HLRS

High-Performance
Computing Center
Stuttgart

Kingshuk Haldar

Trace-based, time-resolved analysis of MPI application performance using standard metrics

Visualise with `ClockTalk` how the performance metrics evolve

Outline

Background

Scope

Trace-based, time-resolved analysis of **MPI application** performance using standard metrics

- Trace based analyses
- MPI-only applications

Trace-based performance analysis workflow

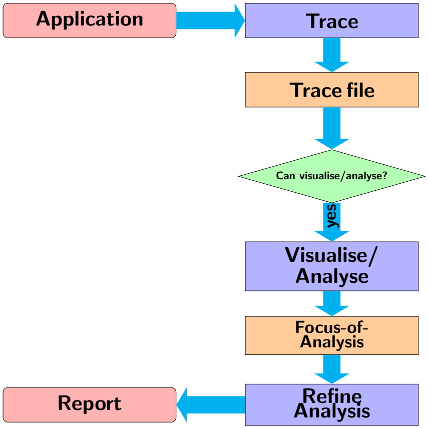

Figure 1: Analysis workflow outline.

Trace-based performance analysis workflow

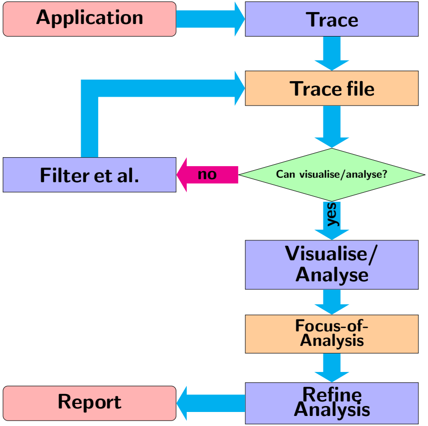

Figure 1: Analysis workflow outline.

What to do when the trace is large

Challenges and opportunities

Recipes

Recipes

- Complain a bit to your fellow analysts

Recipes

- Complain a bit to your fellow analysts
- Digress from detailed trace analysis - generate aggregated profiles

Recipes

- Complain a bit to your fellow analysts

- Digress from detailed trace analysis - generate aggregated profiles
  - Initialisation/finalisation phases may dominate the calculation
  - Function-level resolution is possible, not further
  - Miss out time-local characteristic if a metric seems good overall

Recipes

- Complain a bit to your fellow analysts
- Digress from detailed trace analysis - generate aggregated profiles
  - Initialisation/finalisation phases may dominate the calculation
  - Function-level resolution is possible, not further
  - Miss out time-local characteristic if a metric seems good overall
- Estimate whether applying information-filters would help

Recipes

- Complain a bit to your fellow analysts
- Digress from detailed trace analysis - generate aggregated profiles
  - Initialisation/finalisation phases may dominate the calculation
  - Function-level resolution is possible, not further
  - Miss out time-local characteristic if a metric seems good overall
- Estimate whether applying information-filters would help
  - It might not - the trace might still be quite large

Recipes

- Complain a bit to your fellow analysts
- Digress from detailed trace analysis - generate aggregated profiles
  - Initialisation/finalisation phases may dominate the calculation
  - Function-level resolution is possible, not further
  - Miss out time-local characteristic if a metric seems good overall
- Estimate whether applying information-filters would help
  - It might not - the trace might still be quite large
- Temporal filters

Recipes

- Complain a bit to your fellow analysts
- Digress from detailed trace analysis - generate aggregated profiles
  - Initialisation/finalisation phases may dominate the calculation
  - Function-level resolution is possible, not further
  - Miss out time-local characteristic if a metric seems good overall
- Estimate whether applying information-filters would help
  - It might not - the trace might still be quite large
- Temporal filters
  - Chop time segments and analyse separately - guided (recompiled with user-defined instrumentations) or unguided (trial-and-error)

Recipes

- Complain a bit to your fellow analysts
- Digress from detailed trace analysis - generate aggregated profiles
  - Initialisation/finalisation phases may dominate the calculation
  - Function-level resolution is possible, not further
  - Miss out time-local characteristic if a metric seems good overall
- Estimate whether applying information-filters would help
  - It might not - the trace might still be quite large
- Temporal filters
  - Chop time segments and analyse separately - guided (recompiled with user-defined instrumentations) or unguided (trial-and-error)

Analysts make it work by combining the above techniques and applying expertise

Recourse: what do we already have

Recourse: what do we already have

- Robust metrics to characterise the execution

Recourse: what do we already have

- Robust metrics to characterise the execution

$$\underbrace{\frac{\sum_r (t^r_{\mathrm{OOM}})/|\mathcal{R}|}{t_{\mathrm{execution}}}}_{\text{Parallelisation efficiency}} = \underbrace{\frac{\sum_r (t^r_{\mathrm{OOM}})/|\mathcal{R}|}{\max(t^r_{\mathrm{OOM}})}}_{\text{Load Balance}} \times \underbrace{\frac{\max(t^r_{\mathrm{OOM}})}{t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}}}_{\text{Serialisation}} \times \underbrace{\frac{t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}}{t_{\mathrm{execution}}}}_{\text{Transfer efficiency}} \tag{1}$$

