LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-CONF-525411

Interactive Visualizations for Performance Analysis of Heterogeneous Computing Clusters

A. Landge, J. Levine, P. Bremer, M. Schulz, T. Gamblin, A. Bhatele, K. Isaacs, V. Pascucci

January 30, 2012

GPU Technology Conference
San Jose, CA, United States
May 14, 2012 through May 17, 2012

Disclaimer

Interactive Visualizations for Performance Analysis of Heterogeneous Computing Clusters

Aaditya Landge[1], Joshua A. Levine[1], Peer-Timo Bremer[2], Martin Schulz[2], Todd Gamblin[2], Abhinav Bhatele[2], Katherine Isaacs[3], and Valerio Pascucci[1]

[1]Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, Utah,
{aaditya, jlevine, pascucci@sci.utah.edu}
[2]Lawrence Livemore National Laboratory, Livermore, CA,
{ptbremer, schulzm, tgamblin, bhatele@llnl.gov}
[3]University of California, Davis, Davis, CA,
{keisaacs@ucdavis.edu}

***Abstract*—Performance analysis is a vital step in identifying execution bottlenecks to help target optimizations. This analysis is derived from observations of performance data collected from the computing hardware. Data obtained from computing clusters is necessarily complicated because its collection involves multiple interacting nodes, potentially with several cores each, as opposed to just a single serial execution. Further, heterogeneous clusters, often in the form of nodes combining one or more CPUs working together with several GPUs, are becoming more commonplace and thus the computation involves additional layers of complexity. These characteristics pose a serious challenge to the analysis and improvement of application performance. We present a tool that assists performance analysis by visualizing performance data with the help of various linked views. In the following report, we present the description of these various views and the highlight the advantages of their use with the help of a case study.**

I. Introduction

Supercomputers and clusters are used for executing immensely compute intensive applications. These applications exploit the parallel processing capabilities of these resources to decrease application runtime. But making fullest use of the hardware remains a difficult task for such application developers due to the complex nature of these parallel programs. In most of the cases, this results in programs that underutilize the systems and do not take advantage of the full potential of these clusters or supercomputers. In order to improve the efficiency of these programs one has to understand the performance behavior of the application. Traditionally, profilers have been used to collect performance data of applications. Analysis of this data is then done to understand the code behavior and identify performance bottlenecks.

For a single node system, this data is the hardware performance counters collected from the CPU. In the case of a heterogeneous system, a CPU is working along with one or several GPUs. Performance data must also collected from the GPUs along with that of the CPUs. Since the CPUs and GPUs are interacting with each other during the lifetime of the application, understanding this data can become complicated even at this small scale.

In the case of heterogeneous computing clusters, performance data is collected from the CPUs and GPUs of every node present in the cluster. Since all the nodes in the cluster interact with each other and work together during the execution of the application, the analysis of this data is confounded by multiple levels of interaction. Compared with the single node case, a supercomputer can have hundreds of thousands of nodes. Thus, the data collection and analysis must be tuned to handle the massive volumes of data as well.

To cope with these challenges, various profiling tools have been developed. Conventional profilers collect various hardware performance counters from the hardware. These counters are then consolidated on the basis of function calls and presented to the user in the form of tables or charts. But this type of data representation may be unintuitive when one is dealing with systems at the exascale. As a result, there is a severe need for specialized tools that can assist performance analysis by analyzing and visualizing this data in much more intuitive ways.

Schulz et al. have identified three domains of performance data most familiar to the user: (i) the application domain, (ii) the hardware domain, and (iii) the communication domain [1]. The philosophy states that taking data from each of these domains and projecting, visualizing, and correlating it to the other domains can give valuable insights into the behavior of parallel application codes. Following is a short summary of the way they describe these domains:

The **application domain** is the output of the application and is ideally the most easily understood by the application developer. Often the application output can be represented by some mesh, matrix, or graph. Performance data can then be mapped to this representation. Doing so, one can correlate some phenomenon occurring in the output of the application to the performance data which may provide insights or highlight problems in some sections of the application code.

