MPI
Ping Pong & Reduction

Mayank Chaudhary
Shrikant Vinchurkar

Approach followed (Ping Pong)...

- Many of today's computers wall clock timers (e.g. mpi_wtime) are not able to accurately measure small ping pong times.
- The lack of timer precision is usually overcome by timing many (often thousands of) ping pongs within a single timing loop and then taking the average.
- To calculate the timer overhead

```
do i = 1, n
   t1 = mpi_wtime()
   t2 = mpi_wtime()
   overhead = t2 – t1
enddo
```

Approach followed (Ping Pong) ... (contd)

- To calculate RTT

```
if (rank == source) then
        t1 = mpi_wtime()
        do i = 1, n
                call mpi_send()
                call mpi_recv()
        enddo
        t2 = mpi_wtime()
        time = t2 – t1 - overhead
endif
```

- To be precise, the number of iterations we considered is 1 million.

Overhead of Timer

0.036 microsec for 1M iterations

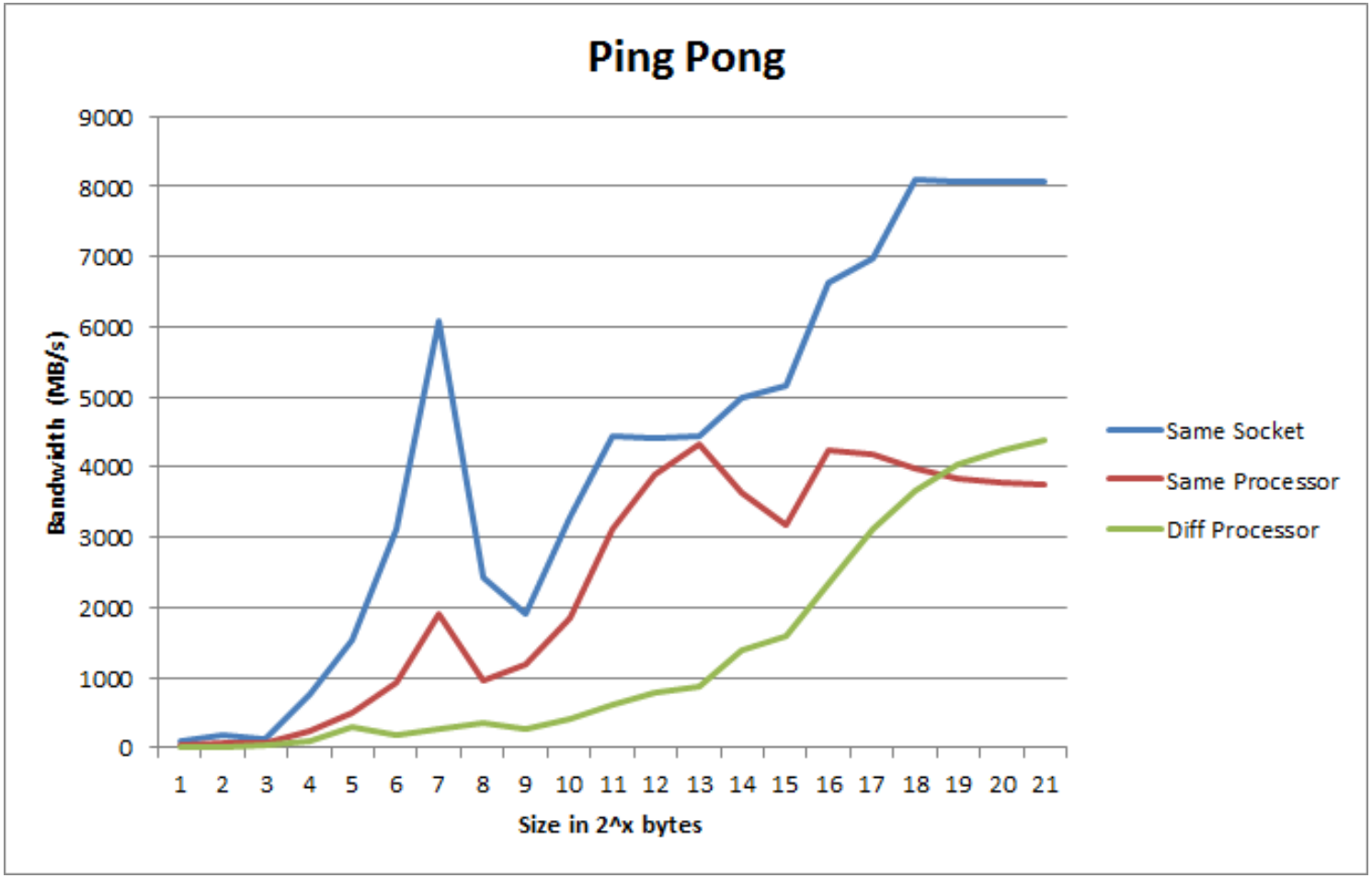
Latency for empty message(RTT)

