An Approach to Creating Performance Visualizations in a Parallel Profile Analysis Tool

Wyatt Spear, Allen D. Malony, Chee Wai Lee,
Scott Biersdorff, and Sameer Shende

Department Computer and Information Science,
University Oregon, Eugene, Oregon, 97403

Abstract. With increases in the scale of parallelism the dimensionality and complexity of parallel performance measurements has placed greater challenges on analysis tools. Performance visualization can assist in understanding performance properties and relationships. However, the creation of new visualizations in practice is not supported by existing parallel profiling tools. Users must work with presentation types provided by a tool and have limited means to change its design. Here we present an approach for creating new performance visualizations within an existing parallel profile analysis tool. The approach separates visual layout design from the underlying performance data model, making custom visualizations such as performance over system topologies straightforward to implement and adjust for various use cases.

1 Introduction

The performance measurement and analysis of large-scale parallel applications requires means for understanding the features within the multi-dimensional performance datasets and their relation to the computational and operational aspects of the application and its execution. While automatic analysis of performance behavior and diagnosis of performance problems is desired, performance tools today invariably involve the user for performance results interpretation. Presentation of performance information has been regarded as an opportunity for conveying visually characteristics and traits in the data. However, it has always been challenging to create new performance visualizations, for three reasons. First, it requires a design process that integrates properties of the performance data (as understood by the user) with the graphical aspects for good visual form. This is not easy, if one wants effective outcomes. Second, unlike visualization of physical phenomena, performance information does not have a natural semantic visual basis. It could utilize a variety of graphical forms and visualization types (e.g., statistical, informational, physical, abstract). Third, with increasing application concurrency, performance visualization must deal with the problem of scale. The use of interactive three-dimensional (3D) graphics clearly helps, but the visualization design challenge is still present.

In addition to these challenges, there are also practical considerations. Because of the richness of parallel performance information and the different relationships to the underlying application semantics, it is unreasonable to expect just a few performance visualizations to satisfy all needs. Where visualization of large performance information does exist, it is generally embedded as "canned" displays in a profile or trace analysis tool. If a user has a different concept in mind, they have very limited ability to make changes. There are ways around the dilemma. One approach might be to use a visualization environment (e.g., VisIt [3] or ParaView [2]) which provides robust support for building and producing visualizations. These environments have a targeted user group, and it is not parallel performance analysts.

An alternate approach is to work within a performance analysis tool and build capabilities for visualization that allow more user design input and interaction. This could be accomplished simply by an interface to query performance data from within the tool, and passing it over to a component implemented with a visualization library that the user programs directly (e.g. VTK [4]). Unfortunately, few users would have the expertise to take advantage of this. Instead, we took a path to constrain how visual layout and user interface components are specified. The tool then constructs a performance visualization from the specifications.

2 Design Approach

The approach we followed for performance visualization design was motivated by our experience with the 3D visualization in the *ParaProf* profile analysis tool [7] provided by the TAU performance system [17]. ParaProf's primary visualization mode presents 2D charts of a metric for each event measured in profiled application. Metrics can be execution time (inclusive, exclusive), event count, or hardware counters. The 2D charts include bar plots, histograms, communication "heat maps," among others. In addition, ParaProf presently provides two types of 3D performance views:

– ***Full profile***: A landscape visualization is used to show the entire parallel profile for all events and threads of execution for a single performance metric. The view is drawn as a triangle mesh or bar plot with events on the X-axis, threads on the Y-axis, and (event,thread) metric value on the Z-axis.
– ***Event correlation***: A scatter plot visualization is used to show the relationship between four events (each with its own performance metric) for all threads of execution. The first three event/metric pairs set the spatial coordinates for each thread with the fourth determining color.

For both visualizations, the UI allows the user to select how the performance event/metric pair is displayed. Both visualizations are implemented in JOGL, Java's interface to OpenGL, and interactive rotation and zooming are provided.

