

# Separating the Wheat from the Chaff: Identifying Relevant and Similar Performance Data with Visual Analytics

Laura von Rűden  
GSC Computational Engineering  
Parallel Programming  
TU Darmstadt  
vonrueden@cs.tu-  
darmstadt.de

Marc-André Hermanns  
JARA-HPC  
RWTH Aachen University  
hermanns@jara.rwth-  
aachen.de

Michael Behrisch  
Data Analysis and Visualization  
University of Konstanz  
michael.behrisch@uni-  
konstanz.de

Daniel Keim  
Data Analysis and Visualization  
University of Konstanz  
daniel.keim@uni-  
konstanz.de

Bernd Mohr  
Jűlich Supercomputing Centre  
Forschungszentrum Jűlich  
b.mohr@fz-juelich.de

Felix Wolf  
Parallel Programming  
TU Darmstadt  
wolf@cs.tu-darmstadt.de

## ABSTRACT

Performance-analysis tools are indispensable for understanding and optimizing the behavior of parallel programs running on increasingly powerful supercomputers. However, with size and complexity of hardware and software on the rise, performance data sets are becoming so voluminous that their analysis poses serious challenges. In particular, the search space that must be traversed and the number of individual performance views that must be explored to identify phenomena of interest becomes too large. To mitigate this problem, we use visual analytics. Specifically, we accelerate the analysis of performance profiles by automatically identifying (1) relevant and (2) similar data subsets and their performance views. We focus on views of the virtual-process topology, showing that their relevance can be well captured with visual-quality metrics and that they can be further assigned to topical groups according to their visual features. A case study demonstrates that our approach helps reduce the search space by up to 80%.

## 1. INTRODUCTION

The basis of viable parallel-performance optimizations is the analysis of parallel-performance data, which can reveal inefficient program behavior. The performance of parallel applications is often far away from the machine's peak performance. This is often due to inefficiencies such as wait-states caused by load and communication imbalances [1]. The detection of such inefficiencies is facilitated by performance tools that are collecting performance-critical data

during the program run. This data can be used by application developers to gain insight into the behavior of their program and thus to get crucial hints for its potential optimization.

However, with hard- and software complexity on the rise, performance data is becoming so data intensive that its analysis poses serious challenges [2]. Performance data comprises numerous performance metrics, sometimes a hundred or more, as well as their distribution across the program, the runtime, or various system resources. The metrics combined with all the entities of program and system location span a very large search space. Increasing program complexity and scale let the amount of performance data grow even further. Consequently, the time and knowledge needed for the proper evaluation of the data is increasing drastically.

One approach to mitigate the exploration efforts is to use visual representations of performance data for developing an understanding of the application behavior. While many visualization techniques that turn performance data into performance views have been developed in the last years [3], the number of individual performance views for large and complex performance data becomes overwhelming. Often, a user can choose among thousands of different performance views—too many to find relevant and similar ones without the right intuition or guidance.

Therefore, we propose to simplify the analysis of parallel-performance data with visual analytics. Visual analytics describes an integral approach combining visualization, data analysis and human interaction [4, 5, 6]. Because neither visualization nor data analysis alone is sufficient for exploring large and complex data, visual analytics combines both techniques for creating fast and valuable insights.

The vision that we follow is that performance analysts do not need to inspect thousands of performance views, but only a small list of performance phenomena. This would significantly reduce the time for performance analysis, so that application developers could again concentrate on developing their applications. The steps that need to be taken in the direction of this vision are a reduction of the performance data to its relevant parts, an identification of similar parts,

an assignment of resulting similarity groups to performance phenomena and a technique for presenting these phenomena to the analyst. In this paper, we take the first steps and describe an approach for the automatic identification of relevant and similar performance data. The contributions of this paper are:

- Presentation of an approach for a performance-data search-space reduction using visual analytics.
- Presentation of concepts and selection of methods for the automatic identification of (1) relevant, and (2) similar performance data based on visual quality and visual features.
- Application to the Sweep3D performance data set, demonstrating the approach’s benefits by a search-space reduction of 80%.

The remainder of this paper is structured as follows: Section 2 describes related work and the background of parallel-performance data that we use in this paper for demonstrating our approach. Section 3 describes the challenges for defining a visual analytics driven analysis workflow. Built on this, Section 4 describes our approach for a search-space reduction with visual analytics. Section 5 then demonstrates the functionality of our approach by presenting results from a prototypical implementation. Section 6 discusses future work for visual analytics of parallel-performance data. Finally, Section 7 concludes this paper.

## 2. BACKGROUND AND RELATED WORK

In this section we describe related work and the background needed for this paper. We first describe the performance data model and the kind of performance views that we use to demonstrate our approach. Then we describe approaches for automatically identifying relevant and similar parts of performance data.