Same processor - 0.70 microsec (1M iterations)

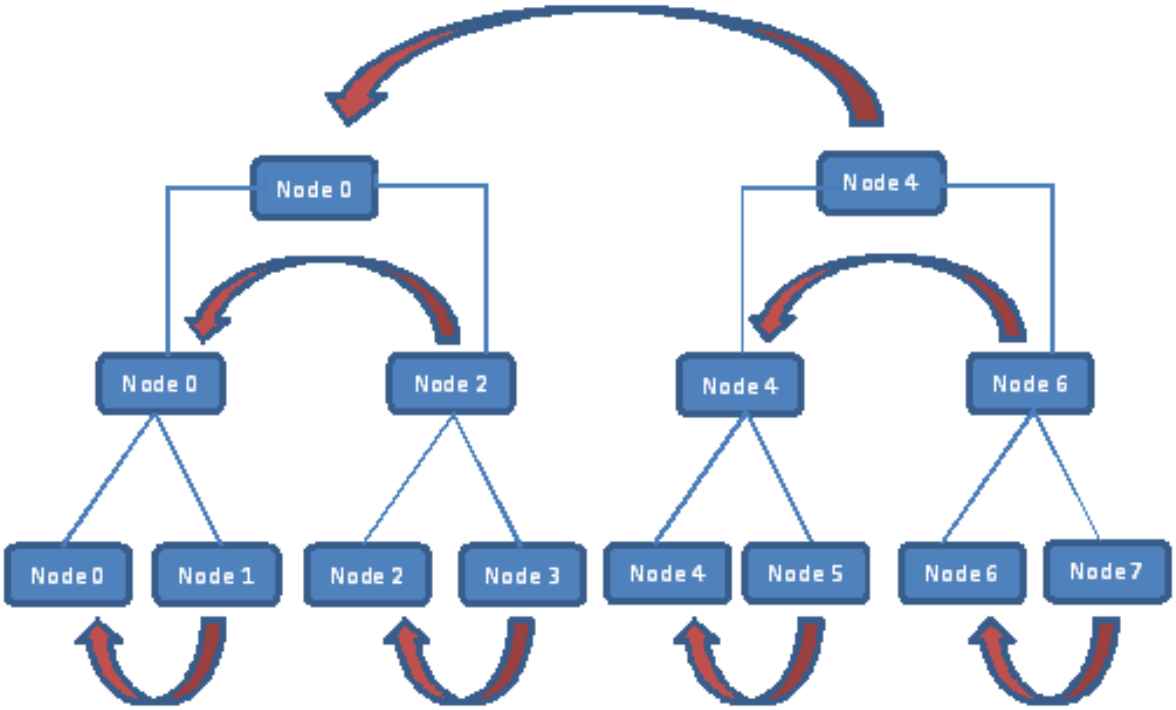
Same node - 1.63 microsec (1M iterations)

Different node - 3.69 microsec (1M iterations)

Bandwidth


Ping Pong
Bandwidth (MB/s)
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Size in 2^x bytes
Same Socket
Same Processor
Diff Processor

Approach for Reduction



Reduction tree for 8 nodes with curved arrows displaying communication at each level of tree. Each node is a MPI Process. At every level, local computed sum is sent from sender to receiver. The receiver then adds its own local sum to the sum it receives

Usage for Ping Pong & Reduction

- Ping Pong

  - mpiexec -n {no of mpi processes} ./PinPon {source} {destination}

- Reduction

  - mpiexec -n {no of mpi processes} -/Reduction {Array_SIze}