CS 426/525: Project 3

Spring 2025

In this project, you will perform the matrix-vector multiplication operation in the parallel conjugate solver you implemented in the second project using 2D (checkerboard) partitioning.

In this approach, a square matrix A of size $n \times n$ is divided into a block-checkerboard pattern and distributed across a $\sqrt{p} \times \sqrt{p}$ mesh of processors where p is the total number of processors and each processor is assigned to a square subblock of $A_{\alpha\beta}$. The vectors X and Y are also partitioned into slices X_β and Y_α corresponding to the matrix partitions. The matrix-vector multiplication operation $Y = A \cdot X$ is performed in four main phases:

- **Expand Phase:** Each processor in a column receives the relevant portion of vector X required for local computation.
- **Local Computation Phase:** Each processor $P_{\alpha\beta}$ computes the partial result $Y_{\alpha\beta} = A_{\alpha\beta} \cdot X_\beta$.
- **Fold Phase:** The processors in each row perform a fold operation to sum up their local contributions $Y_{\alpha\beta}$ and obtain the full slice Y_α.
- **Transpose Phase:** The partial results are transposed so that each processor ends up with the final value of the corresponding slice of vector Y it is responsible for.

You can find more detailed information in the slide and paper shared with you.

Directives and Notes

- You may use the data you created in Project 2 for this project as well.
- You may use the same serial code that you implemented for the previous project.

- After reading the input matrix and vector, the master process should distribute them to the other processes. It should also collect the final output vector from the other processes and write it to a file. **In this project, the master process should not participate in computation, it should only coordinate input/output operations.** (You may assume that the number of rows in the matrix and vectors is divisible by the number of processors.)
- Communication between processes/cores during the matrix-vector multiplication operation must be P2P.
- You should start the timer after the initial distribution phase is finished and end it just before the final reduction of the output vector.
- Name of the source code file and the output file (txt or out) must be parallel.c and p_output.txt/out, respectively.

Experiments

- Run your code on ORFOZ.
- Run your parallel implementation on 16, 32, 64, 128, and 256 processes/cores and observe the runtimes while process/core count increases.
- Provide at least 2 strong scaling curves with different input sizes. Compare the 1D row-parallel matrix-vector multiplication you implemented in the previous project with this new 2D version as well. Your strong scaling curves should include both instances.
- Provide a weak scaling curve.
- Use MPI_Wtime function for time measurements and do not forget to put necessary barriers while measuring the time.
- To obtain stable runtime results, take runs at least two different times (on different days maybe).
- Before starting your measurements, run your solver a few times to avoid unstable runtime results caused by cold starts. (Warm-up)

Submission Format

Submit a single zip file containing source files, a Makefile, a SLURM script file and a project report discussing your findings to Moodle. name_surname_p3.zip file should include the followings:

- Your implementation with source file:
  - parallel.c

- A makefile that generates the following executable:
  - parallel
- Your report (3 pages at most):
  - Briefly explain your implementations. Plot graph(s) with various thread numbers and input sizes indicating the performance of your implementation by comparing serial and parallel. What are your observations? Does parallel implementation improve performance? If not, what might be the reasons?

Grading

- Parallel Implementation: 80 points
- Report: 20 points

Important Notes

- **DEADLINE: May 15, 2025 23:59**
- **No Late Submission Allowed!**
- Your submission file **MUST BE** in a **ZIP** format. (not RAR, TARBALL etc.)
- In order for your project to be graded, it must be able to compile with the Makefile you provided and must run without any problem on the TRUBA/ARF system with the SLURM script file you provided.
- You can always reach out to me via email. (can.bagirgan@bilkent.edu.tr)