### 2.1 Performance Data Model

Performance data can be categorized with respect to the granularity into traces and profiles [7]. Traces provide fine-grained data through the collection of individual runtime events, whereas profiles provide coarse-grained data through aggregated performance metrics for every function call path and every system location. In this paper, we focus on performance profiles.

The Scalasca performance toolset [7] was designed with the idea of prescreening voluminous runtime-event traces to find execution patterns and aggregate the search results into much more compact profiles [8]. These profiles can be analyzed using Cube [9], a browser to explore performance metrics along the function call tree and the system resources.

The underlying Cube performance-data model is essentially a mapping of  $(m, c, s)$ -tuples onto a performance value  $p$ , i.e.,

$$p : M \times C \times S \rightarrow \mathbb{R},$$

where  $m \in M$  denotes the performance metric,  $c \in C$  the call path and  $s \in S$  the system location [8].

As illustrated in Figure 1, the data can be logically represented as a three-dimensional matrix, from

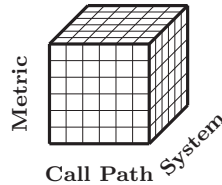


Figure 1: The Cube performance-data model.

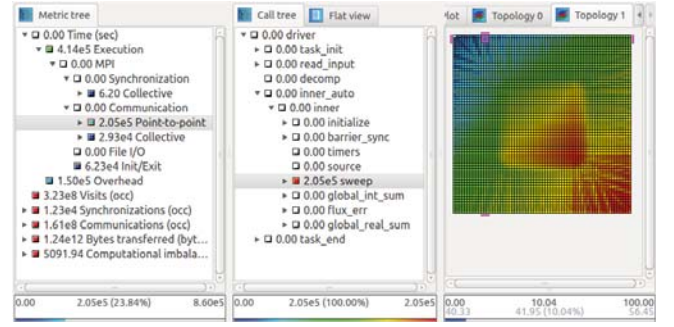


Figure 2: The Cube performance-data browser. The browser visualizes the data in three panes: the metric pane (left), the call path pane (middle), and the system pane (right). Here, the system pane shows the virtual-topology view.

which the name Cube is derived. All three dimensions are organized hierarchically: the performance metrics in a specialization hierarchy with subset semantics (e.g., communication time is a subset of execution time); the call paths in the call-tree hierarchy; and the locations in a relatively rigid machine hierarchy with the levels machine, node, process, and thread.

Cube performance data can be explored in the Cube performance-data browser. As shown in Figure 2, the browser provides three panes: the metric pane, the call-path pane and the system pane. Each of these panes visualizes its entities by default in a tree hierarchy. In the most common usage scenario, the user first chooses a metric in the left pane to see its distribution across the call tree in the middle pane and then selects a call path for which the chosen metric exhibits a high value. The distribution across the system resources for this particular metric call-path combination is then displayed in the right pane. In this paper, we denote the data for a metric call-path combination as a *data subset*.

### 2.2 Performance Views

Each data subset can be visualized and results in an individual performance view. A specific kind of a performance view is the *virtual-topology view*, which describes a mapping of process ranks onto multi-dimensional coordinates, representing the position of the process in the simulated domain. As shown in Figure 2, for Cube performance profiles, the spatial distribution of performance values along the system dimension can be visualized with virtual-topology views [13].

The effectiveness of visualizing performance data on intuitive domains, such as virtual-topologies views, is subject of current research. Schulz et al. showed that interpreting performance data across hardware, application and communication domains can give valuable insights into the behavior of parallel programs [10]. Spear et al. presented an approach to creating performance visualizations in a parallel profile analysis tool [11]. Huck et al. also presented an approach for linking performance data into scientific visualization tools [12].

Although such performance views are well suited for giving insight into the program behavior, they are often too numerous to be explored manually. In this paper, we therefore prescreen the performance views and automatically analyze them for relevance and similarity.

### 2.3 Identifying Relevant and Similar Performance Data

Current performance analysis tools can identify relevant and similar parts of performance profiles with techniques from data analysis, such as data mining.

In Cube, the relevance of a performance data subset is currently equated with the intensity of its performance metrics. The metric values are visualized in the tree hierarchies with color coded markers (see Figure 2) summarizing the intensity of a given metric for a whole data subset [8]. In the default color encoding, cold colors indicate low values and warm colors indicate high values.

PerfExplorer is a framework for performance-profile data mining [14]. This framework was developed together with the profile visualization tool ParaProf [15], which is used in the TAU performance system [16]. PerfExplorer uses statistical analysis packages, such as R or Octave, for performing cluster or correlation analysis. Cluster analysis is used for finding groups of processes that exhibit similar performance metrics. Correlation analysis is used for determining the relationship between different performance metrics.

