

Enabling Advanced Visualization Tools in a Web-Based Simulation Monitoring System

Emanuele Santos, Julien Tierny, Ayla Khan, Brad Grimm, Lauro Lins, Juliana Freire, Valerio Pascucci, Claudio Silva, Scott Klasky, Roselyne Barreto, et al.

▶ **To cite this version:**

Emanuele Santos, Julien Tierny, Ayla Khan, Brad Grimm, Lauro Lins, et al.. Enabling Advanced Visualization Tools in a Web-Based Simulation Monitoring System. IEEE Conference on eScience, 2009, Oxford, United Kingdom. 10.1109/e-Science.2009.57 . hal-01211179

HAL Id: hal-01211179
https://hal.science/hal-01211179

Submitted on 7 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Enabling Advanced Visualization Tools in a Web-Based Simulation Monitoring System

Emanuele Santos, Julien Tierny, Ayla Khan, Brad Grimm, Lauro Lins, Juliana Freire, Valerio Pascucci, Cláudio T. Silva
Scientific Computing and Imaging (SCI) Institute
University of Utah
Salt Lake City, UT, USA
{*emanuele, jtierny, ayla, bgrimm, lauro, juliana, pascucci, csilva*}*@sci.utah.edu*

Scott Klasky, Roselyne Barreto, Norbert Podhorszki
Oak Ridge National Laboratory
P.O. Box 2008, Oak Ridge, TN 37831, USA
{*klasky, barreto, pnorbert*}*@ornl.gov*

***Abstract*—Simulations that require massive amounts of computing power and generate tens of terabytes of data are now part of the daily lives of scientists. Analyzing and visualizing the results of these simulations as they are computed can lead not only to early insights but also to useful knowledge that can be provided as feedback to the simulation, avoiding unnecessary use of computing power. Our work is aimed at making advanced visualization tools available to scientists in a user-friendly, web-based environment where they can be accessed anytime from anywhere.**

In the context of turbulent combustion for example, visualization is used to understand the coupling between turbulence and the turbulent mixing of scalars. Although isosurface generation is a useful technique in this scenario, computing and rendering isosurfaces one at a time is expensive and not particularly well-suited for such a web-based framework. In this paper we propose the use of a summary structure, called *contour tree*, that captures the topological structure of a scalar field and guides the user in identifying useful isosurfaces. We have also designed an interface which has been integrated with a web-based simulation monitoring system, that allows users to interact with and explore multiple isosurfaces.

I. Introduction

In many scientific disciplines, the use of simulations is commonplace. As computing power and storage become more abundant, these simulations become more complex and data intensive. Simulations of turbulent lifted flames [**?**] for example, can take millions of CPU-hours and result in multiple terabytes of data. Running these simulations on resources such as the TeraGrid is very costly, and due to the high demand, cycles are scarce. Thus it is important that scientists be given the ability to analyze these results as they are computed, which can not only lead to early insights, but also to useful knowledge that can help them steer the simulation, to remedy potential errors and avoid wasting cycles, or zoom into areas of potential interest.

But doing so poses important challenges. First and foremost, it is not feasible to move the simulation results around since the I/O costs are prohibitive. It is thus important to push as much of the analysis and visualization as possible to the high-performance computing (HPC) environment. This requires a tighter integration between the simulations and analysis, and the creation of workflows that support both tasks in an HPC environment. Another challenge comes from the complexity of the required analysis. Because complex simulations deal with large numbers of parameters, a potentially infinite number of summaries can be generated to help users explore different aspects of the results. Because computing these summaries is itself an expensive task, both due to the size of the raw data and complexity of summarization techniques, there is an increasing need for efficient techniques that help users quickly identify useful regions of the data and specific summaries to explore.

In this paper, we explore a web-based analysis and visualization solution to this problem in the context of turbulent combustion simulations. To understand the coupling between turbulence and the turbulent mixing of scalars, such as temperature and species concentrations, it is important to generate isosurfaces that represent those interactions. Isosurfaces are one of the most widely-used visualization techniques and efficient to compute: the complexity of standard marching cubes, the most popular isosurface algorithm, is linear [**?**]. Although it is possible to efficiently generate an isosurface for a given isovalue, computing and rendering a large number of isosurfaces, as required in this scenario, is expensive and incurs a high network overhead for transferring the results to a web browser. This makes such a solution impractical for a web-based analysis tool. To address this problem, we propose the use of a summary structure, called *contour tree*, that captures the topological structure of a scalar field and guides the user at identifying useful (important) isosurfaces, see Section III. We have also designed a user interface that allows users to interact with and effectively explore multiple isosurfaces (see Section IV). By applying the contour tree algorithm to find isosurface values in situ with the computation (Section III), it is possible to selectively browse through multiple visualizations and quickly understand the complex data being generated during the

