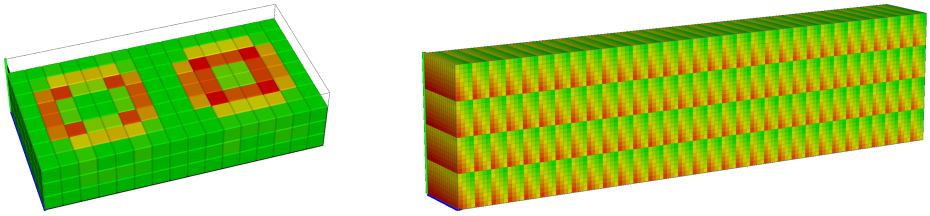
Visualizing Performance Data With Respect to the Simulated Geometry

Tom Vierjahn[1,3], Torsten W. Kuhlen[1,3],
Matthias S. Müller[2,3], and Bernd Hentschel[1,3]

[1] Visual Computing Institute, RWTH Aachen University, Germany
[2] Chair for High-Performance Computing, RWTH Aachen University, Germany
[3] JARA – High-Performance Computing, Germany



(a) Zeus-MP/2 (b) Sweep3D

Fig. 1: Visualizing performance in an HPC system's network topology may reveal the underlying geometry in small cases (a), but obscures it in larger cases (b).

Abstract. Understanding the performance behaviour of high-performance computing (HPC) applications based on performance profiles is a challenging task. Phenomena in the performance behaviour can stem from the HPC system itself, from the application's code, but also from the simulation domain. In order to analyse the latter phenomena, we propose a system that visualizes profile-based performance data in its spatial context in the simulation domain, i.e., on the geometry processed by the application. It thus helps HPC experts *and* simulation experts understand the performance data better. Furthermore, it reduces the initially large search space by automatically labelling those parts of the data that reveal variation in performance and thus require detailed analysis.

1 Introduction

Optimizing an application to use the compute power efficiently that is offered by a modern high-performance computing (HPC) system requires powerful tools for performance analysis. Such tools ought to reveal an application's performance behaviour clearly. Visualizing performance data with respect to the HPC

system's network topology (Fig. 1) helps analysts understand an application's performance behaviour. However, it does not clearly reveal performance phenomena in the simulation domain that are for instance caused by sub-optimal domain decompositions.

In simple cases, an analyst might be able to infer the simulation domain from the visualization in the HPC system's network domain: in case of Fig. 1(a) a sphere, cut in half and then mapped to the HPC system's nodes. For larger cases the simulation domain can hardly be inferred: in case of Fig. 1(b) a 2D gradient, cut several times. Thus, in order to provide complete insight into an application's performance behaviour, analysis tools ought to visualize the available data also with respect to the system domain. Only few tools take this into account, and if they do they restrict it, e.g., to regular grids.

Therefore, we propose a tool that facilitates visualizing performance data in its spatial context on arbitrary geometry in the simulation domain, e.g., triangle meshes, provided there is a mapping from the computing resources to the geometry. In order to help the analyst find meaningful views on the data, the tool automatically identifies and suggests views that reveal variation in performance.

2 Related Work

Isaacs et al. give an overview of the state of the art in performance visualization [3]. They list only few techniques considering the simulation domain. Schulz et al. stress the importance of taking the simulation domain into account during performance analysis [4]. Wylie and Geimer use Cartesian grids [5] in the Cube performance profile browser [2] in order to visualize performance with respect to the simulation domain.

We propose a tool that is similar in spirit to the Cube performance profile browser but it enables visualizing performance data on arbitrary geometry.

3 Nomenclature: Performance Profiles, Severity Views

Profiling is a common technique in performance analysis. A profile summarizes performance data over an application's complete run-time. Data is collected according to *performance metrics* $m \in \mathcal{M}$, e.g., execution time, for the *call paths* $c \in \mathcal{C}$ of the application's functions executed on the *system resources* $s \in \mathcal{S}$, i.e., processes or threads. During analysis, by selecting a pair of metric m and call path c, analysts specify a *severity view*

$$v_{m,c} : \mathcal{S} \mapsto \mathbb{R} \quad ,$$

with $v_{m,c}(s)$ yielding the severity of, e.g., execution time, for a user-selected pair (m, c) on a system resource s.

Instead of analysing performance on a thread or process level, this work focuses on the individual MPI ranks $r_i \in \mathcal{S}_{\text{MPI}}$ that each execute a set $\mathcal{S}_{r_i} \subseteq \mathcal{S}$ of processes or threads, so that $\mathcal{S} = \bigcup_{r_i \in \mathcal{S}_{\text{MPI}}} \mathcal{S}_{r_i}$ and $\forall_i \forall_{j \neq i} \mathcal{S}_{r_i} \cap \mathcal{S}_{r_j} = \emptyset$.