However, additional contextual information from the visualizations of performance data (i.e., from performance views) has not yet been taken into account for automatically identifying relevant and similar performance data. In this paper, we therefore fill this gap and use an integral visual-analytics approach for parallel-performance data.

## 3. CHALLENGES IN EXPLORING PERFORMANCE DATA

Parallel-performance analysis often entails large and complex performance data. In this section we describe typical challenges in exploring performance data that need to be considered for defining a visual-analytics driven analysis workflow.

### 3.1 Exploring Large Search Spaces

Table 1 shows the sizes of performance-data search spaces from three applications examples (namely Sweep3D [17, 18], CICE [19, 20] and PFLOTRAN [21, 22]). Taking into account the number of the performance metrics, call paths and system resources together with its tree structures and in- and exclusive entities, the number of available data items is in the order of a billion. Even the metrics and call paths span a search space that already consists of thousands of individual data subsets resulting in thousands of individual performance views. The exploration of such large performance-data sets takes a significant amount of time and often is infeasible. To mitigate this problem, we prescreen the performance data and automatically identify those data subsets and views that are relevant for the understanding for performance phenomena.

In the following, we demonstrate our approach by using a performance data set from Sweep3D. Sweep3D is an ASCII benchmark code, solving a time-independent discrete ordinates 3D Cartesian geometry neutron transport problem [17, 23]. For the parallelization, the originally 3D computational domain is mapped onto a 2D virtual topology. The performance data set used is a profile created through an event-trace analysis with Scalasca [18]. The experiment was conducted with 294,912 MPI processes on an IBM Blue Gene/P system at the Jülich Supercomputing Centre.

Data set	$ M $	$ C $	$ S $	$ M \times C $	$ M \times C _{\neq 0}$
Sweep3D	95	41	294,912	8,694	846
CICE	110	54	2,634	11,826	844
PFLOTRAN	97	1533	16,384	356,580	33,330

Table 1: Sizes of performance-data sets and their search spaces. For each performance measurement the number of available performance metrics  $|M|$ , call paths  $|C|$ , and used system resources (processes)  $|S|$  is given. Taking into account their tree structures with in- and exclusive entities, the resulting number of data subsets, that is possible metric call-path combinations, is  $|M \times C|$ . The number of data subsets with non-zero values is given as  $|M \times C|_{\neq 0}$ .

### 3.2 Selecting Data or View Space

Performance data can be explored in the data space or in the view space. The data space describes the space of raw data, whereas the view space describes the space of views that result from mapping the raw data onto topologies, i.e., from visualizing the data [24]. An advantage of exploring performance data in the data space is the aggregation of information into simple statistics. Examples are the tree hierarchies in Cube [9] or the clustering in PerfExplorer [14]. An advantage of exploring performance data in the view space is the inclusion of contextual information. Examples are topology views, such as performance data mapped onto hardware, application or communication topologies as in [10], or onto virtual topologies [13].

In this paper, we explore performance data in its view space so that additional domain information is taken into account. In particular, we prescreen virtual-topology views and automatically identify those that are relevant and similar.

### 3.3 Defining Relevance

Only a small part of the large performance-data search space is relevant for performance analysis. Our understanding of relevance is motivated by the approach that Scalasca’s specific trace analysis follows: Load and communication imbalances manifest themselves in wait states that prevent parallel applications from making full use of available computing resources [1]. The Scalasca performance toolset searches for such wait states by measuring temporal displacements between matching operations [7]. Therefore, relevant performance data subsets are not necessarily the metrics or call paths with the largest values, but those with a combination of large and small values. We denote performance data subsets or views as relevant when they reveal load and communication imbalances between groups of processes. In contrast to that, irrelevant data subsets show well-balanced parallel behavior.

We found out that relevant performance views generally exhibit a high visual quality, that means they contain visual structures. Table 2 shows examples of performance views and illustrates *relevance* and *visual quality*. The first column shows an example of an irrelevant performance view. It reveals balanced parallel behavior in the form of a homogeneous view. The execution time in the call path `inner_auto` is evenly distributed and only shows noise in the virtual-topology view. The next four columns show relevant performance views, revealing imbalances in the form of visual structures. For example, the inclusive time spent in the call path `MPI_Recv` reveals imbalances between differ-



1. Relevance	✓	✓	✓	✓
2. Similarity				
Lines	✓			✓
Edges	✓	✓		✓
Gradients	✓		✓	

Table 2: Examples of performance views. The examples in this table show virtual-topology views from the Sweep3D performance-data set. The first column shows an example for an irrelevant view and the remaining columns show examples for relevant views. The relevant views can be further partitioned into similarity groups by considering three visual-feature classes: lines, edges (demarcating regions), and gradients (smooth gradients). These five performance views are used as reference views for illustrating our approach.

ent groups of processes. These are exposed in the virtual-topology view as visual structures, such as the edges from the central overload region, the gradient of increasing time from northwest to southeast, and the oblique lines radiating from the overload region to the borders.