Both of these 3D views were developed to target specific use cases. Without any additional support, any new visualization would also be implemented that way. Thus, when we wanted to develop a new visualization for a single event and metric that was based on a layout of threads according to topological information, it became clear that a more general methodology was required; otherwise, a separate implementation would be needed for each different layout.

To move towards a more general method of creating 3D visualizations, we considered the salient components of the existing versions and the need to specify aspects of a 3D design in a more flexible manner. Two ideas resulted: 1) separate the visualization layout design from visualization user interface (UI) design, and 2) allow the properties of each to be specified by the user. The new design methodology is shown in Figure 1.

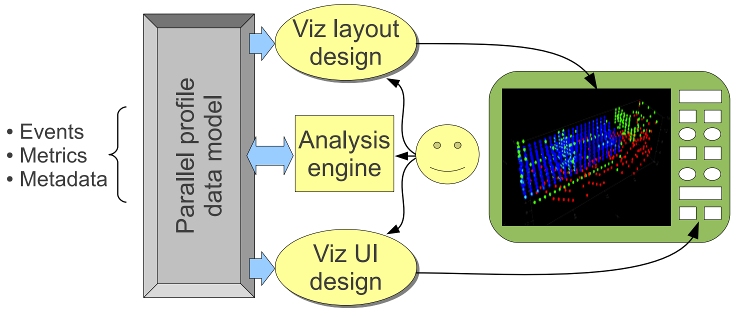

Fig. 1. Visualization architecture

Visualization layout design is concerned with how the visualization will appear. Our approach allows the visual presentation to be specified with respect to the parallel profile data model (events, metrics, metadata) and possible analysis of this information. Two basic layout approaches we support are mapping to cartesian coordinates provided by MPI and filling a space of user-defined dimensionsions in order of MPI rank. We have also worked to develop a specification language for describing more complex layouts of thread performance in a 3D space. In our initial implementation of these custom layouts, mathematical formulae define the coordinates and color value of each thread in the layout. The formulae are based on *variables* provided by the profile data model. These input variables include event and metric values for the current thread being processed as well as global values such as the total number of threads in the profile. The specification is applied successively to each thread in the profile to determine X, Y and Z coordinate values and color values which are used to generate the visualization graphics. Our initial implementation for expression analysis uses the MESP expression parser library[5]. MESP provides a simple syntax for expressing mathematical formulae but is powerful enough to allow visualization layouts based on archetecturally relevant geometries or the mathematical relationship of multiple performance variables.

Visualization UI design is concerned with how the visualization will be controlled. The key insight here is to have the UI play a role in "binding" data model variables used in the layout specification. This approach implements the functionality present in the current ParaProf views, where the user is free to select events and metrics to be applied in the visualization as inputs to layout formulae. However, for large performance profiles of many threads/processes, the specified layout can result in a dense visualization that obscures internal structures. The current ability to zoom and rotate the the topology in the UI partially ameliorates this issue. Our model for visualization UI further allows more sophisticated filtering techniques.

3 Examples

The performance visualization design approach is being developed in the ParaProf profile analysis tool. Here we demonstrate our current prototype for three applications: Sweep3D, S3D and GCRM/ZGrd. Our initial focus is on topology visualization. In addition, we illustrate the flexibility of these techniques by recreating ParaProf's event correlation view.

3.1 Sweep3D

For development and testing of our 3D visualization approach we used data from the Sweep3D[6] particle transport code. The Sweep3D performance data set we used was generated from a 16k core run on an IBM Blue Gene/L system and contains Cartesian coordinates of each MPI rank from the MPI system [19]. The most obvious topology mapping scheme is to take the rank-to-coordinate mapping and use it to lay out the points representing the ranks in a 3D space. Figure 2 shows this performance view for the exclusive time in `MPI_Barrier`. The layout specification is defined with respect to MPI ranks while event and metric variables are selected in the UI.