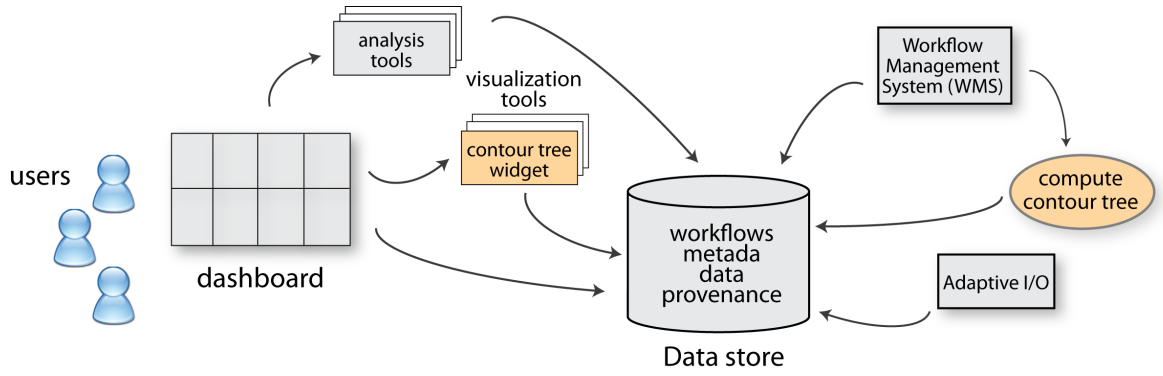


Figure 1. eSimMon system architecture.

simulation. The contour tree tool has been integrated with the eSimMon dashboard system [**?**], which provides an environment for scientists to monitor, manage and explore simulation results (see Section II-A). In Section V, we present a case study where we show that integrating the dashboard with the interactive contour tree tool leads to an effective and efficient means to explore the turbulent combustion simulation results.

II. System Overview

We have implemented and integrated our techniques with the eSimMon system. As Figure 1 shows, the central component of the eSimMon system is a *Data store*: a database that stores all of the information and collects all the provenance information about the simulations, including the lineage of data products. The Workflow Management System (WMS) orchestrates the jobs and populates the data store with the job information. The processes launched by the workflow management system may use the *Adaptable I/O System* (ADIOS) [**?**], which is a componentization of the I/O layer. The *eSimMon Portal* allows scientists to access their simulations from any location using the Web through a browser. We have added the process to compute the contour tree and the isosurface images so it could be launched by the WMS and developed a graphical interface which was added to the set of tools available through the *eSimMon Portal*.

A. The eSimMon Dashboard System

Monitoring petascale simulations typically requires that a diverse group of scientists look at the same massive amount of data from different angles. The eSimMon dashboard provides an overview of the status of a simulation. It is a common access point to the simulation data for many different types of users, including simulation scientists, theoretical scientists, experimentalists, performance analysts, and visualization experts. These researchers not only have different expertise, but they also use different tools to monitor, analyze and visualize their data. The purpose of the eSimMon dashboard is to facilitate management, analysis, sharing and visualization of simulation data. In other words, the goal is to provide a single, easy-to-use graphical user interface for several scientists to converge and collaborate on. To tackle ease of use, the workflow-dashboard system attempts to hide implementation details from its users and allow them to focus on scientific discovery. It does so using Web 2.0 technology [**?**] on the front end and provenance tracking in the back end during workflow execution.

The eSimMon is composed of two main sections: machine monitoring and simulation monitoring. The machine monitoring (home) page displays job queues from available U.S. Department Of Energy and National Energy Research Scientific Computing computers. In this view, users can also grant others access to their simulations runs. Thereafter, they can view the status of their runs (eligible, running, or blocked) as well as of their collaborators. Users also see a list of their past runs. From this first page, scientists can access the simulation monitoring section, shown in Figure 2, by clicking on a specific run or shot. For a running job, they see images of variables updating themselves as they are being generated by the workflow. When a job is no longer running, images from all time stamps are combined into a movie instead. In this latter case, users have more options to visualize and manipulate their data. They can annotate movies or make electronics notes on simulations. Other capabilities include visualization of the data as movies, or vector graphs, provenance information (e.g., full path of the raw data), downloading of the processed and/or raw data, and visualization of the source code or environment information (system provenance). There are different types of analytics tools currently integrated in the dashboard. These tools are built-in or incorporated as hooks into back end analysis software. 3D modules are also being integrated to provide more complex visualizations and interactivity.

