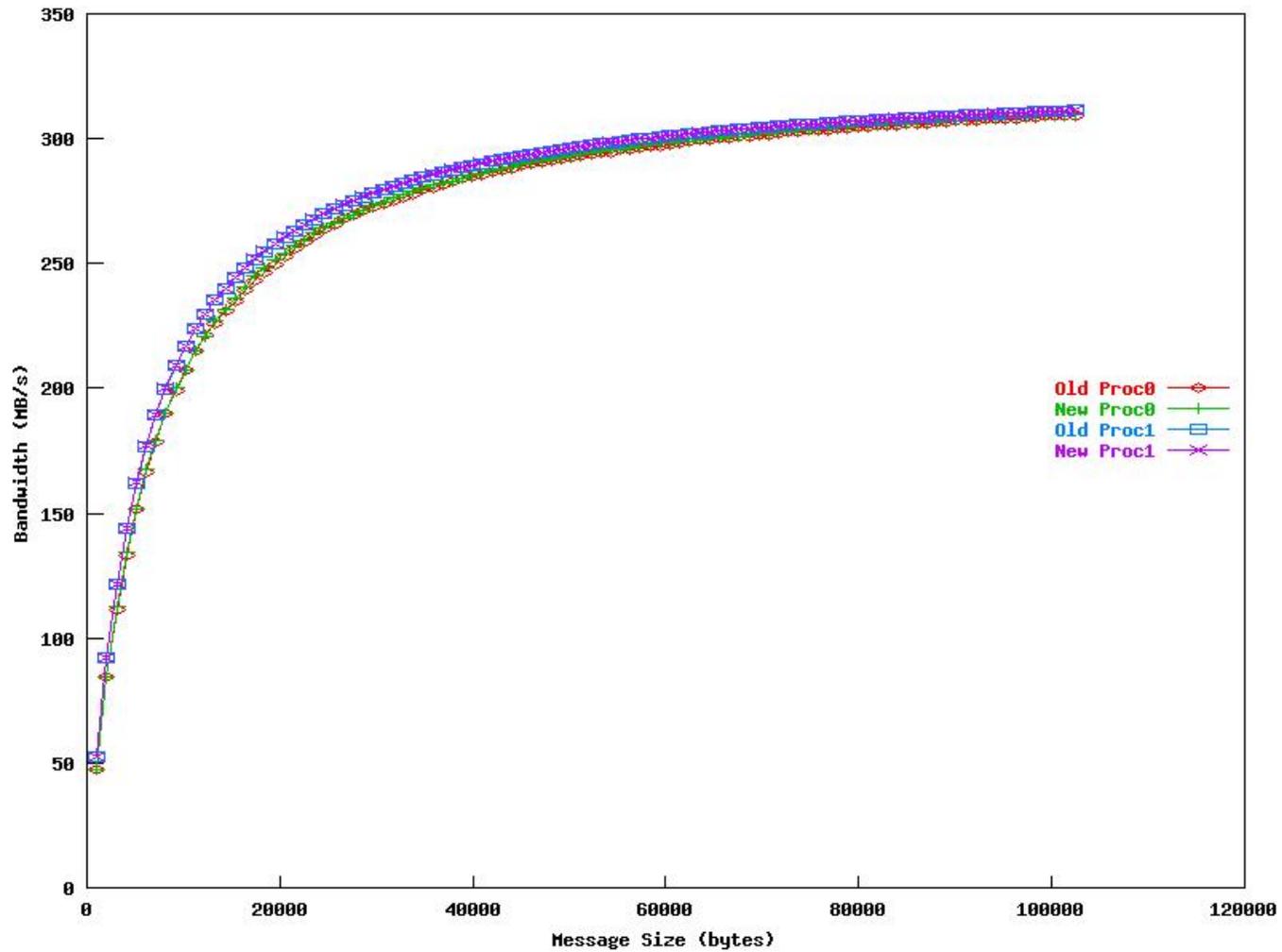


Half Round-Trip Latency





Ping-Pong Bandwidth





Summary

- **COMB measures the ability of an MPI implementation to overlap computation and communication**
- **COMB provides more insight into the relationship between network performance and host CPU performance**
- **Valuable tool in diagnosing a significant performance problem on ASCI/Red**