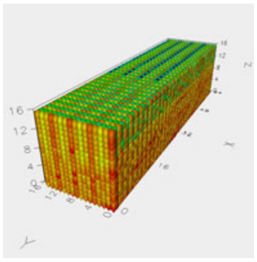

Fig. 2. Sweep 3d BG/L 16k-core mapping as provided by MPI

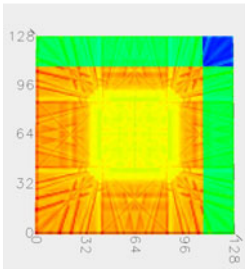

Fig. 3. Sweep 3d BG/L 16k-core user-defined mapping

When information about the physical layout of a system is provided, it is straightforward to write the layout expression and to interpret the results. However there are a number of cases where basic MPI provided coordinate mapping will not suffice. Just as mapping schemes vary between MPI implementations, the means of accessing topology mapping data is also variable. The relevant mapping information may not be available in performance data from which a topological display is desired. Even when coordinate data is available there are potential issues which may render it inappropriate for the performance analysis task at hand. For example, the underlying, machine-level topology may not be the topology of interest. Higher level topologies, relating to how work is allocated may have different or no ready means of programatically associating ranks with topological coordinates. Another issue that can arise is the need to incorporate another dimension, such as thread (core) ID, in the display. In such situations it is necessary to find other means of rank mapping. In general, greater flexibility in how ranks are visualized allows for more complete analysis of an application with respect to topology.

Figure 3 shows a user-specified visualization defining a topology with meaningful spatial context for performance data based on a block-wise layout of MPI ranks, in this case in two dimensions. This show that even a basic linear stacking with respect to rank id can produce valuable interpretive effects.

More general topological renderings produced by mathematical expressions can serve a number of purposes, including defining more complex hardware topologies and other spatial representations of computational activity. To demonstrate the power of the layout specification, Figure 4 illustrates a spherical visualization of the same Sweep3D performance data.

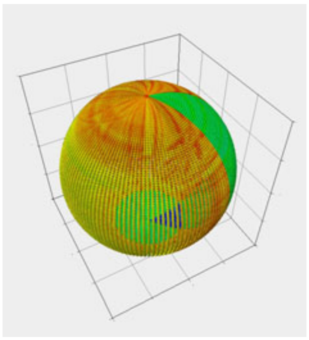

Fig. 4. Sweep 3D 16k-core mapping with spherical topology

Listing 1.1. Topology configuration file for sphere transformation

```
BEGIN_VIZ=sphere
rootRanks=sqrt(maxRank)
theta=2*pi() /
   rootRanks*mod(rank,rootRanks)
phi=pi() /
   rootRanks*(floor(rank/rootRanks))
x=cos(theta)*sin(phi)*100
y=sin(theta)*sin(phi)*100
z=cos(phi)*100
END_VIZ
```

Listing 1.1 shows the expressions mapping ranks to points on the surface of a sphere. The X, Y, and Z formulae are required, but additional helper functions may be provided to simplify the expressions. Several variables, such as `maxRank` and `rank` are provided internally. The topology formulae are defined in a standard text file using MESP's syntax. The file may be loaded and refreshed from within ParaProf, allowing rapid development and adjustment of application or purpose specific topologies, as well as easy sharing of topological definitions between collaborators.

Once a visual layout is defined, the UI can control selection and filtering. A particularly effective setting defines a displayable range based on minimum and maximum values. Thread points are not displayed if the value of the metric event combination representing color exceeds the maximum or is less than the minimum. If the minimum is set above the maximum, only threads with values above the minimum and below the maximum are shown, meaning only high and low values are displayed. This is very important for identifying the topological patterns of performance outliers, and is shown for Sweep3D in Figure 5.

