Identifying Focus-of-Analysis Regions in MPI-Traces Using Transfer-Efficiency Monitors

Kingshuk Haldar

High Performance Computing Center Stuttgart (HLRS), University of Stuttgart
kingshuk.haldar@hlrs.de

Introduction

- HPC applications do not exhibit uniform performance characteristics throughout execution.
- Regions with different performance characteristics are the different *Focuses of Analysis* or *FoA*s.

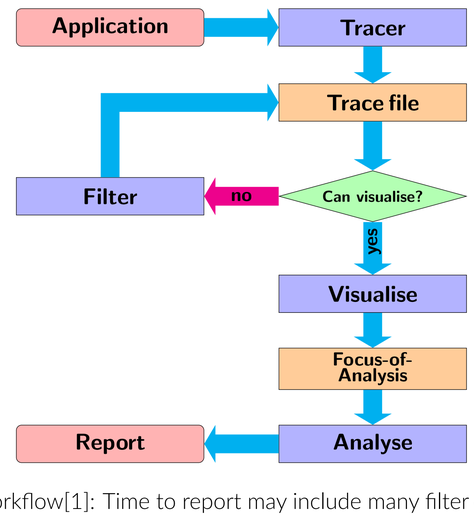
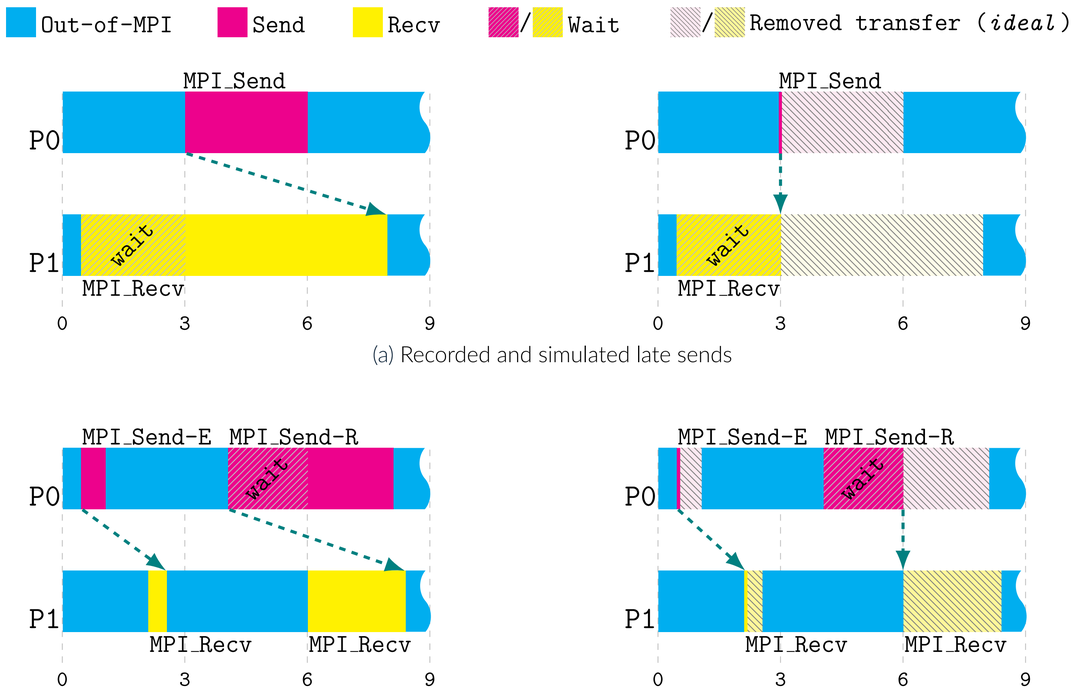


Figure 1. Performance analysis workflow[1]: Time to report may include many filter cycles before visualisation is possible.

- Fast identification of the *FoA*s facilitates streamlined performance analysis.
- Trace visualisation is the most reliable way for FoA identification.
- Trace size is the critical bottleneck to visualisation.

Ideal Network Simulation

- An ideal network has **zero latency** and infinite bandwidth.
- **Causality** dictates that receive directives cannot finish before the corresponding sends are posted.
- The simulation minimises the MPI durations as much as possible without violating the causality.
- **Wait times** inside the MPI are accounted for with causality.



