Wikiprint Book

Title: Integrating TAMPI and ParaStationMPI NAM windows

Subject: DEEP - Public/User_Guide/TAMPI_NAM

Version: 25

Date: 28.04.2024 15:33:47

Table of Contents

Integrating TAMPI and ParaStationMPI NAM windows

Table of contents:

Quick Overview

Heat Benchmark

In this section, we exemplify the use of TAMPI and NAM windows through the Heat benchmark. We use an iterative Gauss-Seidel method to solve the Heat equation, which is a parabolic partial differential equation that describes the distribution of heat in a given region over time. This benchmark simulates the heat diffusion on a 2D matrix of floating-point elements during multiple timesteps. The 2D matrix is logically divided into 2D blocks, potentially with multiple block rows and block colmuns. In a timestep `t`, the computation of an element at position `M[r][c]` depends on the value of the top and left elements (`M[r-1][c]` and `M[r][c-1]` respectively) computed in the current timestep `t`, and the right and bottom elements (`M[r][c+1]` and `M[r+1][c]` respectively) from the previous timestep `t-1`. We can extrapolate this logic in the context of blocks so that a block has a dependency on the computation of its adjacent blocks. Take into account that the computation of blocks in a diagonal is concurrent given that they do not have any dependency among them.

There are three different MPI versions and all them distribute the 2D matrix across ranks assigning a consecutive rows of blocks to each MPI rank. Thus, the matrix is block-based distributed vertically but not horizontally. Each MPI rank has a neighboring rank on the top and another on the bottom, except the first and last ranks. This distribution requires the neighboring ranks to exchange the halo rows in order to compute their local blocks in each timestep.

The first version is based on an MPI-only parallelization, while the other two are hybrid MPI+OmpSs?-2 leveraging tasks and the TAMPI library. We briefly describe each version below:

- `01.heat_mpi.bin`: A straightforward MPI-only implementation using blocking MPI primitives (MPI_Send and MPI_Recv) to send and receive the halo rows.
- `02.heat_itampi_ompss2_tasks.bin`: A hybrid MPI+OmpSs?-2 version that instantiates a task to compute each of the blocks inside each rank and for each of the timesteps. It creates a task for sending and receiving the halo rows for each of the external blocks. The execution of tasks follow a data-flow model because tasks declare the dependencies on the data they reading/modifying. Moreover, communication tasks leverage the non-blocking mechanism of TAMPI (`TAMPI_Iwait`), so communications are fully non-blocking and asynchronous from the user point of view.
- `03.heat_tampirma_ompss2_tasks.bin`: An implementation similar to `02.heat_itampi_ompss2_tasks.bin` but using MPI RMA operations (`MPI_Put`) to exchange the halo rows. This version also leverages TAMPI and uses the new `MPI_Win_ifence` function. The `TAMPI_Iwait` mechanism is used to perform the `MPI_Win_ifence` completely non-blocking and asynchronously. The calls to `MPI_Put` are assumed to be already non-blocking. We register multiple MPI RMA windows to allow concurrent communications through the different RMA windows. Each communication task exchanges the part of the halo row assigned to a single MPI window.

References

- ?https://pm.bsc.es/ompss-2
- ?https://github.com/bsc-pm
- ?https://github.com/bsc-pm/tampi
- ?https://en.wikipedia.org/wiki/Gauss-Seidel_method
- ?https://pm.bsc.es/gitlab/DEEP-EST/apps/Heat