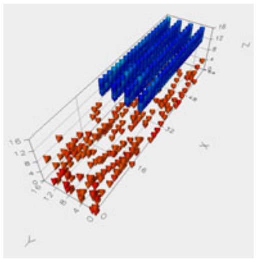

Fig. 5. Sweep 3d Topology with middle values excised

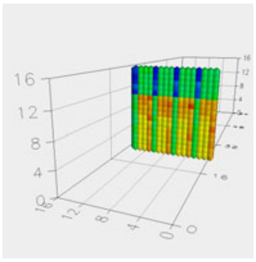

Fig. 6. Sweep 3d topology slice along x axis

The UI also allows exclusion by locality. For example, if a value along the X axis is selected, only points appearing along that "slice" will be visible. Excluding by one, two or three axes results in the visualization of a plane, line or point respectively. The average value of the metric for the ranks in a selected area will be displayed in each case, with three selected axes displaying the actual value for the single selected rank. This is demonstrated in Figure 6 for Sweep3D.

3.2 S3D Use Case

The S3D application [9] is a massively parallel turbulent combustion simulator developed by Sandia National Laboratories. The core DNS (Direct Numerical Simulation) solver code of S3D is parallelized using three dimensional domain decomposition over a Cartesian mesh. We examined data collected by TAU from several scaling runs of S3D on the Intrepid IBM Blue Gene P system. Previous performance analysis of S3D [16] has suggested topology dependent performance behavior centered around MPI communication. The communication-topology dependent nature of the code made it a strong candidate for re-analysis with topological performance visualization.

The Cartesian topology coordinates provided by MPI for the S3D run on intrepid cover the three spatial dimensions of nodes within the system's layout. However, on BG/P, each node contains two processors with two cores per processor. The default behavior of the topology view, given pre-defined coordinates with a 4th thread or core-level dimension, is to take the average of the thread or core values on each node and display that value in the 3D node level topology.

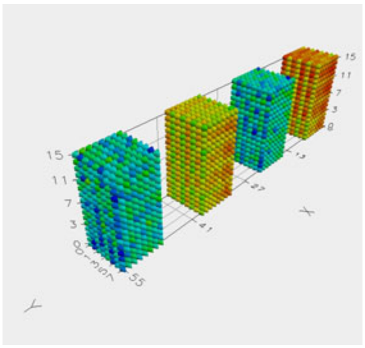

Fig. 7. Time in `MPI_Allreduce` for S3D 4K-core run on BGP with core-based topological layout

Listing 1.2. Topology configuration file for processor blocks

```
BEGIN_VIZ=4Px16Block
xdim=8,ydim=8,zdim=16
x=mod(rank,xdim)+16*floor(rank/1024)
y=mod(floor(rank/xdim),ydim)
z=mod(floor(rank/xdim/ydim),zdim)
END_VIZ
```

We wanted to break down the visualization further to display core level activity. We elected to use the mathematical expression topology definition system to display each of the four cores as its own point in a distinct node level topology. The result is shown in figure 7. Each of the four blocks represents the activity one of the four core ids laid out in the specified node level topology.

Discussing the internal topologies of each block is outside the scope of this paper. However an interesting high level phenomenon is immediately visible. In each node, overall, the cores are operating in pairs of high and low utilization. That is, for each chip, one core is spending significantly more time in the routine under observation (`MPI_Allreduce`) than the other. This core wise breakdown is likely related to the way individual cores are assigned to handle communication.

The formulae used to distinguish the topology by core is defined in listing 1.2. Note that this topology definition is only applicable to topologies in which the thread rank is the last to repeat.

There are numerous alternative thread-conscious, four dimensional layouts. The ideal layout will vary by the application selected and the topological behavior being observed. For example, we have also had success with grouping the threads or cores that comprise a single node and arranging each of these groups in the context of the greater node-level system topology.

By opening the node and thread layout to formulaic definition we have expanded the scope of topological performance visualization from machine dictated layouts to arbitrary node configurations. This is especially useful for mapping ranks to program domain decompositions which may have no direct relationship with the hardware topology.