$\mathcal{R}$ is the set of all ranks; $r \in \mathcal{R}$
t_{OOM}: out-of-MPI durations
$t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}$: execution time assuming ideal network

Recourse: what do we already have

HLRS

- Robust metrics to characterise the execution

$$\underbrace{\frac{\sum_r (t^r_{\mathrm{OOM}})/|\mathcal{R}|}{t_{\mathrm{execution}}}}_{\text{Parallelisation efficiency}} = \underbrace{\frac{\sum_r (t^r_{\mathrm{OOM}})/|\mathcal{R}|}{\max(t^r_{\mathrm{OOM}})}}_{\text{Load Balance}} \times \underbrace{\frac{\max(t^r_{\mathrm{OOM}})}{t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}}}_{\text{Serialisation}} \times \underbrace{\frac{t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}}{t_{\mathrm{execution}}}}_{\text{Transfer efficiency}} \quad (1)$$

$\mathcal{R}$ is the set of all ranks; $r \in \mathcal{R}$
t_{OOM}: out-of-MPI durations
$t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}$: execution time assuming ideal network

- The model factors quantify the different bottleneck categories
- Applies to entire or part of the execution duration

Recourse: what do we already have

- Robust metrics to characterise the execution

$$\underbrace{\frac{\sum_r (t^r_{\mathrm{OOM}})/|\mathcal{R}|}{t_{\mathrm{execution}}}}_{\text{Parallelisation efficiency}} = \underbrace{\frac{\sum_r (t^r_{\mathrm{OOM}})/|\mathcal{R}|}{\max(t^r_{\mathrm{OOM}})}}_{\text{Load Balance}} \times \underbrace{\frac{\max(t^r_{\mathrm{OOM}})}{t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}}}_{\text{Serialisation}} \times \underbrace{\frac{t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}}{t_{\mathrm{execution}}}}_{\text{Transfer efficiency}} \tag{1}$$

$\mathcal{R}$ is the set of all ranks; $r \in \mathcal{R}$
t_{OOM}: out-of-MPI durations
$t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}$: execution time assuming ideal network

  - The model factors quantify the different bottleneck categories
  - Applies to entire or part of the execution duration

- Isolate time intervals and calculate performance metrics

Recourse: what do we already have

- Robust metrics to characterise the execution

$$\underbrace{\frac{\sum_r (t^r_{\mathrm{OOM}})/|\mathcal{R}|}{t_{\mathrm{execution}}}}_{\text{Parallelisation efficiency}} = \underbrace{\frac{\sum_r (t^r_{\mathrm{OOM}})/|\mathcal{R}|}{\max(t^r_{\mathrm{OOM}})}}_{\text{Load Balance}} \times \underbrace{\frac{\max(t^r_{\mathrm{OOM}})}{t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}}}_{\text{Serialisation}} \times \underbrace{\frac{t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}}{t_{\mathrm{execution}}}}_{\text{Transfer efficiency}} \tag{1}$$

$\mathcal{R}$ is the set of all ranks; $r \in \mathcal{R}$
t_{OOM}: out-of-MPI durations
$t^{\mathrm{ideal\ n/w}}_{\mathrm{execution}}$: execution time assuming ideal network

  - The model factors quantify the different bottleneck categories
  - Applies to entire or part of the execution duration

- Isolate time intervals and calculate performance metrics
  - Indicate the performance characteristics within that duration

Trace analysis frameworks

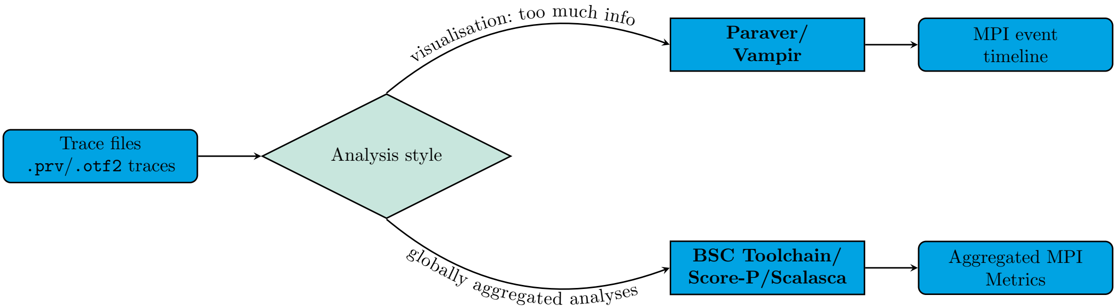

Figure 2: Conceptual placement of the analyses/visualisation frameworks.