Therefore, the severities measured for the individual processes or threads need to be aggregated in order to compute the severity for the i-th MPI rank r_i: $\sum_{s \in \mathcal{S}_{r_i}} v_{m,c}(s)$. With a slight abuse of notation we use the shorthand

$$v_{m,c}(r_i) := \sum\nolimits_{s \in \mathcal{S}_{r_i}} v_{m,c}(s) \quad ,$$

with $v_{m,c}(r_i)$ denoting the severity of, e.g., execution time, for a user-selected pair (m, c) on the i-th MPI rank r_i. Since we require the performance data to include a mapping from the MPI ranks to the individual parts of the geometry in the simulation domain, $v_{m,c}(r_i)$ also denotes the severity for the part of the geometry that is computed by the i-th MPI rank r_i.

4 Detecting Variation in the Data

Visualizing the severity $v_{m,c}(r_i)$ for the individual MPI ranks may provide valuable insight for finding root causes of performance bottlenecks. However, such a detailed visualization is only sensible if there is a certain amount of variation in performance across the MPI ranks. Otherwise, a single number representing the accumulated severity $\sum_{r_i \in \mathcal{S}_{\text{MPI}}} v_{m,c}(r_i)$ of the selected performance metric would do.

In order to identify large-variation severity views, our system uses the variation coefficient

$$q_{m,c} = \frac{\sigma_{m,c}}{\mu_{m,c}}$$

as an indicator. Here, $\mu_{m,c}$ denotes the mean severity of the MPI ranks in the selected severity view $v_{m,c}$, and $\sigma_{m,c}$ denotes the standard deviation, with

$$\mu_{m,c} = \frac{\sum_{r_i \in \mathcal{S}_{\text{MPI}}} v_{m,c}(r_i)}{|\mathcal{S}_{\text{MPI}}|} \quad \text{and} \quad \sigma_{m,c} = \frac{\sum_{r_i \in \mathcal{S}_{\text{MPI}}} \left(v_{m,c}(r_i) - \mu_{m,c}\right)^2}{|\mathcal{S}_{\text{MPI}}|} \quad .$$

According to the feedback provided by HPC experts, a threshold of $\tau_q = 0.01$ turned out to be sensible for detecting severity views of interest with $q_{m,c} \geq \tau_q$. However, τ_q can be adjusted by the analyst.

5 Interactive Visualizations

The proposed system provides several visualizations that have been developed according to requirements posed by HPC experts. These facilitate interactive analysis of profile-based performance data in a top-down fashion in order to find and analyse severity views of interest that reveal performance phenomena.

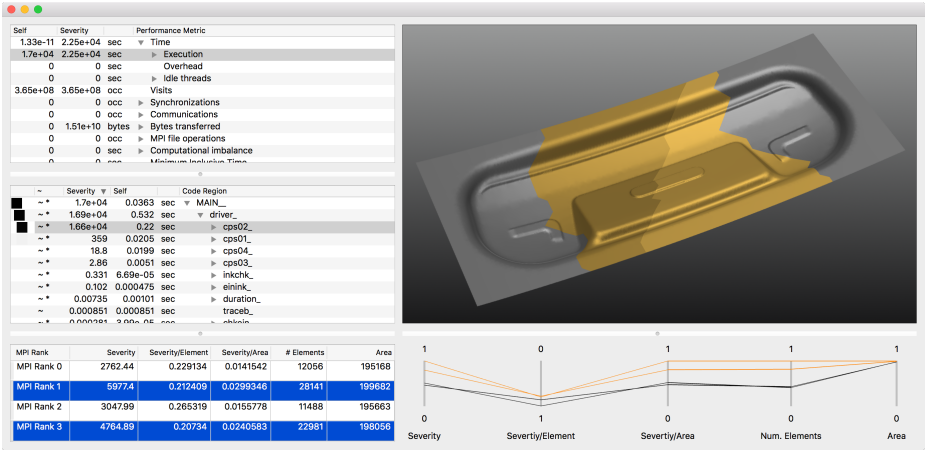


Fig. 2: Proposed user interface: the severity of "Time/Execution" in "cps02_" is visualized on the geometry (top right). The data from the table (lower left) is visualized by parallel coordinates (lower right). MPI ranks 1 and 3 are selected.

5.1 Performance Metrics and Call-Path Tree Widgets

The hierarchies of the metrics and call paths are visualized in tree widgets on the left of the user interface (Fig. 2). For each entry the total severity including the descendants (column "Severity") and the net severity of only the entry itself (column "Self") are printed. Both widgets can be sorted by total or net severity.

When the analyst selects a metric, the severities in the call-path tree widget are updated accordingly. When they select a pair of metric and call path, that severity view gets visualized in the remaining parts of the user interface. The columns "Severity" and "Self" can be swapped in any of the two tree widgets. The leftmost determines whether total or net severity is being visualized.