The **hardware domain** is spatial representation of the physical hardware of the computing system in terms of its nodes or computing cores. Performance data can then be shown in the context of the entire system and patterns in hardware behavior could be easily understood. Mapping the application data to the hardware domain can help in understanding how parts of the application get attributed to certain nodes or computing cores.

The **communication domain** consists of a general graph that can represent the communication that occurs between the various MPI processes during the execution of the application. Understanding the communication patterns with respect to the hardware and application domain can lead to the identification of performance bottlenecks caused by communication between the nodes or MPI processes.

In the present project, we attack the challenges mentioned in the earlier section faced in performance analysis of clusters and supercomputers by developing a tool that provides various visualizations of performance data collected from clusters and supercomputers. The tool visualizes the performance data in (i) multiple domains, (ii) multiple granularities and, (iii) provides visualization for combined CPU-GPU performance data for heterogeneous clusters. These visualizations are interactive in nature and are linked together which helps understand the performance behavior of heterogeneous parallel applications in an intuitive manner.

We take inspiration from the domains described by Schulz et al. and provide visualizations for the application and hardware domains. Standalone views for each of these domains provides good insights about the respective domains, but they may lack in showing the correlations and mapping between these domains. As presented in [1], drawing correlations between the domains is a much more intuitive way of understanding performance data. In this regard, we bring about these correlations by linking the visualizations together and making each of them interactive. By linking these views, one can see the corresponding data in the other views when a particular selection is made in any one of the views.

Inspired by the ”focus plus context” philosophy, which is often used in information visualization [3], these views help to visualize the data at various granularities in the appropriate context. For example, one can view the performance data at the cluster level and understand the behavior of nodes with respect to each other; or one can compare the performance of a GPU on one node to a GPU of some other node without losing the context of the entire cluster.

By providing a combined visualization of the CPU and GPUs, one can get a good understanding of the behavior of the application in this heterogeneous environment. Such combined analysis is beneficial as one can understand the behavior of CPU and GPUs with respect to each other and armed with this knowledge redesign code to make best use of both the processing units.

The contributions of the project are summarized as follows:

- We extend the idea of multiple data domains [1] and introduce visualizations for the complex performance data in the application and hardware domain;
- Correlations between the hardware and application domain are shown by coupling the visualizations of these two domains through interactions;
- Based on the above two, we focus on presenting visualizations for the analysis of the complex behavior of heterogeneous computing clusters; and
- We highlight the various insights that can be obtained by the usage of such interactive linked visualizations and how it assists performance analysis with the help of a case study.

II. Interactive Linked Visualizations

As performance data can be visualized in the application and hardware domain, we have developed views that represent each of these domains. The application domain is visualized using the Application view. The hardware domain is considered at multiple granularities which we represented using the cluster level and node level views. Performance data is then mapped on these domains and visualized in these interactive linked views. As the views are linked to each other, data from one view can be easily correlated to data visualized in other views. This enables viewing the same data in the context of the application domain and the hardware domain. Viewing the data in these multiple contexts simultaneously gives a good intuition about the behavior of the application, which standalone views would fail to provide.

Since each of these views is interactive, they allow the user to have multiple channels of interaction with the performance data. The user has the freedom to interact with the data in the application or hardware domain or at the granularity of the cluster level or at the node level or at the CPU or GPU level. In switching between these granularities the user never loses track of the global context of the computing cluster since views showing the