### 3.4 Finding Similarities

An analyst needs to examine and compare several data subsets for finding links between performance metrics and call paths for understanding performance inefficiencies. For a particular phenomenon often the spatial distribution in the performance view is characteristic. Therefore, we identify similar performance-data based on similar visual structures in their performance views.

We found out that the similarity of performance views can be described based on three general visual-feature classes: lines, edges and gradients. A line describes an abrupt difference of the color level in an image and is typically embedded in a single homogeneous region. It can describe an abrupt increase and following decrease of the color level, or vice versa. An edge also describes an abrupt difference of the color level in an image, but it demarcates two different regions. It can describe either an abrupt increase of the color level, or an abrupt decrease. A gradient, specifically a smooth gradient, describes a gradually changing color level. Table 2 shows how the relevant reference views are assigned to these visual-feature classes. Since each view can reveal not only a single performance phenomenon, but also a superposition of phenomena, each view can be assigned to several feature classes.

The visual-feature classes can indicate different kind of performance inefficiencies. Lines indicate various kinds of performance imbalances that can be due to the parallel algorithm itself, the scheduling, or the mapping. Edges indicate algorithm specific under- or overload regions. (It is not the region that describes a performance inefficiency, but the imbalance between different regions. Such an imbalance can visually be described by the separation between the regions, that is the edge demarcating them.) Smooth gradi-

ents indicate algorithm-specific waiting times in collective operations.

Since the performance views that are assigned to the same feature class can still be different from each other, these data views can be further partitioned into similarity groups, each representing a particular performance phenomenon. While the feature classes are generic, the similarity groups are specific for each performance-data set.

## 4. REDUCING THE PERFORMANCE-DATA SEARCH SPACE WITH VISUAL ANALYTICS

We simplify the analysis of performance data with visual analytics. According to the visual-analytics process, as defined by Keim et al. [4], data can be turned into knowledge through visualizations, data models, or an integration of both. Visualization techniques can not only be applied to raw data, but also to data models. The other way around, automatic data analysis methods can not only be applied to raw data, but also to visualizations (i.e., to views).

### 4.1 Automatic Analysis of Performance Views

We combine data analysis and visualization for identifying relevant and similar performance views. We identify relevant performance data by analyzing it in the view space, because the performance views enhance the data with contextual information. In particular, we analyze the data subsets based on their virtual-topology views.

As illustrated in Figure 3, we reduce the performance data and view space through an automatic identification of first the relevant, and second the similar data subsets, respectively their views. For this purpose, we prescreen all virtual-topology views. We process the views as matrices of performance values linearly converted to grayscale. In the first step of the search-space reduction, we automatically identify the metric call-path combinations that lead to potentially relevant performance views. In the second step, we analyze the potentially relevant performance views for



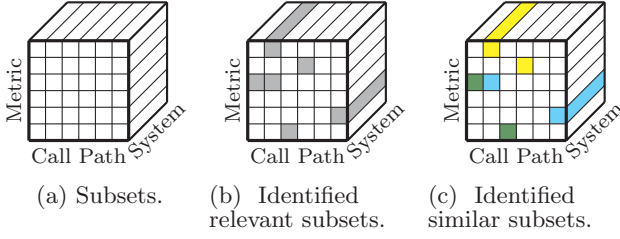


Figure 3: Steps of the performance-data search-space reduction. We analyze performance data based on data subsets and their performance views (3a). A data subset contains the performance values along the system dimension for a metric call-path combination and can be visualized in a virtual-topology view. The search space of performance data subsets, or performance views, can be reduced in two steps: First, the relevant performance views are identified (3b). Second, the relevant performance views are partitioned into similarity groups (3c).

similarity and partition them into similarity groups, each representing a particular performance phenomenon.

The benefit of our approach—identifying relevant and similar performance data and its views with visual analytics—is a reduction of the data and view search space. In particular, it leads to both a faster and more comprehensive performance analysis process. First, the analyst can concentrate on the analysis of a small, relevant portion of the original data. Since data subsets that are potentially relevant are identified automatically, the analyst does not need to search for them manually anymore. The information about the relevant data subsets can either be used to distill a reduced performance-data set or to guide the analyst during the analysis to the relevant views. Second, the analyst automatically gets additional information about the performance data in the form of similarity groups, each representing a particular performance phenomenon. This way, the analyst not only saves the time of finding the similarities manually, but also gets additional insights in the parallel program’s behavior. Such additional insights can be, for example, an overview of all occurring performance phenomena, or information about (hidden) dependencies between metrics and call paths.

