High Performance Computing 2023 - Exercise 2c

Marco Zampar - SM3800032

September 25, 2024

Problem Statement

- The goal is to parallelize the computation of the **Mandelbrot Set** using:
  - **MPI** (Message Passing Interface) for distributed memory parallelism
  - **OpenMP** for shared memory parallelism
- Mandelbrot Set is computed by checking the boundedness of a recursive sequence for each complex point `c`.

[
z_{n+1} = z_n^2 + c
]

If (|z_n| < 2) for `n <= Imax`, the point is in the set.

Mandelbrot Set Visualization

- The Mandelbrot set is represented as a 2D grid.
- Each pixel is checked for boundedness.
- **Symmetry** along the x-axis can be leveraged for optimization.

Computational Resources

- The computation was performed on **2 EPYC nodes** of the **ORFEO cluster**.
    - **OpenMP**: 64 cores per socket (4 NUMA regions).
    - **MPI**: 2 nodes (256 cores total).
- Resources were carefully allocated:
    - **MPI process binding**: Each process bound to a socket.
    - **OpenMP thread binding**: Threads were placed using `OMP_PLACES` and `OMP_PROC_BIND`.

Implementation Details

- **OpenMP**:
    - Dynamic scheduling to handle workload imbalance across rows.
- **MPI**:
    - Rows distributed based on process rank.
    - Balanced workload, with processes getting rows based on rank.
- **I/O Strategy**:
    - Avoided MPI parallel I/O due to overhead.
    - Used **MPI_Gatherv** for gathering data on rank 0, then serial writing to the PGM file.

Scaling Performance

- We evaluated **strong scaling** and **weak scaling** for both **MPI** and **OpenMP**.

MPI Strong Scaling

- **Serial execution time**: 156.17 s
- **256-process execution time**: 0.97 s
- **Speedup**: 161.18 (99% reduction in time)

OpenMP Strong Scaling

- **Serial execution time**: 156.74 s
- **64-thread execution time**: 2.44 s
- **Speedup**: 64.2 (98.5% reduction in time)

Weak Scaling

- **MPI** weak scaling shows linear increase in time due to the overhead of `MPI_Gatherv`.
- **OpenMP** weak scaling is efficient up to 64 threads, after which resource contention increases the execution time.

Final Considerations

- **OpenMP** scales efficiently up to 64 threads but struggles beyond that due to resource contention.
- **MPI** can handle larger-scale problems but incurs overhead due to communication.

Hybrid Scaling

- We tested **Hybrid MPI+OpenMP scaling** using both processes and threads.
- The best configuration achieved an execution time of 1.61 s (99% reduction).