TP - Distributed-memory programming: MPI (II)

ING2-GSI-MI – Architecture et programmation parallèle

Academic year 2023–2024

First steps with collective communication calls

(1) From the `skeletons.zip` file on Arel, open the `01-showroom.c` file. It contains some MPI collective operations. Compile it and execute it with 4 processes. Make sure that you understand the code and how each collective call uses the ids and $rxids$ arrays in a different way. ☐

Matrix operations

(2) Revisit the code of the matrix product from exercise 5 in the previous TP. Rewrite it using collective operations when possible. Note that:

— Matrix B will be sent to all slave processes (*broadcast*)

— Matrix A will be distributed by row chunks (*scatter*)

☐

(3) Following the process described in pages 19 and 20 from the course slides, use the `MPI_Reduce_scatter` function to implement a matrix-vector product. ☐

Normalisation

4 Let us normalise a v_i vector using MPI:

$$v_i = \frac{v_i}{\sum_j v_j}$$

Programmatically, we can implement it using the following sequential code:

```
sum = 0;
for(j=0;j<N;j++) sum = sum + v[j];
for(i=0;i<N;i++) v[i] = v[i] / sum;
```

To implement it in MPI, follow these steps:

1. Generate an array of size N (Process 0)
2. Broadcast N or $N/NProcs$ (size of the local vector)
3. Scatter vector v (only the chunk corresponding to each process)
4. Compute locally the partial sum (i.e. each process computes the sum on its chunk)
5. Allreduce of the partial sums (so that in the end all processes receive all values)
6. Compute locally $\frac{v_i}{\sum_j v_j}$
7. Gather of the local vector (to Process 0)
8. Result output (Process 0)

□