## 4.2 Methods for Automatic Analysis

Based on our notion that relevant performance views are characterized by high visual quality (as explained in Section 3.3), they can be automatically identified with methods from visual-quality analysis. As surveyed by Bertini et al. [24], visual-quality analysis uses automatic analysis methods to evaluate the visual quality, that is the potential relevance, of given views. These methods apply quality metrics—measures that assess the quality of views by abstractly quantifying their information content. Their main objective is to rank a group of views according to their potential relevance. Quality metrics that can be used for evaluating performance views are metrics that are designed for pixel-based visualization techniques, for example, the Noise-Dissimilarity measure [25], or the entropy and standard deviation that are used in the Pixnostics approach [26]. Whereas the Noise-Dissimilarity evaluates the mappings based on their dissimilarity to a noise function, Pixnostics evaluates them based on entropy or standard deviation. Since the Noise-

Dissimilarity measure can filter various visual structures, we prefer this measure for evaluating performance views.

Based on our notion that similar performance views are characterized by similar visual structures (as explained in Section 3.4), they can be automatically identified with methods from visual-feature detection. We first analyze the views for the occurrence of visual features, and possibly assign them to corresponding feature classes. For this, detection methods from image processing [27] can be used. We further partition the views that are assigned to a feature class into similarity groups, each representing a particular performance phenomenon. For this, we use methods from data mining, specifically Gaussian kernel estimation [28].

## 5. RESULTS FROM PROTOTYPE

To demonstrate our approach for identifying relevant and similar performance views using visual analytics, we show results from our prototype. First, we show results from the relevance estimation with the Noise-Dissimilarity method. Second, we show results from the similarity identification based on visual features.

### 5.1 Identifying Potentially Relevant Views

Before we analyze the performance views with the actual analysis method, i.e., the Noise-Dissimilarity method, we filter out all views with performance value distribution that is constantly zero. This preliminary step reduces the Sweep3D performance data set, which initially provided 8,694 views (see Table 1), to 846 views.

These views are then analyzed for relevance with the Noise-Dissimilarity method [25]. It evaluates the visual quality of an image based on a measure (the Noise-Dissimilarity measure  $NDM(g, g)$ ) that quantifies the dissimilarity between the original image (with its gray values  $g$ ) and the corresponding noise image (with its gray values  $g$ ). The noise image is created from the original image through a random permutation of the pixel values. The larger the Noise-Dissimilarity Measure  $NDM(g, g)$  of an image, the higher is its potential relevance.

Table 3 illustrates how the Noise-Dissimilarity method measures the potential relevance of performance views by showing results for the Sweep3D reference views. Each row illustrates the results for a single performance view converted to a linear grayscale, which is shown in the first column. The second column shows corresponding noise images. The dissimilarity images in the third column illustrate the dissimilarity between the original image and the corresponding noise image. The dissimilarity images reflect the visual structures, which are characteristic for a relevant performance view, of the original image. The automatic judgment of the relevance, which is based on the quality metric and a fixed relevance threshold, agrees with the manual judgment for all five reference views.

From all views of the Sweep3D performance data set (processed in peer-percent mode), the Noise-Dissimilarity method identifies 225 views as potentially relevant. Taking into account all available views (i.e., 8,694 views, including the views showing a performance value distribution that is constantly zero, which we filtered out in a preliminary step) we reduce the search space by 97%. Considering the 846 non-zero views as the true search space, the Noise-Dissimilarity method reduces the search space by 74%. Assuming that the size of the view search space is proportional to the time
















Noise Dissimilarity			Quality Metric		
Image $g$	Noise $g$	$\Delta(g, g)$	NDM( $g, g$ )	Rank	Rel.
			0.01	431	
			0.13	69	✓
			0.10	120	✓
			0.15	50	✓
			0.06	186	✓

Table 3: Example results of the Noise-Dissimilarity method for the Sweep3D performance data set. Each row shows intermediate and final results for a single reference data subset. Intermediate results are the image  $g$ , the noise image  $g$ , and the dissimilarity image  $\Delta(g, g)$ . The quality metric NDM( $g, g$ ) approximates the visual quality of the original image, respectively the potential relevance of the performance data subset. The predicted rank is based on the performance data set without zero views, which consists of 846 data subsets. The lower the rank, the more relevant. For dividing the performance views in relevant and irrelevant views, we used a threshold of  $\text{NDM}(g, g)_{\text{rel}} \geq \text{NDM}(g, g)_{\text{thresh}} = 0.04$ .

that is needed for the analysis of performance data set, even our prototypical implementation of visual-quality analysis can reduce this time significantly.

## 5.2 Grouping Views with Similar Visual Structures

We analyze the performance views that have been identified as being relevant in a further step for similarities. Our aim is to group views with similar visual structures.