(a) Recorded and simulated late sends

(b) Recorded and simulated late receives

Figure 2. Demonstrating the ideal network behaviour of late posts. Arrows run from the start of send directives to the end of receive directives. Late receives cause `MPI_Send` to choose between the *eager*(`E`) and the *rendezvous*(`R`) protocols.

- **Ideal network clocks** run outside the MPI runtime.
- At each MPI function exit, the values are compared and updated to retain causality such that
  - early receives always wait for late sends;
  - early sends either finish instantly for **eager** protocol or wait for the receive for **rendezvous**;

Metric of Interest - Transfer Efficiency

Transfer Efficiency [2] characterises the network efficiency for inter-process messages:

$$Trf = \frac{execution\ time_{ideal\ n/w}}{execution\ time_{observed}} = \frac{t_{simulated}}{t_{observed}} \tag{1}$$

- $execution_{ideal n/w}$ is calculated by replaying a trace on the hypothetical ideal network.
- The data-**transfer** durations are **eliminated**.
- **Wait durations** inside the MPI runtime **persist** on an ideal network.
- **FoA** regions have different transfer efficiencies.

Paraver, Dimemas, and Basic Analysis

- **Paraver** [3] is a trace-file format.
- **Dimemas** [4] emulates various linear characteristics.
- The Python script **Basic Analysis** [5] runs Dimemas on traces with Paraver format, where it
  - emulates a *zero-latency* and *infinite bandwidth* network characteristics;
  - generates a simulated trace with the same MPI events as the original.
  - The simulated timestamps reflect the removal of the transfer durations.
- The proposed method uses both the original and the simulated traces.

Contribution

Performance metrics like transfer efficiency characterise a complete execution.

- This work monitors the evolution of the transfer-efficiency metrics along the timeline.
- The monitors are calculated from the timings of the original and the simulated traces.
- The monitors show the varying local transfer efficiencies that indicate different FoAs.

Monitoring Transfer Efficiency

- The application timeline is conceived as consecutive pairs of compute and MPI regions.

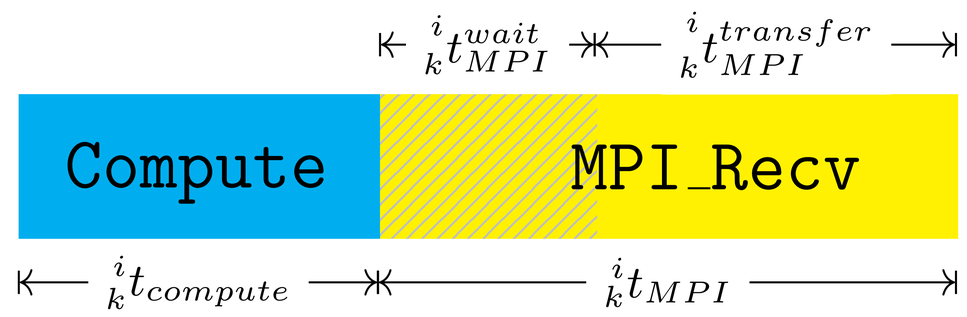


Figure 3. Resolving i^{th} compute region of MPI rank-k followed by an MPI region (`MPI_Recv`). The MPI regions include both wait ($^{i}_{k}t^{wait}_{MPI}$) and transfer ($^{i}_{k}t^{transfer}_{MPI}$) durations.

- The local transfer efficiency monitor is defined for each pair. For the i^{th} pair,

$$^{i}_{k}Trf_{local} = \frac{^{i}_{k}t_{simulated}}{^{i}_{k}t_{observed}} = \frac{^{i}_{k}t_{compute} + {}^{i}_{k}t^{wait}_{MPI}}{^{i}_{k}t_{compute} + {}^{i}_{k}t^{wait}_{MPI} + {}^{i}_{k}t^{transfer}_{MPI}} \tag{2}$$

  This acquires the local behaviours but may exhibit drastically different values as artefacts for consecutive pairs with significantly different durations.
- To avoid the local artefacts, a cumulative monitor at the i^{th} pair is defined:

$$^{i}_{k}Trf_{cumulative} = \left(\sum_{j=1}^{i} {}^{j}_{k}t_{simulated}\right) \Big/ \left(\sum_{j=1}^{i} {}^{j}_{k}t_{observed}\right) \tag{3}$$

- Causal connection among all ranks through message passing propagates the wait times included in the ideal network clocks. *Hence, monitoring one rank is sufficient for our purpose.*