The provenance information is key to link processes, output data and input data [**?**]. The recorded information in our system includes the history about all data transformations (lineage of data), all operations executed (process provenance), and environment information combined with source code of executed simulations (system provenance) and all actions of the users on the data (activity provenance) [**?**].

Provenance allows users to analyze and visualize the data by focusing on the scientific variables calculated in

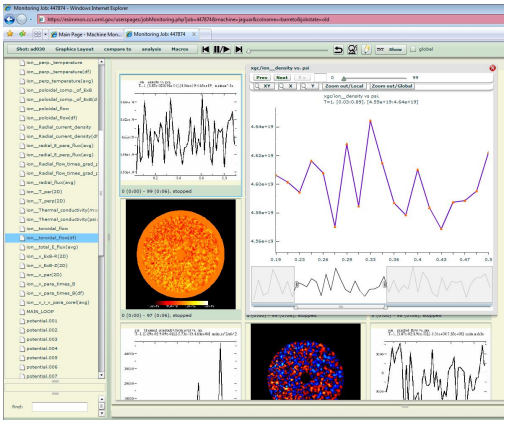


Figure 2. eSimMon dashboard. Tree on the left shows the scientific variables that can be dragged onto the canvas and shown as movies. The right hand side window is a vector graphics of a 1D variable in Flash that can be edited.

the simulation, not by filename(s). To accurately connect user's actions and requests on the front-end to simulation data eSimMon uses the provenance recorded in the dashboard database during the simulation monitoring workflow [**?**] execution. The recorded information includes the history about all data transformations (lineage of data), all operations executed (process provenance), and environment information combined with source code of executed simulations (system provenance) and all actions of the users on the data (activity provenance) [**?**]. The workflow records the metadata in a MySQL database which is later queried by the dashboard to access files on disk. Provenance tracking is key in taking the scientist's focus away from files to actual science. From the dashboard, users see a tree view of scientific variables generated by the simulation as shown on the left hand side of Figure 2. By simply dragging and dropping variables from the tree view to the main canvas, they can see that particular entity evolving through time in form of a flash movie. Users do not need to know or track which raw data file(s) was used to generate that movie. The link is made automatically by the dashboard.

III. The Contour Tree

The contour tree is an efficient data structure that captures the topological structure of a scalar field. Thanks to simple, robust and fast algorithms [**?**], it has a wide spectrum of applications in scientific visualization, such as seed-set computation for fast isosurface extraction [**?**], topologically clean isosurface extraction [**?**] and automated transfer function design [**?**].

In this framework, we use the contour tree as an efficient indexing key to quickly access isosurfaces and query them in a flexible manner. This section details the formal definition of the contour tree as well as the simplification and isosurface query processes.

A. Definition

The contour tree is a special case of the more general concept of a *Reeb graph* [**?**]. Let $f : \mathbb{M} \rightarrow \mathbb{R}$ be a scalar field defined on a manifold $\mathbb{M}$. One fundamental way to study the scalar field f is to extract its level sets. For a given scalar w, the level set $L(w)$ is defined as the inverse image of w onto $\mathbb{M}$ through f, $L(w) = f^{-1}(w)$. We call each connected component of the level set $L(w)$ a *contour*.

One aspect that is well understood in Morse theory [**?**] is the evolution of the homology classes of the contours of f while w changes continuously in $\mathbb{R}$. The points at which the topology of a contour changes are called *critical points* and the corresponding function values are called *critical values*. If all the critical points of f are non-degenerate and have distinct values, then f is a *Morse function*.

The Reeb graph $\mathcal{R}(f)$ of f is the quotient space induced by the equivalence relation *"two points p_1 and p_2 are equivalent if they belong to the same contour of f"* [**?**]. Adjacent contours are mapped in the Reeb graph to adjacent nodes and distinct contours are mapped to distinct nodes. Notice that branching in $\mathcal{R}(f)$ only occurs at critical values of f and we call the corresponding nodes *critical nodes*.

In other words, one can see the Reeb graph of a scalar field f as a continuous contraction of f contours as w changes continuously over $\mathbb{R}$, as illustrated in Figure 3. A Reeb graph is called a *contour tree* when it has no loops. This is guaranteed in particular if $\mathbb{M}$ is simply connected. In practice, our input data is given as a regular grid where each vertex is associated with a scalar field value. As regular grids are by definition simply-connected, the Reeb graph will always be a contour tree. Then, the efficient algorithm presented by Carr et al. [**?**], with $O(nlog(n))$ time complexity (where n is the number of vertices), can be used.