For this, we proceed the following way: First, we analyze all the potentially relevant views for visual structures. For each of the defined feature classes (lines, edges, smooth gradients) a separate detection method is used. We apply image analysis techniques, such as a line-detection algorithm [27]. The methods analyze the occurrence of the respective feature in a performance view and possibly assign the view to the respective feature class. Second, we analyze for each feature class the views assigned to it for similarities. For this, we measure statistics for each view, such as the mean line length, and group those views with similar statistics. For the grouping we use kernel density estimation, in particular Gaussian kernel estimation [28].

Table 4 illustrates the results of the similarity group identification for the performance views of the Sweep3D data set. For this data set, we identify five similarity groups: Three similarity groups in the line-feature class, one in the edge-feature class, and one in the gradient-feature class. 20% of

the 846 performance views are assigned to any of the similarity groups. This represents a search-space reduction of 80%. Even better: we reduce the search space to only five similarity groups, each representing a particular performance phenomenon.

With respect to the similarity groups, the following performance phenomena are detected: (1) The first similarity group in the line-feature class, which is shown in 4% of the performance views, exposes a visual structure consisting of many short horizontal lines. This structure can be assigned to the mapping of the processes to the hardware resources. The processes that become visible as a group through a short horizontal line are processes that are assigned to one node board. The lines show that the work is not perfectly balanced between the node boards and thus probably reveal a mapping problem. (2) The second similarity group in the line-feature class, which is shown in 4% of the performance views, exposes a visual structure consisting of oblique lines radiating from the interior to the border of the domain. As described by Wylie et al. in a Sweep3D scaling study [18], these oblique lines reveal a computational imbalance that is due to so called “fixup iterations”, which apply corrections to negative fluxes and are necessary for a physically realistic solution. (3) The third similarity group in the line-feature class, which is shown in 8% of the performance views, exposes a visual structure consisting of long vertical lines. These lines illustrate the systematic deviations in the execution time for a broadcast operation and is thus the effect of a communication imbalance. (4) The one similarity group in the edge-feature class, which is shown in 8% of the performance views, exposes a visual structure consisting of a central rectangle. The central rectangle reveals a load imbalance and is, like the oblique lines of similarity group (2), also the result of the “fixup iterations” [18]. (5) The one similarity group in the gradient-feature class, which is shown in 2% of the performance views, exposes a visual structure consisting of a diagonal smooth gradient. This gradient illustrates the difference in the processes’ waiting time, which results from the subsequent sweeping wavefronts. While the last wavefront is sweeping through the domain all other processes have to wait until it is finished, resulting in a gradually changing waiting time for the domain.

## 6. FUTURE WORK

While our prototype already demonstrated the usefulness of using an integral visual-analytics approach for parallel-performance data, we have identified four central research directions that should be further explored in the future.

We demonstrated the benefits of using an integral visual-analytics approach specifically for performance profiles. However, this approach, particularly the automatic identification of relevant and similar performance views, can and should also be used for other kinds and performance data and views. For these the definition of relevance and similarity should be adapted and methods from visual-quality analysis and visual-feature detection should be selected accordingly.

Another important research question for future work is the correlation between specific visual features and performance phenomena; and possibly also physical phenomena. Our vision is that performance analysts do not need to explore thousands of performance views, but only a few groups, each representing a particular performance phenomenon. Striving for such a significant simplification of the whole perfor-

Kernel estimate	Group	Struct.	Ex.
<p>Mean line angle</p> <p>L. 1 (4%)</p> <p>L. 2 (4%)</p> <p>L. 3 (8%)</p>			
<p>Mean edge length</p> <p>E. 1 (8%)</p>			
<p>Mean smooth gradient</p> <p>G. 1 (2%)</p>			
<b>ANY: 20%</b>			

Table 4: Results of the similarity group identification for the performance views of the Sweep3D performance data set. The table shows the similarity groups for each visual feature class. The first row shows the results for the lines, the second for edges, and the third for smooth gradients. The first column shows the Gaussian kernel estimate for the corresponding visual feature—each maximum represents a similarity group. The similarity groups are illustrated in the remaining columns. The group percentages gives the share of views assigned to the similarity group (with respect to a search space of 846 views). In total, 20% of the views are assigned to any group. The views on the right show the visual structure representing the similarity group, as well as a performance view example.

mance analysis process, the meaning of visual features in the context of parallel performance—the connection of specific visual structures and their similarity groups to specific performance phenomena and their root causes—should be further investigated.

Future work should address the integration of the automatic analysis and its results in the graphical user interface of the performance-data explorer. This should include visual guidance to the relevant performance views and an overview of the similarity groups of occurring performance phenomena. Analytic workflows that follow the Overview-and-Detail Mantra [29] could be used for representing general relationships among the performance metrics, program and system locations, or combinations thereof. However, the depiction of representatives for the found groups, as well as the visualization of the inter-group relationships, is challenging in itself and might influence the analytic workflow. Especially promising is also the coupling of data analysis and

visualization through human-computer interaction: Interactive relevance-thresholds allow a filtering of performance data until the desired level is reached.