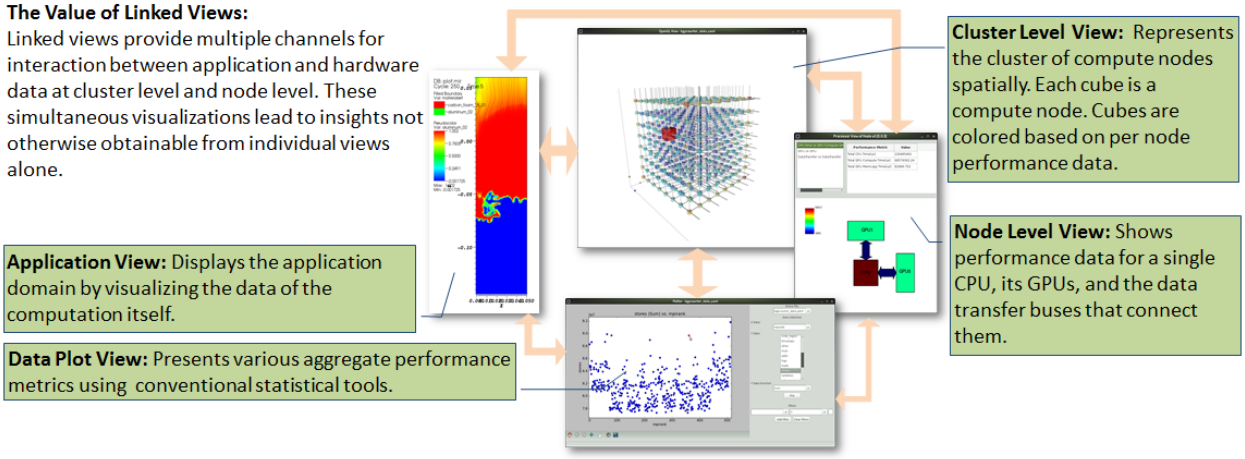

Fig. 1. **Interactive Linked Views:** The various views presented by the system. The **Application View** represents the application domain. The **Cluster Level View** represents the spatial representation of the hardware domain. The **Data Plot View** consists of conventional statistical plot views. The **Node Level View** shows the internals of a heterogeneous compute node present in the cluster. The arrows show the various linkages between these views.

data at different granularities are always linked together and visible to the user all the time.

For heterogeneous systems, we present a view that enables visualizing the CPU and GPU performance data together. This type of view helps optimize the performance of the applications taking advantage of both the CPUs and GPUs. It also helps compare GPUs from the same node as well as from other nodes. This view is also linked with the other views and domains. This linking enables the user to understand and relate parts of the application to specific GPU/s in the nodes of the cluster.

All these features combine to form a powerful tool for visualization of performance data of heterogeneous computing clusters that can give valuable insights about the parallel code behavior in an intuitive way. We shall now describe each of the views provided by the tool in detail.

a) Application View: This view represents the application domain. The application output is decomposed as a suitable mesh, graph, or matrix. Performance data can then be mapped onto this decomposition to gain valuable insight. As the application developers are most familiar with this representation, it becomes simple to relate the performance characteristics to the application output. Also, it may expose any application data specific performance issues in the application. For instance, a particular node may take some more time to perform computations which can be attributed to some characteristic of the data it is processing.

b) Cluster View: The hardware domain is spatially represented in this view. The computing system is shown as a grid of cubes, where each cube represents a computing node or computing core. Various performance metrics from the CPU or from the GPUs are aggregated for each node and shown on this grid. Visualizing data in this fashion helps in identifying patterns in behavior of nodes with respect to that performance metric. For example, nodes can be colored based on total compute time of CPU or GPUs which enables easy identification of time consuming nodes, also the spatial representation gives an idea about the performance of neighboring nodes which can help in better understanding about the application behavior.

Application data can also be mapped to this domain to provide insights into how the application decomposes the problem domain on to the physical hardware. For example, one can easily represent the size, shape, or other characteristics about the chunk of input data processed by each node to get an idea about the load balancing.

c) Data Plot View: This view is a conventional statistical view incorporating plots like scatter plots and histograms where the user can control the plotting axes. Performance metrics can be plotted against each other for comparisons and gaining insights. Also, various application specific data can be plotted against performance data. For example, number of cells processed per node vs. MPI rank.