As illustrated in Figure 3, the contour tree can provide useful visual insights on the structure of the scalar field and can help the users understand their data. However, with real-life simulation data, the number of critical points is usually very high and so is the number of arcs in the contour tree. Consequently, to have a progressive understanding of the scalar field, topological simplification hierarchies are computed.

B. Simplification hierarchy

Persistent homology [**?**] provides a sound theoretical framework for noise removal, progressive simplification and multi-scale topology abstractions. In practice, very simple algorithms have been used to compute multi-scale representations of the contour tree. Given an input *scale threshold* s, persistence based simplification consists of iteratively removing, by increasing order of function span, the arcs containing a leaf and whose function span (*persistence*) is lower than s. Consequently, simplifying the contour tree at several scale thresholds defines a progressive hierarchy of contour trees, as illustrated in Figure 4, where the small details are progressively removed and the major features are progressively highlighted. Simplification hierarchies then provide to the user a progressive understanding of the field, allowing him/her to zoom-in or zoom-out in the details of the topology abstractions.

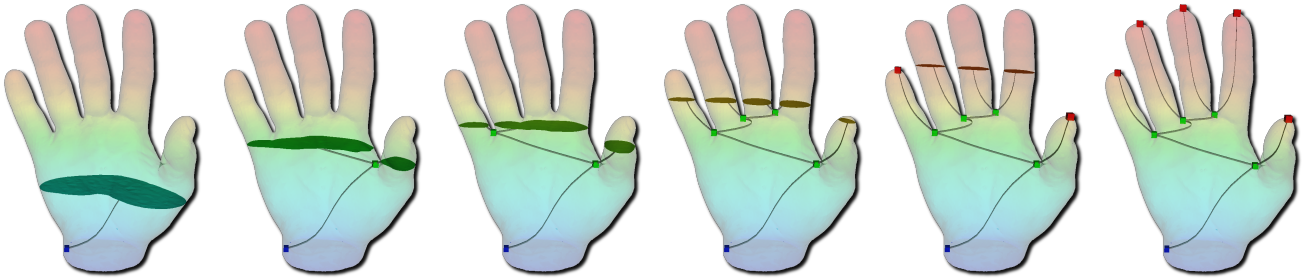


Figure 3. Height function on a manifold volume. From left to right: the contours of f are continuously contracted to nodes in the contour tree as w increases. Right: final contour tree.

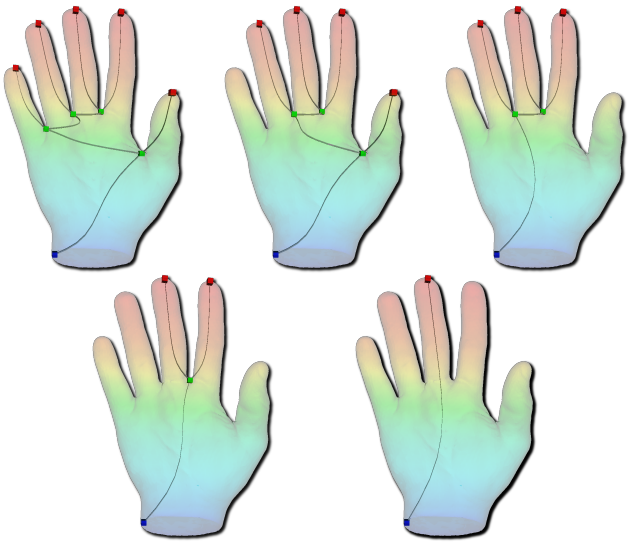


Figure 4. Progressive persistence based simplification of a contour tree (from top left to bottom right): the arcs corresponding to the fingers of the hand are progressively removed according to their *size* (persistence).