Trace analysis frameworks

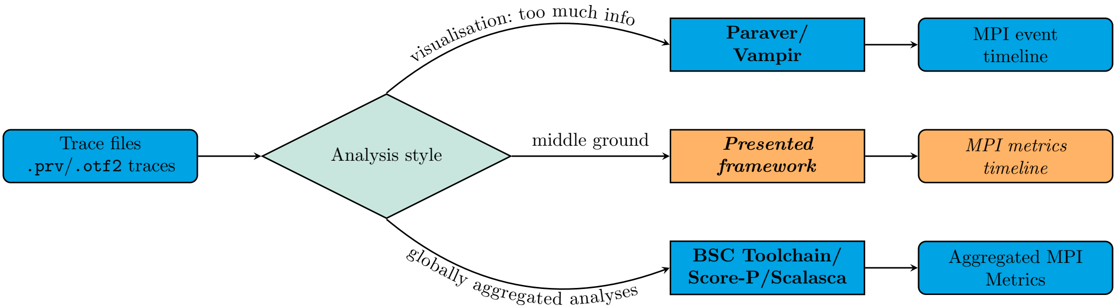

Figure 2: Conceptual placement of the analyses/visualisation frameworks.

Metrics and their scopes

Capturing the time-varying performance characteristics

Refining scopes of metrics

- Calculating model factors of an MPI application ran for 456*s*

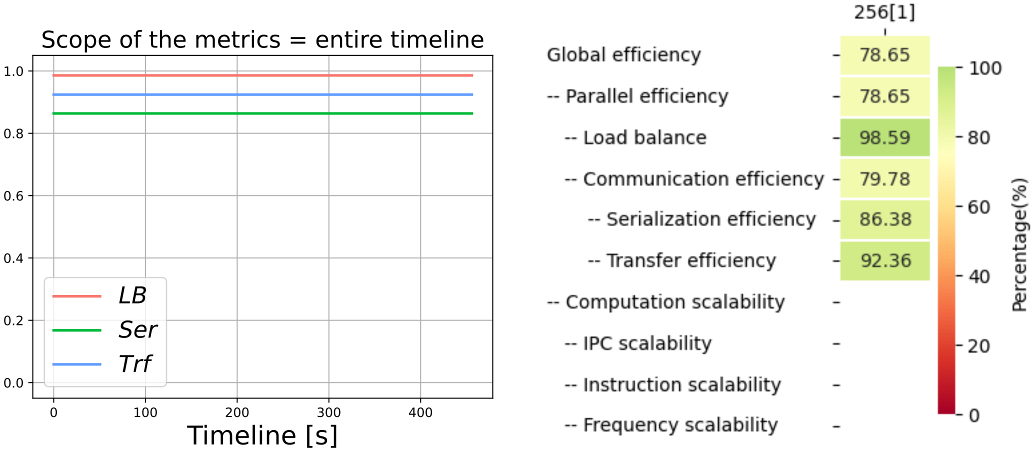

Figure 3: Scope refinement brings out the time-varying characteristics of the metrics.

Refining scopes of metrics

- Calculating model factors of an MPI application ran for 456*s*

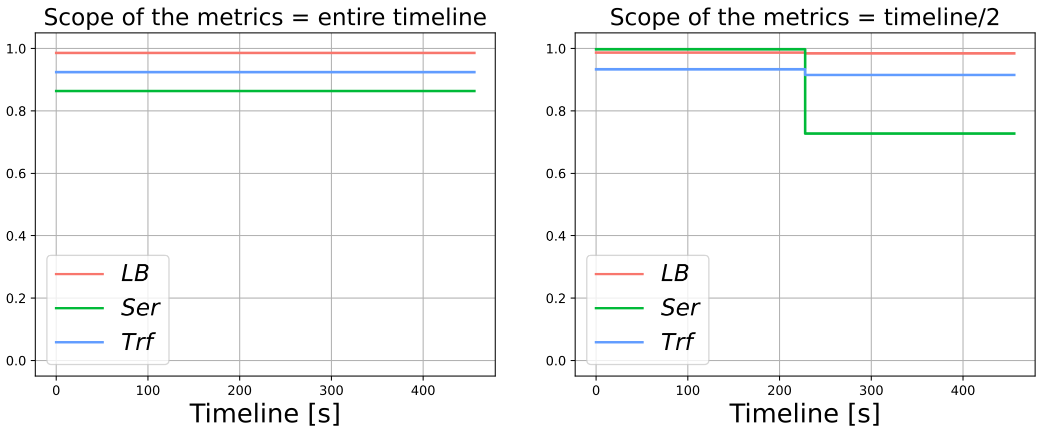

Figure 3: Scope refinement brings out the time-varying characteristics of the metrics.