The Cluster view and Application view both highlight qualitative aspects of the data, while the Data Plot view can show complimentary quantitative behavior. Also, this view gives the user the freedom to compare data entities from any of the domains.

d) Node Level View: This view represents the internals of the heterogeneous computing node enabling visualizing the performance data at finer granularity. The CPUs and the GPUs within a node are shown and a combined visualization of their performance metrics can be made. As each CPU and GPU is visualized separately,

a comparison between individual GPUs and CPUs is also possible. Performance of the data transfer buses between the CPUs and GPUs can also be visualized. In general, an understanding of the heterogeneous system can be made which can help in making optimum use of both CPUs and GPUs.

This view also allows comparing multiple nodes at a finer level. For example, one can compare a CPU or GPU from one node with a CPU or GPU of another node. This can be done by opening multiple node level views corresponding to different nodes.

Interactions: All the aforementioned views and the linkages amongst them are shown in Figure 1. An example of the interaction can be seen in Figure 3. The application view in this figure, shows the way in which the input image is partitioned and processed by the various nodes in the cluster. The rainbow color coding is used to represent the time taken by each node to compute the part of the image assigned to it; going from red that took the maximum time to blue that took the least time. The user has clicked on the partition of the image that has taken the maximum time in the Application View, the corresponding node in the Cluster view has been highlighted by a larger red cube. At the same time, the data point pertaining to the MPI rank of that node has been highlighted in red in the Data Plot View. Right-clicking the highlighted red cude in the Cluster View opens up the Node Level View for that particular node to show the performance statistics of the CPUs and GPUs present in that node.

Such type of interactions are possible amongst all the views. For example, a click in the Cluster View will highlight corresponding areas in the Application and Data Plot views. This enables the user to attack the problem of perform analysis from both - the application domain as well as the hardware domain and understand the correlations between these two domains, giving better insights into the application behavior.

III. Case Study

Application Description: We try to understand the behavior of a Hybrid CPU-GPU Solver for Gradient Domain Processing of Massive Images [2]. This application basically takes a number of images and seamlessly stitches it into a gigapixel panoramic image. This particular process is done using MPI and CUDA.

The algorithm decomposes the input images into 1024×1024 pixel sized tiles that are distributed across all MPI processes. The number of MPI processes is decided on the available nodes of the computing cluster. Each MPI process then processes some tiles on a CPU or on the GPUs depending on available resources on the particular node.

This application was executed on an HP cluster located at the Scientific Computing and Imaging Institute, University of Utah. Fifty nodes of the cluster were used for the experiments. Each node consists of a Intel Xeon X5550 2.6Ghz Processor having 8 cores and 2 NVIDIA Tesla T10 GPUs. The cluster uses an Infiniband network for communication between the nodes.

Fig. 2. **Above:** Input to the application consisting of photographs that are used to construct the panorama. **Below:** The 3.27 gigapixel panorama generated as the output by the application.

The dataset used for the application runs is a 126,826×29,633, 3.27 gigapixel panorama decomposed into 124×29 spatial tiles. The application takes about 5430 seconds to generate the output for this input set on the specified cluster.

Data Collection: The performance data was collected by direct instrumentation of the source code of the application. Code was inserted at various parts of the application to collect the desired data. Data was collected specifically for each tile that was processed by the application. Collecting the data on a per tile basis allowed us to represent the performance data in the application domain as opposed to at a per-function call granularity. The CPU performance metrics were collected using PAPI and the GPU performance metrics were collected using CUPTI provided in the NVIDIA CUDA Tools SDK.

The **application domain** for this case study was chosen as a 2D rectilinear grid to represent the output of the application which is a panoramic image in this case. Each cell in this grid represented a spatial image tile of the panorama image which was processed by the assigned MPI process on a CPU or on a GPU.