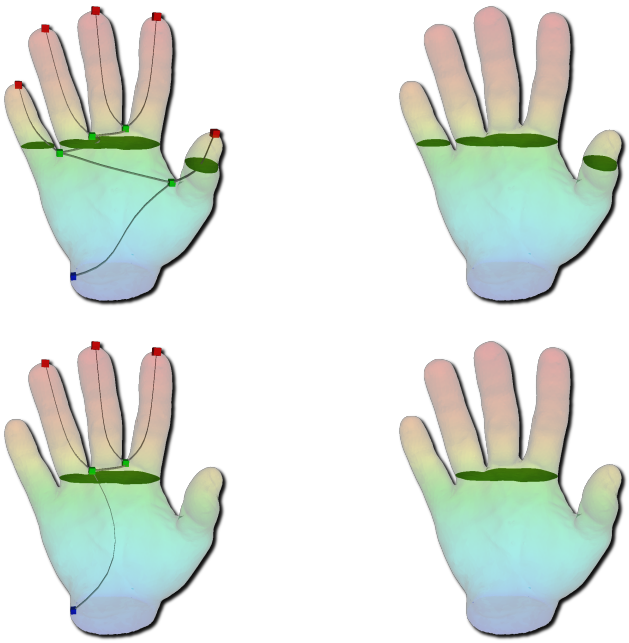


Figure 5. Flexible isosurface extraction. Top: Seed sets are extracted from the contour tree and isosurfaces are computed by standard traversal techniques. Bottom: Isosurface extraction on a simplified contour tree leads to the removal of the small components of the isosurface.

Notice that sophisticated geometry-aware simplification criteria have been proposed [**?**], in order to provide the user full flexibility for noise definition (integrating, instead of the persistence criterion, the actual size of the isosurfaces for example).

C. Flexible isosurface queries

Since they capture in a concise manner the full structure of the input scalar field, contour trees can be employed as an efficient indexing key to access isosurfaces. Given an isovalue w, an isosurface extraction consists in: *(i)* identify all the arcs crossing w, *(ii)* for those arcs, extract the *seed vertex* with the function value closest to w and finally *(iii)* start standard isosurface traversal at the extracted seed vertices. In particular, when queried from simplified contour trees, isosurfaces benefit from the simplification scheme, as illustrated in Figure 5, where the components corresponding to small arcs are not extracted.

As a result, the user also benefits from a progressive understanding of the isosurfaces, where noise can be progressively removed and where the most important features are progressively highlighted. Finally, the user can also directly interact with the contour tree by explicitly highlighting arcs from where to extract an isosurface component, providing him/her with fully interactive exploration of the data.

IV. Contour Tree Widget

We developed a web interface called *Contour Tree Widget*, whose purpose is to provide scientists with better ways of exploring simulation results. In particular, it uses contour trees to facilitate the interaction with isosurfaces of these simulation results. The interface design of this widget is illustrated in Figure 6. It consists of three main interactive components: the *3D View* on the left; the *Contour Tree View* on the top right and the *Complexity Slider* on the bottom right.

The Contour Tree View displays the contour tree of the current dataset. Usually the displayed tree is a simplified version of the complete contour tree where only the "largest" features are represented. The contour tree is laid out satisfying the following constraints: the y coordinate