Refining scopes of metrics

- Calculating model factors of an MPI application ran for 456s

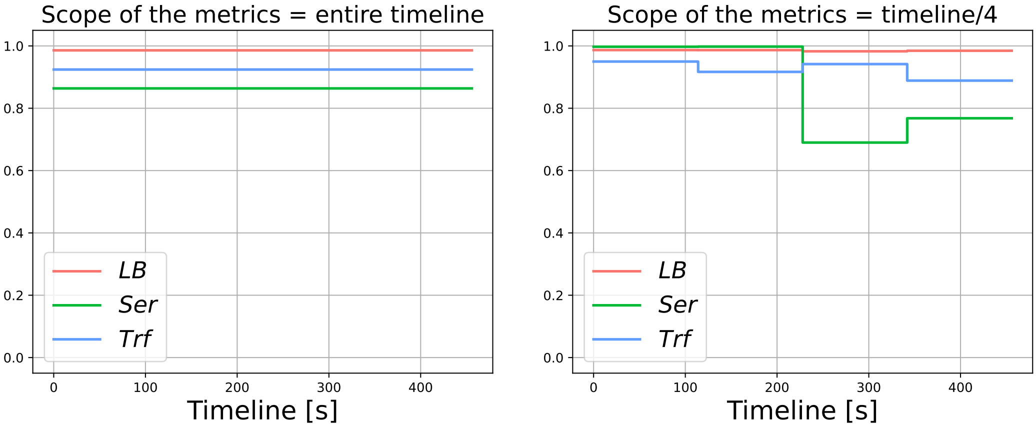

Figure 3: Scope refinement brings out the time-varying characteristics of the metrics.

Refining scopes of metrics

- Calculating model factors of an MPI application ran for 456s

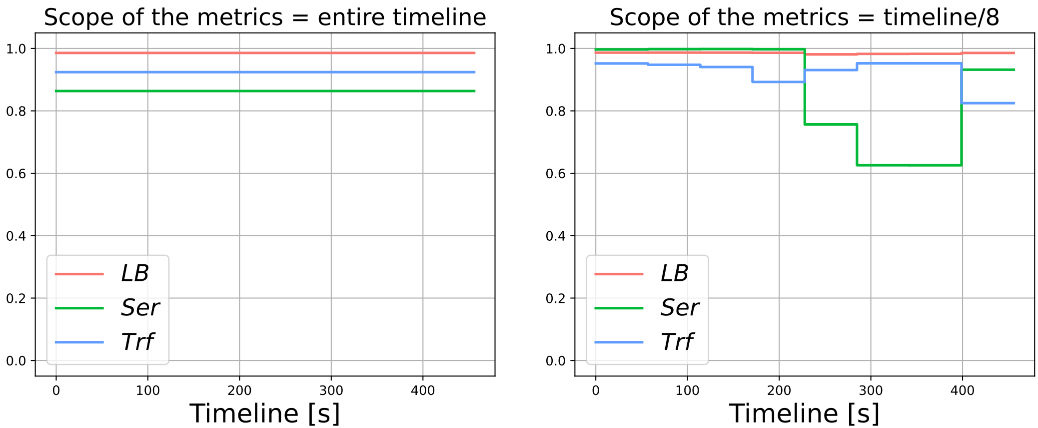

Figure 3: Scope refinement brings out the time-varying characteristics of the metrics.

Extent of scope refinement

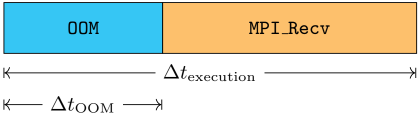

Figure 4: An out-of-MPI (OOM) region followed by an MPI region of an arbitrary rank.

- Compute load $\leftarrow$ the out-of-MPI regions

Extent of scope refinement

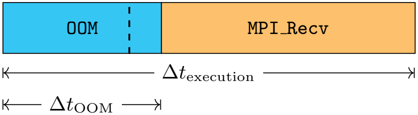

Figure 4: Refinement boundary passes through the OOM region.

- Left of the boundary has 100% computation - not meaningful
- Load balance: every rank requires at least one out-of-MPI region

Extent of scope refinement

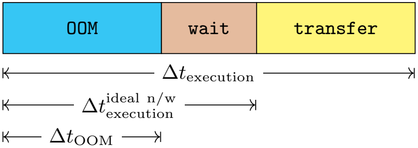

Figure 4: MPI region is now resolved into a wait and a transfer regions.

- Ideal-network constrained analysis finds wait and transfer durations
- Calculating serialisation and transfer efficiencies require resolving MPI regions

Extent of scope refinement

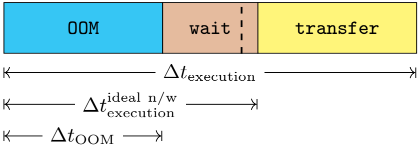

Figure 4: Refinement boundary passes through an estimated wait region.

- Entire MPI region on the left is blamed to serialisation

Extent of scope refinement

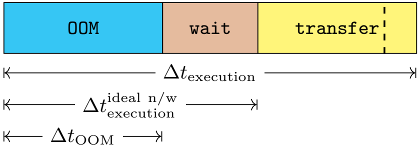

Figure 4: Refinement boundary passes through an estimated transfer region.

- Partial capture of the transfer efficiency

Consequences of metrics' scope refinement

Consequences of metrics' scope refinement

- Refining scopes of the metrics reveals the substructures - time-local performance characteristics

Consequences of metrics' scope refinement

- Refining scopes of the metrics reveals the substructures - time-local performance characteristics
- Arbitrary refinement is not possible - discretisation effect
  - Minimum block for all ranks: at least one out-of-MPI and the following MPI region in a scope- *three* events
  - Increasing this threshold expands this scope.