3.3 GCRM/ZGrd Use Case: 3D Correlation Plot

The Global Cloud Resolving Model (GCRM) [1] is an atmospheric circulation simulator. We selected a 10240 core run of its ZGrd sub-application to demonstrate our visualization system's flexibility by duplicating ParaProf's existing

3D scatter plot functionality, which shows the correlation of four event/metric combinations specified in the UI. Three value are shown spatially and one using color.

The spatial dimensions and color of each rank are set by distinct event/metric pairs. For example in Figure 8 the values used to calculate the Y axis position of each point are determined by the exclusive time spent in `MPI_Allreduce`. As shown in Listing 1.3 the three spatial dimensions are associated with internally defined values. The event/metric pairs used to populate these values are specified by the user in the ParaProf UI. The restrictDim value set to 1 causes the 3D visualization to normalize dimensions, ensuring a cubic rendering space even if the different event/metric pairs use values with different orders of magnitude.

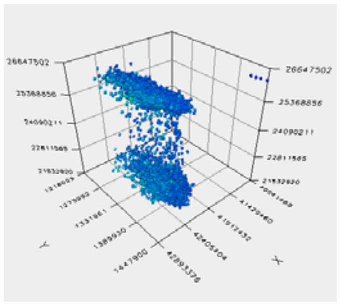

Fig. 8. 3D Correlation plot of 10240 core ZGrd run

Listing 1.3. Topology configuration file for scatterplot

```
BEGIN_VIZ=ScatterTest
restrictDim=1
x=(event0.val-event0.min)/
   (event0.max-event0.min)
y=(event1.val-event1.min)/
   (event1.max-event1.min)
z=(event2.val-event2.min)/
   (event2.max-event2.min)
END_VIZ
```

4 Related Work

The interest in parallel performance visualization truly began as parallel systems began to scale. The classic ParaGraph [13] used network topology to display performance of message passing programming. The Prism [18] programming environment for the CM-5 demonstrated the benefits of topological layout for display of debugging, data, and performance information for data parallel programs. Previously studied concepts and methods for parallel performance visualization [10,14,15] emphasize the importance of using various forms of semantics in effective presentation. Three-dimensional graphics techniques have also proven useful for scalable performance visualization [12,11], again with structural elements providing a necessary context for interpretation.

As the scale of parallel systems grew, the challenges of performance data size, analysis complexity, and presentation intensified. The Cube application provided by the Scalasca project is a performance analysis tool the features of which include visualization of performance data in a Cartesian topology layout. Cube populates this view based on the coordinates provided by Scalasca, collected from the MPI system at runtime.

The aforementioned tools, where topology based visualization is provided at all, generally support topological displays mediated by communication patterns or literal hardware layouts. In contrast ParaProf's 3d topology visualization system introduces a number of novel features to increase the scope of topological layouts which can be applied to performance data inputs, including site and application specific topologies. Unlike proprietary or architecture-specific tools the ParaProf 3d topology visualizer provides a consistent, portable interface for topological analysis regardless of platforms and programming models that produced the data. Additionally, the scope of the performance data made available by the TAU framework allows for topological analysis with respect to a wide array of metrics and program decompositions.

Significant work has been done by the Charm++ project[8] among others on topology mapping algorithms. These solutions generally focus on selection of ideal layouts for a given application and are complimented by diagnostic performance analysis with respect to the topological configuration.

There are some similarities in graphical presentation between our topological performance display and systems for configuring and monitoring HPC clusters. These management tools are often focused on a specific subset of hardware and generally are more concerned with static evaluation of system topologies than post-mortem analysis. Such applications include XT3D[20] which presents 3d visualizations of system topologies on Cray clusters.

5 Conclusion

Parallel performance visualization can be a useful technique for better understanding performance phenomena. However, it is important to integrate the capabilities within a performance analysis framework. This paper describes a performance visualization design methodology and its incorporation in the TAU ParaProf tool. Its initial implementation concentrates on topology-oriented layout and examples are given for the Sweep3D and S3D applications.