Evaluation

- A 2D 5-point stencil heat-equation code with 4 different execution patterns

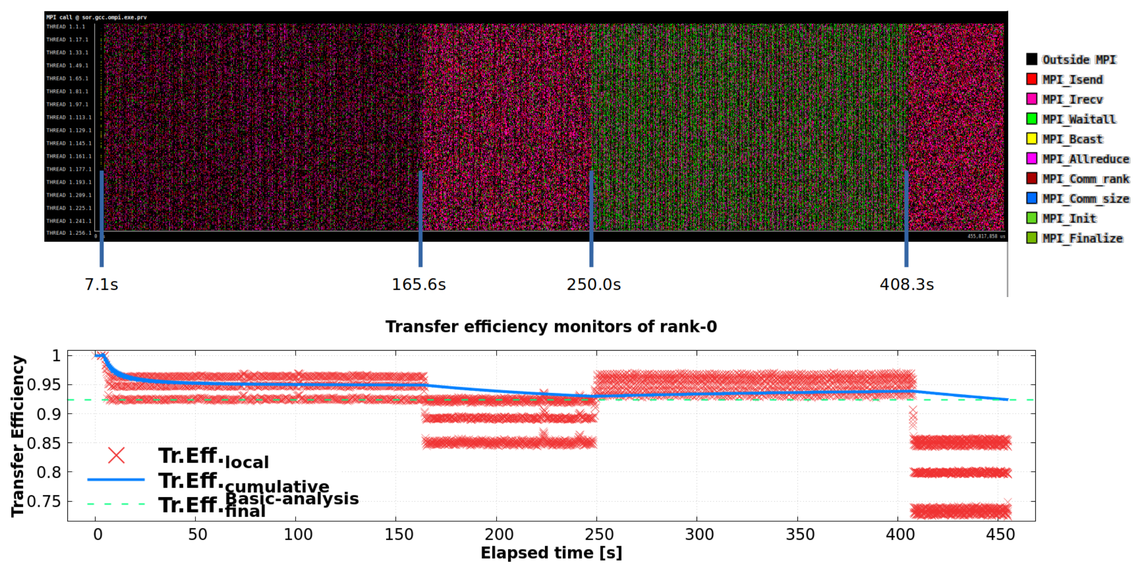


Figure 4. Timeline visualisation with the proposed local and cumulative monitors helping to identify the different execution phases precisely with their start/end times. The slope difference between two FoA regions decreases with increasing elapsed time. Limited pixel space results in seemingly multiple values at a given timestamp.

Outlook

- The monitors quickly identify the FoAs with useful previews of localised behaviours.
- We will refine the monitors to tend to the drastic changes in local trends.
- We will extend the monitoring for shared memory paradigms.

Related Works

- MSA [6] with density-based clustering groups and identify similar compute durations.
- Spectral analysis [7] identifies execution structures to pick a statistically representative region.

References

[1] M. Garcia-Gasulla, "POP methodology." `https://www.vi-hps.org/cms/upload/material/tw42/POP_methodology.pdf`, May 2022.

[2] C. Rosas, J. Giménez, and J. Labarta, "Scalability prediction for fundamental performance factors," *Supercomput. Front. Innov.: Int. J.*, vol. 1, p. 4–19, July 2014.

[3] V. Pillet, J. Labarta, A. Cortes, and S. Girona, "Paraver: A tool to visualize and analyze parallel code," in *World occam and Transputer User Group Technical Meeting*, p. 17, 04 1995.

[4] J. Labarta, S. Girona, and T. Cortes, "Analyzing scheduling policies using dimemas," *Parallel Computing*, vol. 23, no. 1, pp. 23–34, 1997. Environment and tools for parallel scientific computing.

[5] B. S. Center, "Basicanalysis." `https://github.com/bsc-performance-tools/basicanalysis`, May 2018.

[6] J. Gonzalez, J. Gimenez, and J. Labarta, "Performance analytics: Understanding parallel applications using cluster and sequence analysis," in *Tools for High Performance Computing 2013* (A. Knüpfer, J. Gracia, W. E. Nagel, and M. M. Resch, eds.), (Cham), pp. 1–17, Springer International Publishing, 2014.

[7] M. Casas, R. M. Badia, and J. Labarta, "Automatic phase detection and structure extraction of mpi applications," *Int. J. High Perform. Comput. Appl.*, vol. 24, p. 335–360, Aug. 2010.

This work has received funding from the German Federal Ministry of Education and Research (grant number 16ME0584).

SPONSORED BY THE Federal Ministry of Education and Research