Consequences of metrics' scope refinement

- Refining scopes of the metrics reveals the substructures - time-local performance characteristics

- Arbitrary refinement is not possible - discretisation effect
  - Minimum block for all ranks: at least one out-of-MPI and the following MPI region in a scope- *three* events
  - Increasing this threshold expands this scope.

- The ideal-network execution includes wait durations, but not transfer
  - Its changing values are calculated at every event point for all ranks

Consequences of metrics' scope refinement

- Refining scopes of the metrics reveals the substructures - time-local performance characteristics
- Arbitrary refinement is not possible - discretisation effect
  - Minimum block for all ranks: at least one out-of-MPI and the following MPI region in a scope- *three* events
  - Increasing this threshold expands this scope.
- The ideal-network execution includes wait durations, but not transfer
  - Its changing values are calculated at every event point for all ranks
- MPI regions completing multiple requests require special treatment

Modelling wait and transfers within MPI

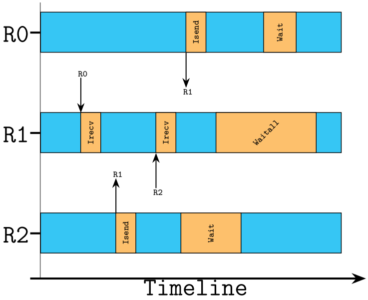

Figure 5: MPI regions recorded with PMPI.

- Without ideal-network constrained calculation wait/transfer regions cannot be resolved

Modelling wait and transfers within MPI

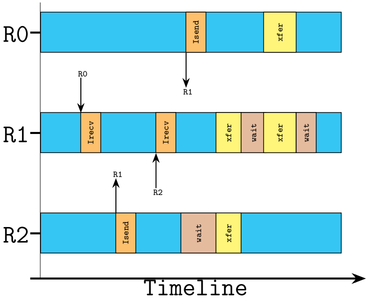

Figure 5: MPI regions resolved into wait and transfer regions.

- Ideal-network constrained calculation brings out the wait/transfer regions

Modelling wait and transfers within MPI

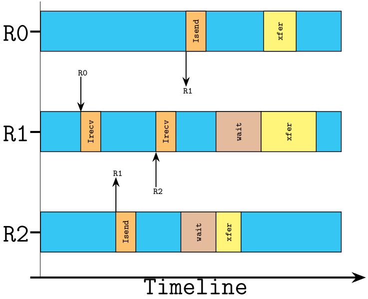

Figure 5: Wait and transfer regions are aggregated within each MPI region.

- Modelled *wait-then-transfer* sequence in each MPI duration simplifies the ideal-network constrained calculation

Evolving clocks

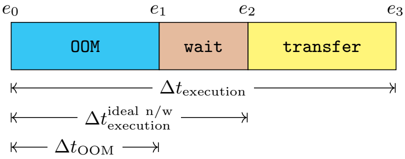

Figure 6: Evolution of clocks.

For calculation, three clocks or timers are defined per rank:

Clock description	Value at e_0	Value at e_1	Value at e_2	Value at e_3
Execution/elapsed	$t^0_{\text{execution}}$	$t^0_{\text{execution}} + \Delta t_{\text{OOM}}$	$t^0_{\text{execution}} + \Delta t^{\text{ideal n/w}}_{\text{execution}}$	$t^0_{\text{execution}} + \Delta t_{\text{execution}}$
Out-of-MPI	t^0_{OOM}	$t^0_{\text{OOM}} + \Delta t_{\text{OOM}}$	$t^0_{\text{OOM}} + \Delta t_{\text{OOM}}$	$t^0_{\text{OOM}} + \Delta t_{\text{OOM}}$
Ideal execution	$t^0_{\text{ideal n/w}}$	$t^0_{\text{ideal n/w}} + \Delta t_{\text{OOM}}$	$t^0_{\text{ideal n/w}} + \Delta t^{\text{ideal n/w}}_{\text{execution}}$	$t^0_{\text{ideal n/w}} + \Delta t^{\text{ideal n/w}}_{\text{execution}}$

Table 1: Various clocks run during different durations.

Interpolation of the clocks' evolution

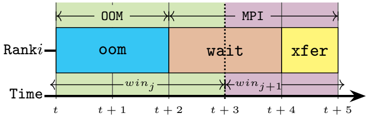

Figure 7: Discretisation boundary passing through a minimum block.

- For almost all ranks, an arbitrary discretisation boundary will cut a minimum block
- An interpolation scheme to project contribution of broken minimum blocks is required

Interpolation of the clocks' evolution

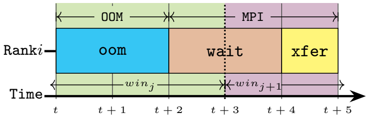

Figure 7: Discretisation boundary passing through a minimum block.