However, the methods we present for visual layout and UI design are more broadly applicable. To demonstrate their versatility, we have recently recreated ParaProf's event correlation view. In general, our goal is to allow the user the full benefit of incorporating their concepts of visual presentation and semantics to improve performance understanding.

Acknowledgements. This research is support by the U.S. Department of Energy, Office of Science, under contract DE-SC0001777. Resources of the Argonne Leadership Computing Facility at Argonne National Laboratory were utilized in the work.

References

1. Global cloud resolving model (gcrm), `https://svn.pnl.gov/gcrm`
2. Paraview, `http://www.paraview.org/`

3. Visit, `https://wci.llnl.gov/codes/visit/`
4. Visualization toolkit (vtk), `http://expression-tree.sourceforge.net/`
5. Math expression string parser (mesp) (2004), `http://expression-tree.sourceforge.net/`
6. The ascii sweep3d code (October 2006), `http://www.llnl.gov/ascibenchmarks/asci/limited/Sweep3D/asciSweep3D.html`
7. Bell, R., Malony, A.D., Shende, S.: A portable, extensible, and scalable tool for parallel performance profile analysis. In: Proc. EUROPAR 2003 Conference, pp. 17–26 (2003)
8. Bhatele, A., Kale, L.V., Chen, N., Johnson, R.E.: A Pattern Language for Topology Aware Mapping. In: Workshop on Parallel Programming Patterns, ParaPLOP 2009 (June 2009)
9. Chen, J., et al.: Terascale direct numerical simulations of turbulent combustion using S3D. Computational Science and Discovery 2(1), 15001 (2009)
10. Couch, A.: Categories and Context in Scalable Execution Visualization. Journal of Parallel and Distributed Computing 18(2), 195–204 (1993)
11. De Rose, L., Pantano, M., Aydt, R., Shaffer, E., Schaeffer, B., Whitmore, S., Reed, D.: An approach to immersive performance visualization of parallel and wide-area distributed applications. In: Proceedings of the Eighth International Symposium on High Performance Distributed Computing, 1999, pp. 247–254 (1999)
12. Hackstadt, S., Malony, A., Mohr, B.: Scalable Performance Visualization of Data-Parallel Programs. In: Scalable High-Performance Computing Conference, pp. 342–349 (May 1994)
13. Heath, M., Etheridge, J.: Visualizing the Performance of Parallel Programs. IEEE Software 8(5), 29–39 (1991)
14. Heath, M., Malony, A., Rover, D.: Parallel Performance Visualization: From Practice to Theory. IEEE Parallel and Distributed Technology: Systems and Technology 3(4), 44–60 (1995)
15. Heath, M., Malony, A., Rover, D.: The Visual Display of Parallel Performance Data. Computer 28(4), 21–28 (1995)
16. Jagode, H., Dongarra, J., Alam, S., Vetter, J., Spear, W., Malony, A.D.: A Holistic Approach for Performance Measurement and Analysis for Petascale Applications. In: Allen, G., Nabrzyski, J., Seidel, E., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2009. LNCS, vol. 5545, pp. 686–695. Springer, Heidelberg (2009), `http://dx.doi.org/10.1007/978-3-642-01973-9_77`
17. Shende, S., Malony, A.D.: The TAU Parallel Performance System. SAGE Publications (2006)
18. Sistare, S., Allen, D., Bowker, R., Jourdenais, K., Simons, J., Title, R.: A scalable debugger for massively parallel message-passing programs. IEEE Parallel and Distributed Technology: Systems and Applications Distributed Technology: Systems and Applications 2(2), 50–56 (1994)
19. Traff, J.: Implementing the mpi process topology mechanism. In: SC Conference, p. 28 (2002)
20. Yanovich, J., Budden, R., Simmel, D.: Xt3dmon 3d visual system monitor for psc's cray xt3 (2006), `http://www.psc.edu/~yanovich/xt3dmon`