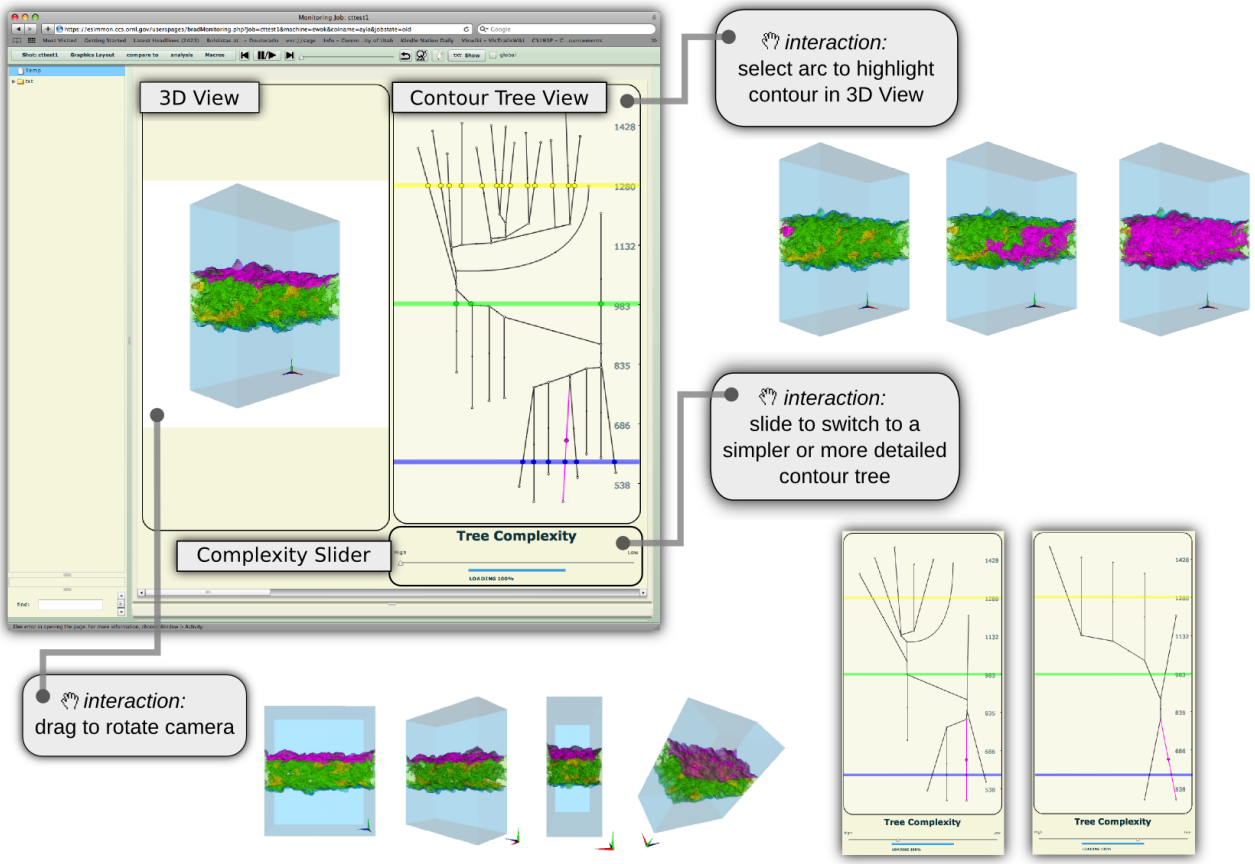


Figure 6. Contour Tree Widget's interactive components: 3D View, Contour Tree View and Complexity Slider. Some examples of the possible interactions are also shown.

of every node corresponds to its value in the scalar field and any horizontal line crosses every arc at most once. The three horizontal lines colored in blue, green and yellow in the Contour Tree View correspond to default isovalues of the three isosurfaces displayed in the 3D View in semi-transparent blue, green and yellow respectively. Note that each of these isosurfaces might contain multiple components. For example, in Figure 6, there are six components in the blue isosurface, three components in the green isosurface and thirteen components in the yellow isosurface because these are exactly the number of crossings of the blue, green, and yellow horizontal lines with the contour tree in the Contour Tree View.

As illustrated in the interaction boxes of Figure 6, by dragging the mouse on the 3D View, the user is able to rotate the camera used by the 3D View and see the isosurfaces from different angles. Another possible interaction is to select arcs from the contour tree in the Contour Tree View. At most one of these arcs can be selected at a time, causing it to be highlighted in magenta and the contour corresponding to the mean isovalue of that arc to be displayed in semi-transparent magenta in the 3D-View.

The Complexity Slider is used to switch the current contour tree in the Contour Tree View to a simplified or to a more detailed version. It has an important role of giving the user a means to understand the dataset in a progression. By starting with a simpler contour tree, only the "largest" features from the dataset will be shown. This results in simpler 3D visualizations with fewer contours and less occlusion, which might lead to a better overall understanding of the dataset. By gradually raising the complexity of the contour tree using the Complexity Slider the user can understand where the "smaller" features appear and how do they relate to the "larger" features, as the 3D visualization become more complex. The meaning of a "larger" or "smaller" feature in our current implementation is related to the notion of persistence explained in Section III, and, as it was also mentioned there, other ways to characterize the "size" of the features (*e.g.,* its volume) can be used in the simplification of the contour tree.

The Contour Tree Widget is designed to work with precomputed information. This design choice fits well the fact that it runs on the web and into the general model where we see its application: *large simulations that are both time and space consuming where there is a relative small time and space overhead in computing useful extra information during the simulation that can lead to an early understanding of (partial) results*. The required "extra

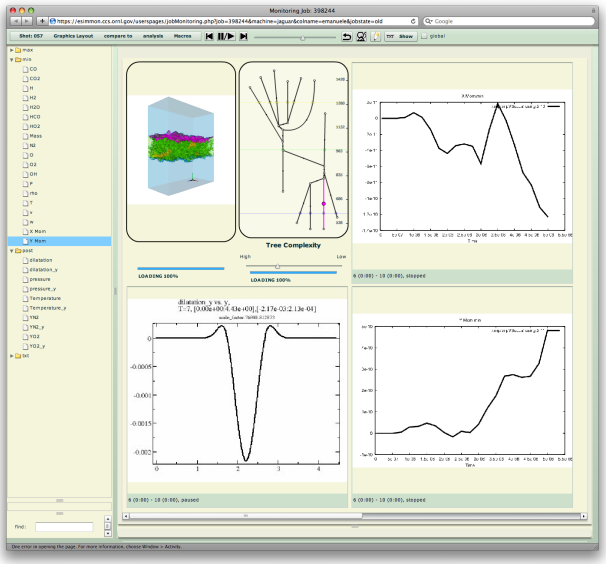


Figure 7. The Contour Tree Web Widget integrated with the eSimMon dashboard.