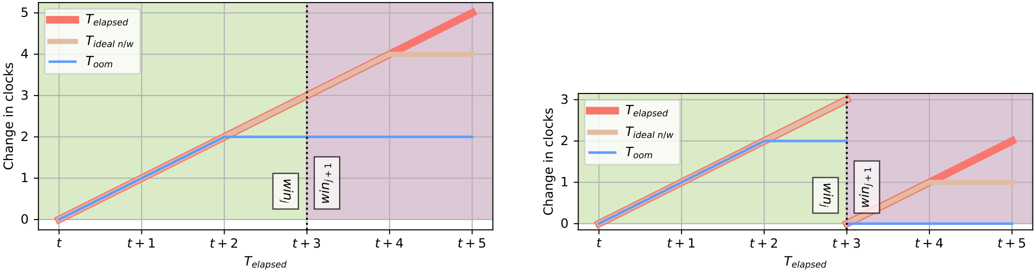

Figure 8: Contribution of the clocks' evolution to each window.

Timeline discretisation framework for time-resolved metrics

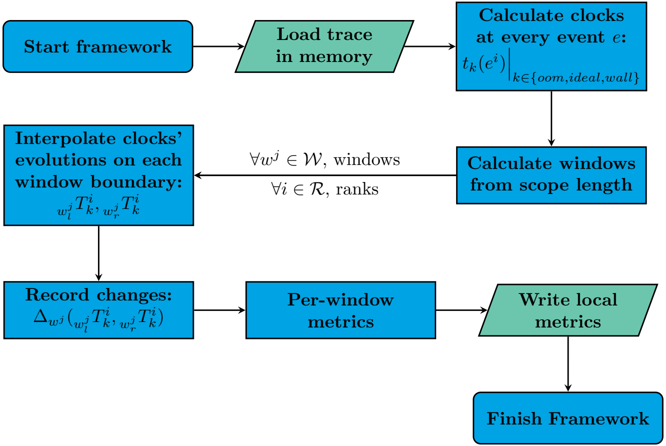

Figure 9: Discretisation framework: ideal-network calculation is a prerequisite for fully resolved metrics.

Contribution

- Timeline discretisation framework for time-scoped MPI metrics - for a given scope
  - Interpolate the clock evolution on the scope/discretisation boundaries
- An MPI-only Paraver trace analyser that calculates ideal-network constrained clock values- ClockTalk
  - Stores the ideal-network and out-of-MPI clock values internally at each event point
- An automated timeline discretisation framework from the stored clocks' evolution for the entire execution
- Case studies showing how time-resolved metrics can help with analysis

Case studies

Extracting time-resolved metrics to aid analyses

Benchmarking stencil application

- A 256-rank stencil application ran for 456s

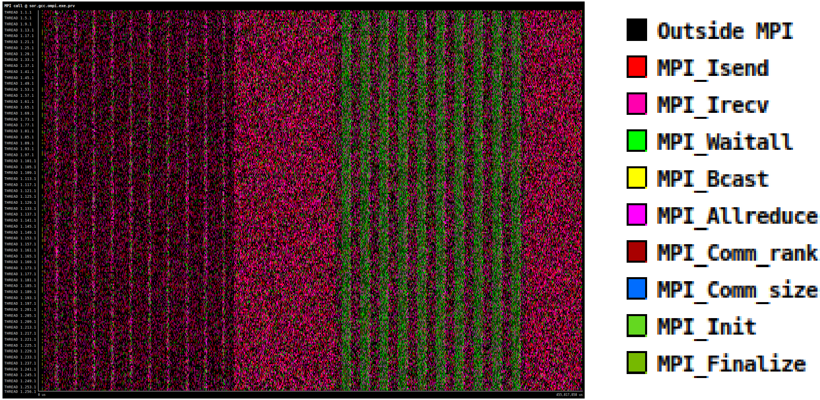

Figure 10: Timeline view of MPI application with four regions with different parallel efficiencies.

Benchmarking stencil application: Time-resolved metrics

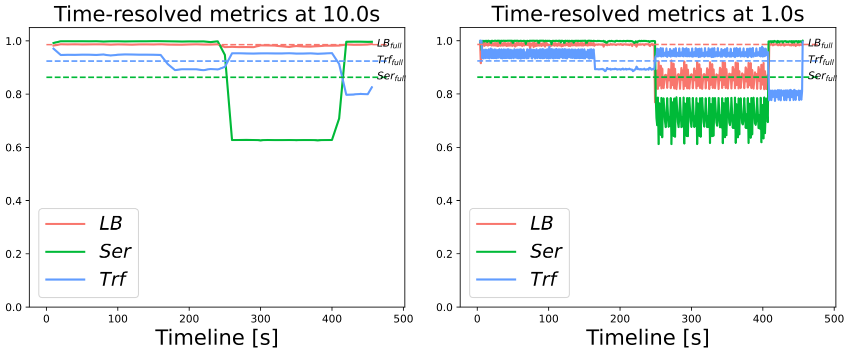

Figure 11: Time-resolved metrics for two different resolutions. Dashed values are the metrics of the entire duration (from Basic Analysis).