The **hardware domain** was chosen as 10×5 grid of cubes to represent the 50 node cluster. Each of the cubes in this grid represented a node of the cluster consisting of the CPU and the two GPUs.

Experiment 1: In this experiment seen in Figure 3, we map the total time taken by each node to the application domain and present it in the application view. This automatically shows us the data partitioning done by the algorithm to distribute work between the nodes. Immediately after looking at this view, we observe that

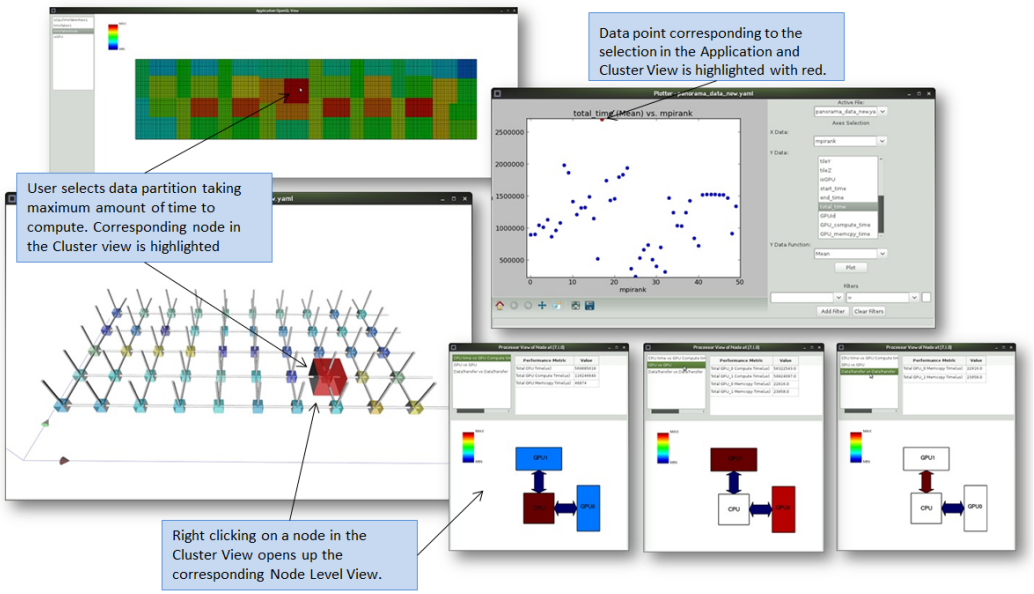

Fig. 3. **Experiment 1:**
Application View: Data partitioning and assignment done by the application on the cluster. The coloring of the partitioned is done using a rainbow color map based on the time taken by each node to complete the computation of the partition assigned to it. A click on any of the partition automatically highlights the node in the other views. **Cluster Level View:** Each node colored using a rainbow color map to show the time taken by each node to complete the computation. A selected node is highlighted with an enlarged red cube. **Data Plot View:** Plot of total time of each node vs. the MPI rank. The data point for the selected node is highlighted in red. **Node Level Views of the selected node(from right to left):** CPU time vs. Overall GPU compute time, GPU0 compute time vs. GPU1 compute time, Data Bus0 transfer time vs. Data Bus1 transfer time.

the node doing computation on the center of the image takes the maximum time. Also, the nodes that process the parts of the image near the border take less time. By looking at the output image and this view, one can infer that since there is a lot of detail in the center of the image, the solver takes more iterations to process that region of the image whereas since there is less detail near the borders, the solver takes less time. We can also see that load is not equally balanced as some nodes take less time whereas some take considerably longer time. This suggests a superior load balancing for this application may need to take into account the complexity of the tiles being processed.