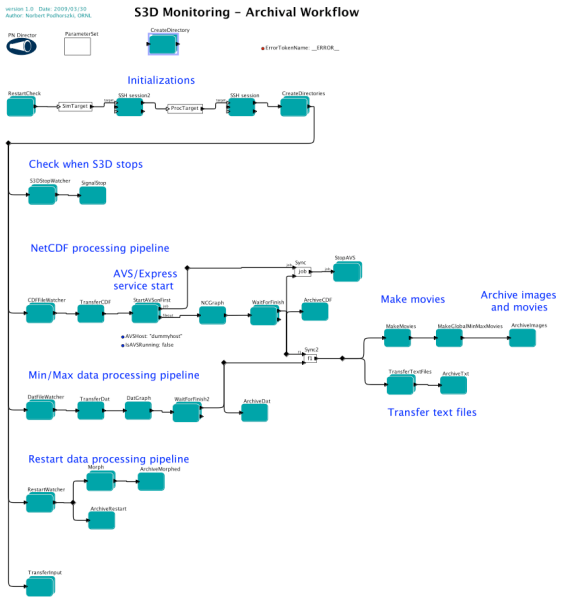


Figure 8. S3D Kepler Workflow.

information" necessary to be able to use the Contour Tree Widget is obtained by computing the contour tree of a dataset, selecting a set of simplified versions of it and composing a set of associated isosurfaces visualizations.

Figure 7 shows the Contour Tree Widget integrated with the web interface of the eSimMon dashboard. Note the presence of other widgets exposing different aspects of the dataset being used in combination with the Contour Tree Widget.

V. Case Study: Visualizing Turbulent Combustion

Combustion scientists in collaboration with computer scientists are interested in understanding the underlying processes in internal combustion engines. Compared to current engines, next-generation combustion engines will function in non-conventional, mixed-mode, turbulent combustion regimes and are likely to be characterized by higher pressures, lower temperatures, higher levels of dilution, and utilization of alternative fuels that exhibit a wide range of chemical and physical properties. Combustion processes in these environments result in complicated interactions that are poorly understood at a fundamental level, and are demanding for high-fidelity direct numerical simulation (DNS) approaches that capture these turbulence-chemistry interactions. These simulations are costly, requiring several million CPU hours on a terascale computer, up to several billion grid points, and generating tens of terabytes of data [**?**].

To solve the complex equations governing these simulations, the scientists use S3D [**?**], a massively parallel DNS solver based on a high-order, non-dissipative numerical scheme that was developed at Sandia National Laboratories. The tens of terabytes of raw data produced by S3D need to be analyzed and visualized to obtain physical insight and/or to validate models.

Among the generated data are NetCDF/HDF5 analysis files, containing both 2D slices and 1D slices from the 3D dataset. Each variable in the NetCDF file is either processed using a plot library (for xy plots only) or AVS/Express [**?**] (for generating the images of colormaps and contours). The scientists use the Kepler workflow system [**?**] to manage the S3D workflow (shown in Figure 8), including movement and provenance of the generated data, and the eSimMon dashboard system described in Section II-A to monitor and visualize the results during the execution of S3D. The types of visualization currently available are displayed in Figure 2. Although the dashboard provides useful tools for interacting with plots, 3D visualization is not explored. The scientists are limited to visualizing 2D slices of a 3D volume over time. As the datasets are very large, the scientists are not able to visualize all the space of isosurfaces and they have to carefully select which isosurfaces they want to be computed in high resolution. Our strategy is to use a low resolution version of the dataset and apply the concept of contour trees (see Section III) to not only display an overview of the dataset but also to guide them in finding interesting features in the visualization.

To achieve that, we extended the S3D workflow by adding a process to compute contour trees, simplify contour trees, extract isosurfaces, and generate multiple images of 3D visualizations necessary to feed the Contour Tree Widget explained in Section IV. This new process on the S3D workflow is launched in batch mode after the NetCDF processing pipeline displayed in Figure 8 is done.