- Larger resolutions provide a smoother representation of the metrics' evolution
- Smaller resolutions bring out even finer compute-communication interaction detail

Real-world MD application (`ls1-MarDyn`)

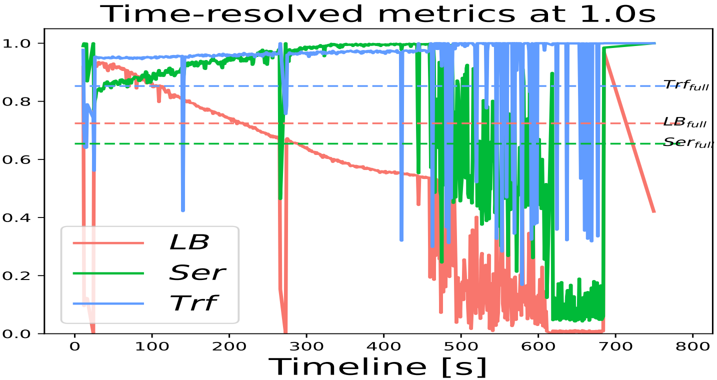

Figure 12: Time-resolved metrics of `ls1-MarDyn` shows the gradual degradation of compute-load balance.

- Trace-size: 465 GB, 2048 MPI processes, 10 minutes cut-off, 50 minutes processing time
  - Existing toolset does not finish in a realistic time
  - Visualising the gradual increase of the compute imbalance was not possible before

Comparing against BSC toolchain

Use case	Trace size [GB]	Analysis time [s]	
		BSC toolchain	ClockTalk
Bench app.	1.3	554.50	9.06
PETSc-1	1.4	905.21	17.38
PETSc-2	3.8	1459.15	44.66
ls1-MarDyn	465.0	Time out	2950.47

Table 2: Comparing analysis times of BSC toolchain and ClockTalk at login-nodes of Hawk.

- Unlike the BSC tools, ClockTalk does not generate any simulated trace
- Skipping the file I/O, speeds up ClockTalk a lot

github/ClockTalk

Summary and Outlook

Summary

- Quick analysis - for initial overview that is reliable even for larger traces
- Better performance overview than uni-valued metrics for the entire execution
- Duration of interest can be isolated - guided even without user-defined region markers (no multiple trial-and-error cycle)
- Clock evolution and interpolation can be readily applied to other programming paradigms

Future outlook

- Enhance trace-analyser for shared-memory paradigm
- Better identification of good discretisation length
- Generate ideal-network trace from store per-event clocks

Comments are welcome

Effects of threshold per window

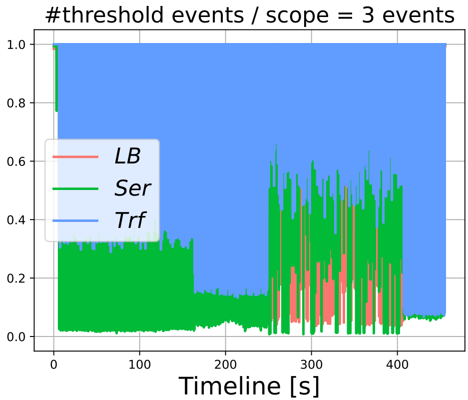

Figure 13: Number of threshold events affect metrics profile.

Effects of threshold per window

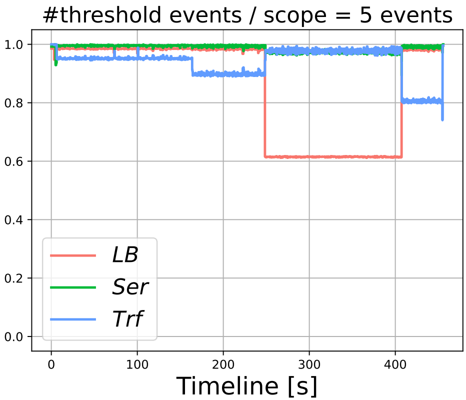

Figure 13: Number of threshold events affect metrics profile.[1]

[1] Kevin A. Huck and Jesus Labarta. "Detailed Load Balance Analysis of Large Scale Parallel Applications". In: *2010 39th International Conference on Parallel Processing*. 2010, pp. 535–544. DOI: 10.1109/ICPP.2010.61.

Effects of threshold per window

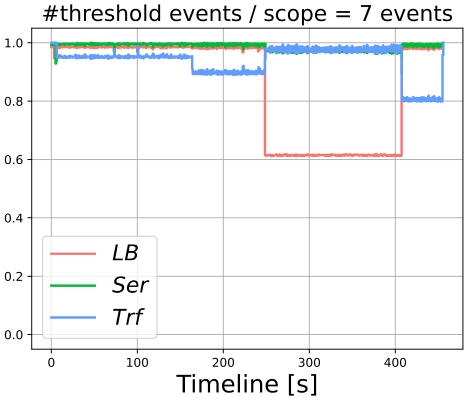

Figure 13: Number of threshold events affect metrics profile.

