Authors' Electronic Version — Published in EuroVis 2016 - Posters
The definitive version is available at http://diglib.eg.org/

Geometry-Aware Visualization of Performance Data

T. Vierjahn, B. Hentschel, and T. W. Kuhlen

Visual Computing Institute, RWTH Aachen University, Germany, and JARA – High-Performance Computing, Germany

Abstract

Phenomena in the performance behaviour of high-performance computing (HPC) applications 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, i.e., on the geometry, in the simulation domain. It thus helps HPC experts but also simulation experts understand the performance data better. In addition, our tool reduces the initially large search space by automatically labelling large-variation views on the data which require detailed analysis.

Categories and Subject Descriptors (according to ACM CCS): D.2.8 [Software Engineering]: Metrics/Measurement—Performance measures, I.3.8 [Computer Graphics]: Applications—, I.6.9 [Simulation, Modeling, and Visualization]: Visualization—

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. These tools need to reveal an application's performance behaviour clearly. Although important aspects of the performance behaviour can be learned from the simulation domain, only few tools take this into account and if they do they restrict it, e.g., to regular grids.

We propose a tool that guides analysts towards important parts of profile-based performance data. The tool facilitates visualizing performance data in its spatial context on arbitrary geometry, e.g., triangle meshes, in the simulation domain, provided there is a mapping from the computing resources to the geometry.

2. Related Work

Isaacs et al. give an overview of the state of the art in performance visualization [IGJ*14]. 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 [SLB*11]. Wylie and Geimer use Cartesian grids [WG11] in the Cube performance profile browser [GSS*12] to visualize performance in the simulation domain. We propose a tool that is similar in spirit to the Cube performance profile browser but enables visualizing performance data on arbitrary geometry.

3. Performance Profiles, Severity Views

Profiling is a common technique in performance analysis. A profile summarizes performance data over an application's complete runtime. 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* $s \in \mathcal{S}$ like compute nodes, CPU cores, etc. Besides the individual metrics and call paths, the system resources are organized hierarchically. That way, individual MPI ranks $s' \in \mathcal{S}_{MPI} \subseteq \mathcal{S}$ can be analysed separately.

During analysis, by selecting a pair of metric m and call path c, analysts specify a *severity view* $v_{m,c} : \mathcal{S} \mapsto \mathbb{R}$ 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. This work focuses on the performance of individual MPI ranks. We require the performance data to include the geometry in the simulation domain and a mapping from the MPI ranks to the parts of this geometry. Then, $v_{m,c}(s')$ yields the severity for the geometry part computed by MPI rank s'.

4. Detecting Variation in the Data

Visualizing the severity 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 variation across the MPI ranks. Otherwise, a single number would do. In order to identify large-variation severity views, our system uses the variation coefficient $q_{m,c} = \sigma_{m,c} \cdot \mu_{m,c}^{-1}$ 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. 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

http://dx.doi.org/10.2312/eurp.20161136

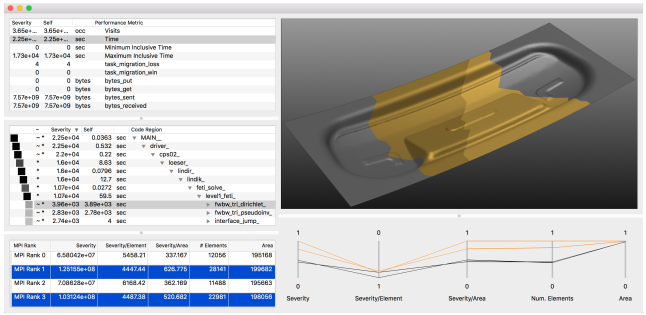

Figure 1: *User interface of our tool: the severity of "Time" in "fwbf_tri_dirichlet_" is visualized on the geometry (top right). The data from the table (lower left) is visualized as a parallel coordinates plot (lower right). The* MPI *ranks 1 and 3 are selected.*

	# Views
Non-Zero	29,482
Suggested	20,959

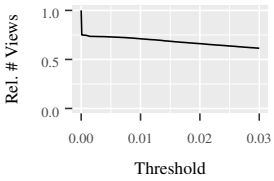

Figure 2: *Search space reduction: the table (left) lists the number of non-zero views and the number of suggested views ($\tau_q = 0.01$). The plot (right) presents the ratio of the number of suggested views to the number of non-zero views depending on the threshold.*

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.

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. 1). 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 tree 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, an asterisk indicates that such a view exists in the descendants.

5.2. Visualizing Performance in its Spatial Context

The 3D viewport in the upper right of the user interface (Fig. 1) 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 default. The simulation domain can be explored by moving a virtual camera with five degrees of freedom using keyboard and mouse. The elevation angle is limited to $\pm 90^\circ$, 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 lists the severities for each MPI rank and the properties, i.e., the number of finite elements and their surface area, of the associated part of the geometry. In addition, the severity is related to these properties. A parallel coordinates plot in the lower right presents the same data as the table for better overview. This is particularly helpful for simulations using many MPI ranks. Each axis is normalized and can be flipped.

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) [Sup]. The automatic suggestion mechanism effectively sieves out low-variation views: for $\tau_q = 0.01$ the search space is reduced by 29 % (Fig. 2). Larger thresholds filter out more views. 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. 1). 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) and by the Excellence Initiative of the German federal and state governments through the Jülich Aachen Research Alliance – High-Performance Computing.

http://dx.doi.org/10.2312/eurp.20161136

References

[GSS*12] 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). 1

[IGJ*14] 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). 1

[SLB*11] 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). 1

[Sup] SuperMUC petascale system [online]. URL: https://www.lrz.de/services/compute/supermuc/systemdescription/ [cited 2016-04-28]. 2

[WG11] WYLIE B. J. N., GEIMER M.: Large-scale performance analysis of PFLOTRAN with Scalasca. In *Proc. 53rd Cray User Group meeting* (2011), Cray User Group Inc. 1

http://dx.doi.org/10.2312/eurp.20161136