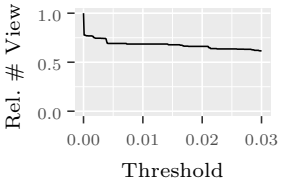
A glyph in the leftmost column of the call-path tree widget guides the analyst to the call paths with the largest severity by colour-encoding the severity relative to the respective parent's severity. The colour map can be user-defined. A linear black (100 % relative severity) to transparent (0 %) map is used by default. A tilde printed in the second column of the call-path tree widget indicates a large-variation severity view for detailed evaluation. If a severity view in the descendants of a call path exposes large variation, an asterisk is printed in order to speed up finding that severity view.

5.2 Visualizing Performance Data in its Spatial Context

The 3D viewport in the upper right of the user interface (Fig. 2) renders the geometry in the simulation domain. The severity for each MPI rank is visualized colour-coded on the respective part of the geometry. The colour map can be user-defined. A linear black (0 % severity) to light grey (100 %) map is used by

	# Views	Reduction
Non-Zero	48,031	—
Suggested	37,696	22 %

(a)



(b)

Fig. 3: Search space reduction: the table (a) lists the number of non-zero views and the number of suggested views ($\tau_q = 0.01$). The plot (b) presents the ratio of the number of suggested views to the number of non-zero views per threshold.

default. The simulation domain can be explored by moving a virtual camera with five degrees of freedom using keyboard and mouse. Elevation is limited to $\pm 90°$, and rotation around the viewing direction is locked in order to keep orientation intuitive and to prevent the analyst from losing track of the perspective.

The table in the lower left of the user interface lists the severities for each MPI rank alongside the properties of the associated part of the geometry, i.e., the number of finite elements and their surface area. In addition, the severity is related to these properties by, for instance, presenting the severity per surface area. A parallel coordinates plot in the lower right presents the same data as the table for a better, concise overview. This is particularly helpful for simulations using many MPI ranks. Each axis is normalized and can be flipped. That way the performance data can be inspected for meaningful structures when related to the geometry.

The views are linked: a geometry part or the associated MPI rank can be selected in the 3D view or the table. Selected geometry and the related information are then highlighted in all three views.

6 Results

We have preliminarily evaluated our system with performance data from a small sheet-metal forming simulation, executed on 4 thin nodes of SuperMUC (Phase 1) [1]. The performance data has been pre-processed using the remapping functionality of the Cube performance profile browser [2]. That way the performance measurements for actual computations get separated from those for MPI calls.

The automatic suggestion mechanism effectively sieves out those views that expose only low variation in performance: for $\tau_q = 0.01$ the search space is reduced by 22 % (Fig. 3). Larger thresholds filter out more views. However, since the variation coefficient is normalized to the mean severity, some high-variation, high-severity views might erroneously be filtered out.

In most of the simulation's functions MPI ranks 1 and 3 required most CPU-time. The data for the ranks even forms two almost separate classes (Fig. 2). The 3D visualization clearly points out that MPI ranks 1 and 3 are computing

high-detail parts of the geometry. With our tool, simulation experts were able to relate the observed performance phenomenon to a disadvantageous domain decomposition that did not consider the forming tool's shape.

7 Conclusion and Future Work

Our system helps analysts evaluate an HPC application's performance behaviour based on profiles by greatly reducing the search space: severity views that do not expose variation in performance are sieved out. Glyphs representing the severity of and labels indicating large-variation severity views quickly guide analysts down the application's call hierarchy towards *important* severity views. Relating the performance data to the simulation domain provides valuable insight. Our tool directed simulation experts to the domain decomposition as the cause for a performance phenomenon. However, tests with improved decompositions and significantly more compute nodes are left for future work.

Acknowledgements

This work has been partially funded by the German Federal Ministry of Research and Education (BMBF) under grant number 01IH13001D (Score-E).
This work has been partially funded by the Excellence Initiative of the German federal and state governments through the Jülich Aachen Research Alliance – High-Performance Computing.

References

1. SuperMUC petascale system, `https://www.lrz.de/services/compute/supermuc/systemdescription/`
2. Geimer, M., Saviankou, P., Strube, A., Szebenyi, Z., Wolf, F., Wylie, B.J.N.: Further improving the scalability of the Scalasca toolset. In: 10th Intl. Conf. Appl. Parallel and Scientific Computing (2012)
3. Isaacs, K.E., Giménez, A., Jusufi, I., Gamblin, T., Bhatele, A., Schulz, M., Hamann, B., Bremer, P.T.: State of the Art of Performance Visualization. In: EuroVis - STARs (2014)
4. Schulz, M., Levine, J.A., Bremer, P.T., Gamblin, T., Pascucci, V.: Interpreting performance data across intuitive domains. In: Proc. 40th Int. Conf. Parallel Process. (2011)
5. Wylie, B.J.N., Geimer, M.: Large-scale performance analysis of PFLOTRAN with Scalasca. In: Proc. 53rd Cray User Group meeting. Cray User Group Inc. (2011)