Effects of threshold per window

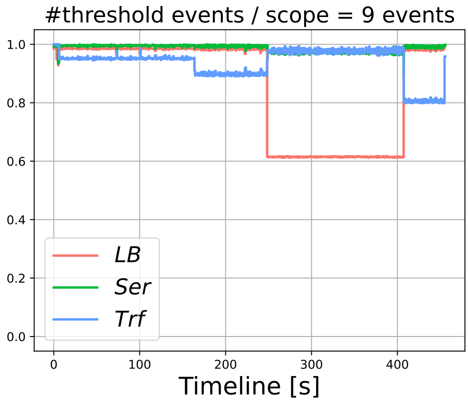

Figure 13: Number of threshold events affect metrics profile.

Effects of threshold per window

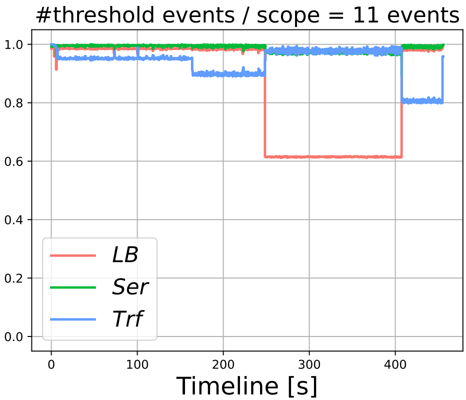

Figure 13: Number of threshold events affect metrics profile.

Effects of threshold per window

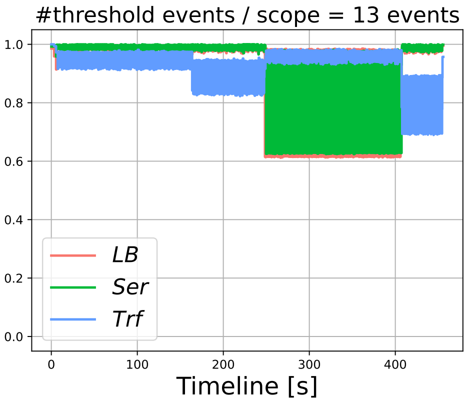

Figure 13: Number of threshold events affect metrics profile.

Effects of threshold per window

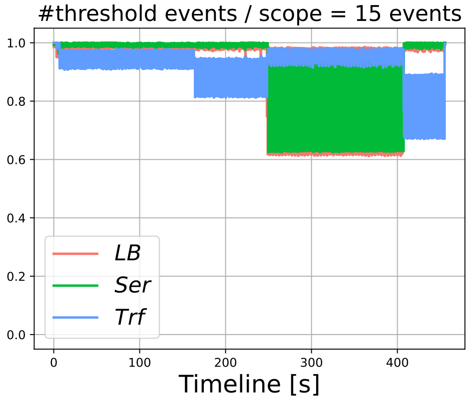

Figure 13: Number of threshold events affect metrics profile.

Effects of threshold per window

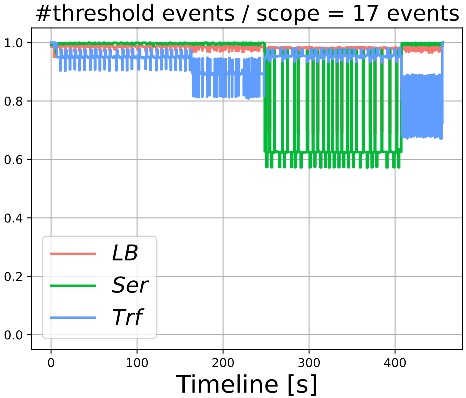

Figure 13: Number of threshold events affect metrics profile.

Effects of threshold per window

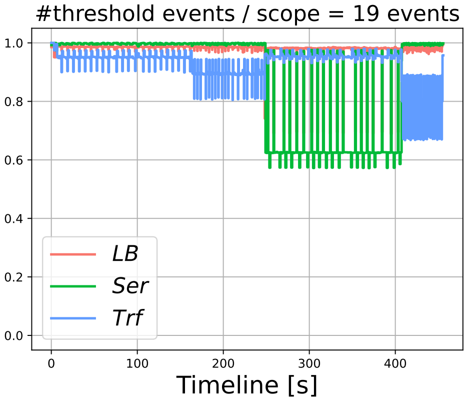

Figure 13: Number of threshold events affect metrics profile.

Effects of threshold per window

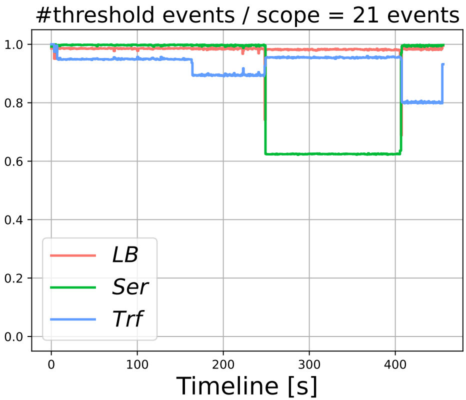

Figure 13: Number of threshold events affect metrics profile.