Though the Application view presents insight about the application behavior, some questions are better answered using a coupled visualization that combines the Cluster view. For example, which exact node took the maximum time; what was its spatial location in the cluster; how much time was taken by its neighbors; and what was the behavior of the hardware performance metrics for that node compared to the others are best considered using the Cluster view along with the Application view. Selecting the node that takes maximum time in the application view automatically highlights the corresponding node in the Cluster view. Its spatial position in the grid can be easily seen as well as time taken by its neighbors is also easily observed. One can then color the nodes on the basis of some other performance metric to get further insights.

The Application and Cluster view give the qualitative aspect of the data, so we look at the Data Plot view to get a notion of the actual quantities. In the Data Plot view we can see that the selected node is highlighted in red. One can easily identify its MPI rank and get an idea of the actual time the node took to finish its computation. Since the interaction is two-way, one also can select some other MPI rank in this view and that corresponding node will be highlighted in the Cluster and Application views.

The Cluster view and Data Plot view present data at a coarser granularity. All the performance metrics for a node are aggregated over its CPUs and GPUs. To look at the data at a much finer granularity one can click on the node that opens up the Node Level view. This view enables the direct comparison of the CPU, the GPUs and data transfer buses within this node. One can view the time taken by the CPU and the GPUs, time taken by individual GPUs and time taken by individual data transfer buses.

This heterogeneous view helps us understand the utilizations of the CPU and GPUs on that node. One can then easily infer if the bottleneck is the CPU, one of

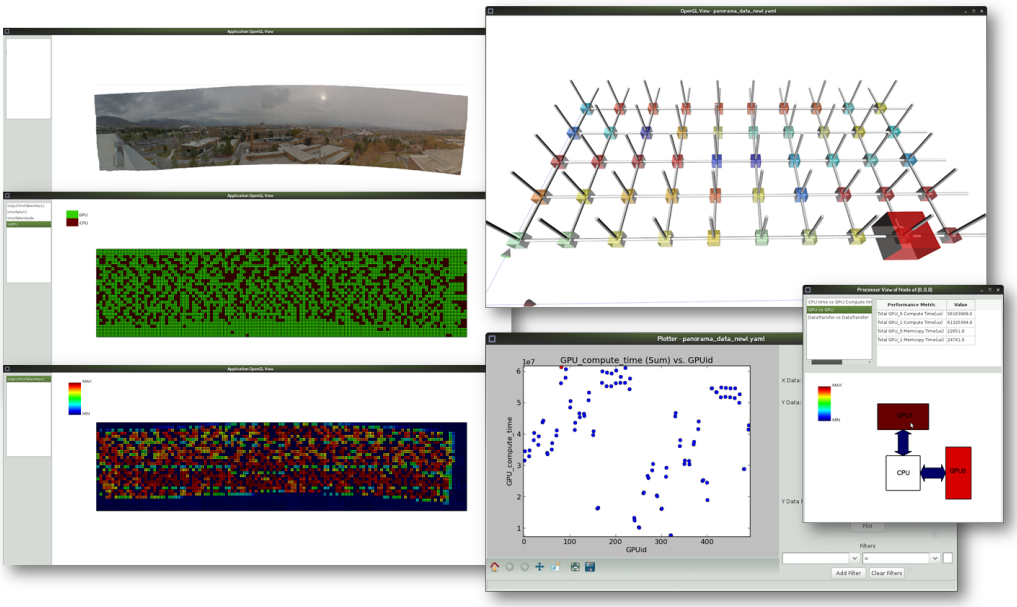

Fig. 4. **Experiment 2:**
Application View (top): Final image generated by the application. **Application View (middle):** Image tiles, colored on the basis of if they were processed by CPUs or GPUs. The green tiles correspond to those processed on the GPU and the red tiles correspond to the tiles processed on the CPU. **Application View (bottom):** Time taken by the tiles processed on GPUs represented with a rainbow color map with red being maximum and blue being minimum. **Cluster View:** Each node colored by GPU compute time with the help of a rainbow color map. Node with max GPU compute time is selected and highlighted with an enlarged red cube. **Data Plot View:** Plot of GPU compute time vs. GPU ID. **Node Level View:** Each GPU in the node colored by the GPU compute time.

the GPUs or the data transfers. Optimizations can then be targeted on the basis of these observations.

