PRACE Course: Intermediate MPI

Day 1

9/11/2022

1 Ping Pong Benchmark

Write a simple ping pong program. This involves passing of a message between two processes P_0 and P_1. The algorithm is as follows

1. let us assume we have an initial message which contains the integer value 10.
2. P_0 increments this message by one and passes it to P_1 (ping)
3. P_1 receives the message it increments it by one and passes it back to P_0
4. the last two steps are repeated n times. (see fig. 1)
5. exclude startup time from measurements

The code shall do

1. rank 0 prints the value of the message it has after n exchanges.
2. rank 0 prints the average time per exchange. **Hint:** use MPI_Wtime() function to get the time.
3. determine the value n for which the measured time is meaningful. **Hint:** check the resolution of the timer with MPI_Wtick()
4. test with the following send modes: MPI_Send and MPI_Ssend.

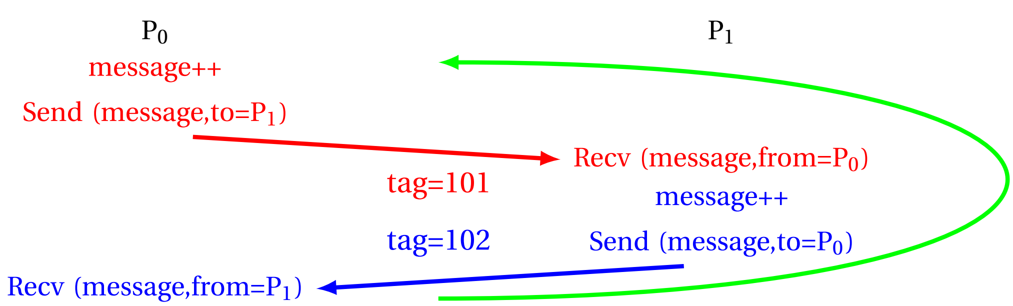


Figure 1. Ping Pong algorithm

5. Latency is defined as the time to transfer a zero length message. Modify the ping pong code to measure the latency (use MPI_CHAR as transfer type).
6. Bandwidth is defined as the siz e of the message in bytes/ transfer time. Modify the ping pong code to measure the Bandwidth (use double or real(kind=8)). Measure the bandwidth for the following sizes 8 B, 512 B, 32 KiB, 2MiB these correspond to arrays of length 1, 2^6, 2^{12} and 2^{18}, respectively.

2 Communication in a Ring

1. Write a MPI program where each rank sends a message to its right neighbour.
2. The message is passed around the ring until it reaches the originator rank.
3. At this point the **message** should contain the sum of all the ranks.
4. So at termination all ranks should have the sum of the ranks.
5. Use non-blocking communications.
6. Write an equivalent operation that does the same using MPI_Ireduce().

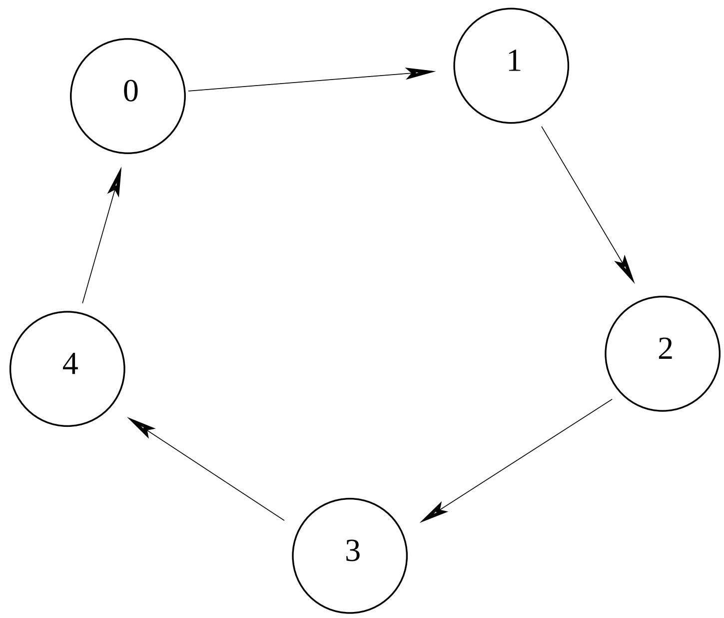


Figure 2. Five processes logically arranged in a ring: the left neighbor of P_0 is P_4 and its right neighbor is P_1.

3 Collectives: Allgather/Gatherv/Reduce

The serial code is given (inc_serial.c / inc_serial.f90). It increments every element of the array by two. Parallelise using MPI as follows.

1. A real vector a of size length is allocated at Process 0.
2. Each process contains a chunk of the vector a_per_process. Length doesn't have to be a multiple of the number of processes. Calculate the size of chunk at each process, allocate the subvector and initialise as 0.0.

3. Each process gets the length of a_per_process from all other processes using MPI_Allgather, calculates the recvcounts and displs arrays.

4. Each process increments every element of the subvectors by 2.0. Execute the increment operation n number of times.

5. Gather subvectors into a in Process 0 using MPI_Gatherv.

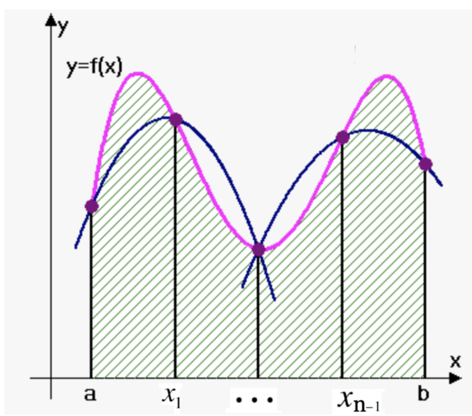
6. Each process has its own timing. Reduce the time at Process 0 to find the maximum average time.

4 Ekstra: Simpson's Rule

Computing the area under the curve of $f(x)$ where $x \in [a,b]$ can be done using the Simpson's rule:

$$\int_a^b f(x)dx \sim \frac{h}{3}(f(x_0) + 4\sum_{i=1,3,\cdots}^{n-1} f(x_i) + 2\sum_{i=2,4,\cdots}^{n-2} f(x_i) + f(x_n))$$

where $x_0 = a$, $x_n = b$ and $h = \frac{b-a}{n}$ with $n-1$ equidistant points between a and b.



The interval $[a,b]$ is partitioned into the set $\{a = x_0, x_1, x_2, ..., x_{n-1}, b = x_n\}$ so that there are n sub-intervals of equal width h where n is an even number. The shaded area bounded by the parabolas is approximately equal to the area bounded by $y = f(x)$.

Find the integral of $f(x) = \sin(x) * sin(x)$ from $0 \to 90$. Compare with the actual result: $\int_0^{90} sin^2(x) = \pi/4$.

The serial code is given (simpson_serial.c / simpson_serial.f90). Parallelise using MPI as follows.

1. Process 0 reads in a, b, and n, and sends them to all other processes.

2. The interval [a, b] is distributed among processes. Each process calculates its subinterval: $[a_{local}, b_{local}]$ which is split up into n_{local} sub-intervals of width h_{local}.

3. Calculate the values of $sin^2(x)$ where x is in radians in the range $[a_{local}, b_{local}]$ every h_{local} degrees. The value of the pts and the value of the function at these pts as arrays. Use the provided function (or subroutine) to convert degrees to radians.

4. Each process applies Simpson's function over its local interval divided into local sub-intervals.

5. Local approximated areas found by each process are summed at process 0 to a final estimation value.

6. Compare the approximated area with the exact value of integral with different number of processes.