Another interesting research question could be the automatic identification of relevant low-dimensional projections for high-dimensional topologies.

## 7. CONCLUSION

Performance analysis is essential for the optimization of parallel programs. However, the data to be analyzed often confronts the analyst with a very large search-space consisting of thousands of individual data subsets and views.

In this paper, we proposed to simplify the performance analysis of parallel programs by means of visual analytics, and presented an approach for a search-space reduction through an automatic identification of relevant and similar performance views. The benefit of our approach is a reduction of the data and view search space. In particular, it leads to both a faster and more comprehensive performance analysis process.

We presented an approach for the search-space reduction of profile-based performance data using visual analytics consisting of the two steps (1) identification of *relevant* performance views and (2) identification of *similar* performance views. We showed that the potential relevance of performance views can be estimated with methods from visual-quality analysis, such as the Noise-Dissimilarity method. Since similar performance views show similar visual structures in their virtual-topology views, we identified them with visual-feature detection. We demonstrated that topology-views can be classified according to the visual features lines, edges and smooth gradients, and that the views assigned to each feature class can be partitioned into similarity groups. Through the automatic identification of relevant and similar performance views with these methods, we achieved a search-space reduction of 80% and condensed the Sweep3D performance-data set to five similarity groups, each representing a particular performance phenomenon.

All in all, our findings extend the knowledge-based approach of the performance tool Scalasca, which already reduces detailed event traces to a condensed profile [7, 8]. We showed that performance profiles can be even more condensed to similarity groups, each representing a particular performance phenomenon. Such a condensation reduces the time needed for performance analysis significantly.

## 8. ACKNOWLEDGMENTS

The work of Laura von Rden is supported by the ‘Excellence Initiative’ of the German Federal and State Governments and the Graduate School of Computational Engineering at Technische Universitt Darmstadt.

## 9. REFERENCES

- [1] D. Bhme, M. Geimer, and F. Wolf, “Characterizing load and communication imbalance in large-scale parallel applications,” in *Proc. of the 26th IEEE Int. Parallel and Distributed Processing Symp. Workshops and PhD Forum (IPDPSW)*, Shanghai, China, pp. 2538–2541, IEEE Computer Society, May 2012.
- [2] P.-T. Bremer, B. Mohr, V. Pascucci, and M. Schulz, “Connecting Performance Analysis and Visualization to Advance Extreme Scale Computing (Dagstuhl