From this experiment one can see that certain key insights are so easily inferred with the help of these interactive views and how correlating the data between different domains provides better intuition of the performance data.

Experiment 2: In this experiment as seen in Figure 4, we visualize the way work is distributed amongst the CPUs and GPUs in the cluster. In the application view we can see that how the tiles in the images are processed on CPUs and GPUs. The green tiles correspond to those processed on the GPU and the red tiles correspond to the tiles processed on the CPU. This gives good understanding of how the load has been balanced between the CPUs and GPUs of the cluster. About 70 percent of the tiles were processed on the GPUs and the rest on the CPUs. This view also shows which exact tiles were processed on the CPUs and the GPUs. With this information, one can design better scheduling algorithms that can schedule tiles having higher detail on to the GPUs, as the GPUs are comparitively much faster at processing the tiles than the CPUs.

In the other application view, the time taken by the GPU to process the tiles is mapped on the application domain. The coloring is done with the help of a rainbow color map where red corresponds to high values and blue to values. One can see that the tiles that contain higher levels of detail from the image consume more time.

In the Cluster view we color the nodes on the basis of total GPU compute time to identify the node that takes the maximum GPU compute time. Since, several nodes have more or less the same higher valued color it becomes difficult to identify the node that takes the maximum time. So, we take the help of the Data Plot view to get the GPU ID of the GPU that has the maximum total GPU compute time. The GPU ID is a unique identifier given to every GPU in the cluster. Selecting this GPU ID in the Data Plot view, the corresponding node is highlighted in the Cluster view and the Node Level view.

From the node level view we can then take a look at the other performance metrics for this GPU to investigate why it takes the maximum amount of time. We can also do a comparison of this GPU with the GPUs of its neighboring node or any other node to get better insights. Thus the user can go deeper into the analysis from a higher granularity to a lower granularity to understand performance bottlenecks.

From this experiment its evident that making use of these interactive linked views is helpful in understanding the behavior of heterogeneous applications. Load balanc-

ing between the CPUs and GPUs can be well understood and optimizations to improve this balance can be done to improve application efficiency. Also, the combined visualization of the CPU and GPU performance data gives deeper understanding of such complex parallel codes.

IV. Discussions

In this project we have developed novel visualizations for visualizing and analyzing performance data. The various linkages between these views and the interactivity they provide give a good intuition of the performance data. We have successfully shown that valuable insights about heterogeneous application behavior can be achieved with the use of interactive linked visualizations that have been developed in this project. These valuable insights help target optimizations resulting in faster execution of parallel applications.

In the future, we would like to extend these views to also show data from the Communication domain which has been left out in the current project. This view should be able to show the network topology of the cluster and show the communication data. The node level view can be enhanced further to show thread specific data. Also, data can be visualized per CPU core and GPU cores adding a finer level of granularity. Novel visualization techniques need to be added to the cluster level view to view larger clusters having hundreds of thousands of nodes.

References

[1] Martin Schulz, Joshua A. Levine, Peer-Timo Bremer, Todd Gamblin, Valerio Pascucci, *Interpreting Performance Data Across Intuitive Domains*. International Conference on Parallel Processing, Sept. 2011.

[2] Sujin Philip, Brian Summa, Peer-Timo Bremer, Valerio Pascucci, *Parallel Gradient Domain Processing of Massive Images*. Eurographics Symposium on Parallel Graphics and Visualization, Apr. 2011.

[3] Stuart K. Card, Jock D. Mackinlay, Ben Shneiderman, *Readings in Information Visualization: Using Vision to Think*. California: Academic Press, 1999.

Prepared by LLNL under Contract DE-AC52-07NA27344