- Perspectives Workshop 14022),” *Dagstuhl Reports*, vol. 4, no. 1, pp. 17–35, 2014.
- [3] K. E. Isaacs, A. Giménez, I. Jusufi, T. Gamblin, A. Bhatele, M. Schulz, B. Hamann, and P.-T. Bremer, “State of the Art of Performance Visualization,” in *EuroVis - STARs* (R. Borgo, R. Maciejewski, and I. Viola, eds.), The Eurographics Association, 2014.
  - [4] D. A. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, eds., *Mastering the Information: Age Solving Problems with Visual Analytics*. Eurographics Association, 2010.
  - [5] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler, “Visual analytics: Scope and challenges,” in *Visual Data Mining*, pp. 76–90, Springer Science + Business Media, 2008.
  - [6] K. A. Cook and J. J. Thomas, *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. May 2005.
  - [7] M. Geimer, F. Wolf, B. J. N. Wylie, E. Abraham, D. Becker, and B. Mohr, “The Scalasca performance toolset architecture,” *Concurrency and Computation: Practice and Experience*, vol. 22, pp. 702–719, Apr. 2010.
  - [8] F. Wolf and B. Mohr, “Automatic performance analysis of hybrid mpi/openmp applications,” *J. Syst. Archit.*, vol. 49, pp. 421–439, Nov. 2003.
  - [9] F. Song, F. Wolf, N. Bhatia, J. Dongarra, and S. Moore, “An algebra for cross-experiment performance analysis,” in *Proc. of the 2004 Int. Conf. on Parallel Processing, ICPP ’04*, (Washington, DC, USA), pp. 63–72, IEEE Computer Society, 2004.
  - [10] M. Schulz, J. A. Levine, P.-T. Bremer, T. Gamblin, and V. Pascucci, “Interpreting performance data across intuitive domains,” in *ICPP* (G. R. Gao and Y.-C. Tseng, eds.), pp. 206–215, IEEE, 2011.
  - [11] W. Spear, A. D. Malony, C. Lee, S. Biersdorff, and S. Shende, “An approach to creating performance visualizations in a parallel profile analysis tool,” in *Euro-Par 2011: Parallel Processing Workshops* (M. Alexander, P. D’Ambrá, A. Belloum, G. Bosilca, M. Cannataro, M. Danelutto, B. Di Martino, M. Gerndt, E. Jeannot, R. Namyst, J. Roman, S. Scott, J. Traff, G. Vallée, and J. Weidendorfer, eds.), vol. 7156 of *Lecture Notes in Computer Science*, pp. 156–165, Springer Berlin Heidelberg, 2012.
  - [12] K. A. Huck, K. Potter, D. W. Jacobsen, H. Childs, and A. D. Malony, “Linking performance data into scientific visualization tools,” in *Proc. of the First Workshop on Visual Performance Analysis, VPA ’14*, (Piscataway, NJ, USA), pp. 50–57, IEEE Press, 2014.
  - [13] N. Bhatia, F. Song, F. Wolf, J. Dongarra, B. Mohr, and S. Moore, “Automatic experimental analysis of communication patterns in virtual topologies,” in *In Proc. of the Int. Conf. on Parallel Processing (ICPP)*, IEEE Society, 2005.
  - [14] K. A. Huck and A. D. Malony, “Perfexplorer: A performance data mining framework for large-scale parallel computing,” in *SC*, p. 41, IEEE Computer Society, 2005.
  - [15] R. Bell, A. D. Malony, and S. Shende, “Paraprof: A portable, extensible, and scalable tool for parallel performance profile analysis,” in *Euro-Par* (H. Kosch, L. Böszörményi, and H. Hellwagner, eds.), vol. 2790 of *Lecture Notes in Computer Science*, pp. 17–26, Springer, 2003.
  - [16] S. S. Shende and A. D. Malony, “The tau parallel performance system,” *Int. J. High Perform. Comput. Appl.*, vol. 20, pp. 287–311, May 2006.
  - [17] Los Alamos National Laboratory, “The ASCI Sweep3D readme file: 3d discrete ordinates neutron transport.” <http://www3.lanl.gov/pal/software/sweep3d/>, 2005. [Online; accessed 02-March-2015].
  - [18] B. J. N. Wylie, M. Geimer, B. Mohr, D. Böhme, Z. Szebenyi, and F. Wolf, “Large-scale performance analysis of Sweep3D with the Scalasca toolset,” *Parallel Processing Letters*, vol. 20, pp. 397–414, Dec. 2010.
  - [19] National Center for Atmospheric Research, “Community ice code (cice) user’s guide, version 4.0, released with ccs4.0.” [http://www.cesm.ucar.edu/models/ccsm4.0/cice/ice\\_usrdoc.pdf](http://www.cesm.ucar.edu/models/ccsm4.0/cice/ice_usrdoc.pdf), 2010. [Online; accessed 04-March-2015].
  - [20] D. Böhme, *Characterizing Load and Communication Imbalance in Parallel Applications*. PhD thesis, RWTH Aachen University, volume 23 of IAS Series, Forschungszentrum Jülich, Feb. 2014. ISBN 978-3-89336-940-9.
  - [21] P. C. Lichtner, G. E. Hammond, C. Lu, S. Karra, G. Bisht, B. Andre, R. T. Mills, and J. Kumar, “PFLOTTRAN Web page,” 2013. <http://www.pfлотran.org>.
  - [22] B. J. N. Wylie and M. Geimer, “Large-scale performance analysis of PFLOTTRAN with Scalasca,” in *Proc. of the 53rd Cray User Group meeting, Fairbanks, AK, USA*, Cray User Group Inc., May 2011.
  - [23] A. Hoisie, O. M. Lubeck, and H. J. Wasserman, “Performance and scalability analysis of multidimensional wavefront algorithms on teraflop-scale architectures,” in *PPSC*, 1999.
  - [24] E. Bertini, A. Tatu, and D. A. Keim, “Quality Metrics in High-Dimensional Data Visualization: An Overview and Systematization,” *IEEE Symp. on Information Visualization (InfoVis)*, vol. 17, pp. pages 2203–2212, Dec. 2011.
  - [25] G. Albuquerque, M. Eisemann, D. J. Lehmann, H. Theisel, and M. Magnor, “Improving the visual analysis of high-dimensional datasets using quality measures,” in *Proc. IEEE Symp. on Visual Analytics Science and Technology (VAST) 2010*, (Salt Lake City, Utah, USA), pp. 19–26, Oct. 2010.
  - [26] J. Schneidewind, M. Sips, and D. A. Keim, “Pixnostics: Towards Measuring the Value of Visualization,” in *Proc. of the IEEE Symp. on Visual Analytics Science and Technology (VAST ’06)*, 2006.
  - [27] T. Acharya and A. K. Ray, *Image Processing - Principles and Applications*. Wiley-Interscience, 2005.
  - [28] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006.
  - [29] B. Shneiderman, “The eyes have it: A task by data type taxonomy for information visualizations,” in *Visual Languages, 1996. Proc., IEEE Symp.*, pp. 336–343, IEEE, 1996.