A. Performance Analysis

The dataset used in this case study consisted of a regular grid of dimensions 45 by 112 by 96. In this case, the code written in C we used to compute the contour tree took 4.99s to finish in a desktop computer with an Intel

Core2 2.83 Ghz architecture and 8 Gb RAM. It is worth mentioning that this C code was optimized for tetrahedral meshes and not for the regular grids we used as input, so there is still margin for improvements. As mentioned in Section III the contour tree algorithm runs in $O(n \log(n))$ where n is the number of vertices of the mesh. In practice this means that it is feasible to extract contour trees for much larger datasets than the one from this case study, depending on the desired application. The running time to simplify the full contour tree into simpler versions is usually just a small fraction of the time to compute the full contour tree. In our dataset it took 0.06s to simplify the full contour tree into the simpler versions we used, this is less than 2% of the time it took to compute the complete contour tree.

As explained in Section III, we can use the contour tree to speed up the isosurface extraction. We used this approach in the dataset of this case study. The time to find the seeds for the isosurface extraction was always less than 0.00001s and the time to propagate the seeds to actually extract the largest isosurface in our dataset took 0.38s and resulted in an isosurface with 277098 triangles.

The total time to compute all the extra information needed to run the Contour Tree Widget in this case study took less two minutes. In the end, the dataset was ready to be explored on the Web through four simplified versions of the contour tree indexing visualizations for almost one hundred potentially relevant isosurfaces and contours available in 26 different angles each. The space required for this extra data was 140Mb with images in a 1024 by 1024 resolution.

VI. Conclusions

The use of remote high-performance computing facilities is becoming ubiquitous. Using these resources, scientists are able to generate unprecedented volumes of data, most of which simply can not be moved back to their sites. Advanced remote data analysis and visualization tools are thus essential in this scenario. In this work, we focus on describing a tool for exploring and identifying useful regions of large datasets. We used a summarization structure called *contour tree*, that captures the topological structure of a scalar field and helps the user identify useful (important) isosurfaces. We designed a specific user interface that allows users to interact with and explore multiple isosurfaces. In order to optimize interactivity, we used caching of the results (in particular images and movies) wherever possible. We implemented our work in the eSimMon dashboard system, and applied it in the context of simulation of combustion processes.

There is substantial future work to be pursued. First of all, we need to further improve the usability of the contour tree widget by doing expert studies. Also, we need to explore alternative ways to optimize the computation and rendering of the isosurfaces. Right now, everything is cached and transferred as precomputed images or movies, but as browsers support more flexible 3-D rendering functionality, we might consider other rendering techniques. We would also like to explore the use of state-of-the-art level of detail and streaming techniques.

Acknowledgments

We would like to thank Jacqueline Chen at the Combustion Research Facility, Sandia National Laboratory for providing the dataset used in the case study. Our research has been funded by the Department of Energy SciDAC (VACET and SDM centers), the National Science Foundation (grants IIS-0905385, IIS-0746500, ATM-0835821, IIS-0844546, CNS-0751152, IIS-0713637, OCE-0424602, IIS-0534628, CNS-0514485, IIS-0513692, CNS-0524096, CCF-0401498, OISE-0405402, CCF-0528201, CNS-0551724), and IBM Faculty Awards. E. Santos and J. Tierny are partially supported by Fulbright scholarships.

References

[1] Hamish Carr, Jack Snoeying, and Ulrike Axen. Computing contour trees in all dimensions. In *ACM Symposium on Discrete Algorithms*, pages 918–926, 2000.

[2] Hamish Carr, Jack Snoeyink, and Michiel van de Panne. Simplifying Flexible Isosurfaces Using Local Geometric Measures. In *IEEE Visualization '04*, pages 497–504, 2004.

[3] J H Chen, A Choudhary, B de Supinski, M DeVries, E R Hawkes, S Klasky, W K Liao, K L Ma, J Mellor-Crummey, N Podhorszki, R Sankaran, S Shende, and C S Yoo. Terascale direct numerical simulations of turbulent combustion using S3D. *Computational Science & Discovery*, 2(1):015001 (31pp), 2009.

[4] Hebert Edelsbrunner, D Letscher, and A Zomorodian. Topological persistence and simplification. *Discrete Computational Geometry*, 28:511–533, 2002.

[5] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T. Silva. Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3):11–21, May-June 2008.

[6] E.R. Hawkes, R. Sankaran, J.C. Sutherland, and J.H. Chen. Direct numerical simulation of turbulent combustion: fundamental insights towards predictive models. In *Journal of Physics: Conference Series*, volume 16, pages 65–79, 2005.

[7] S. Klasky, M. Vouk, M. Parashar, A. Khan, N. Podhorszki, R. Barreto, D. Silver, and S.G. Parker. Collaborative visualization spaces for petascale simulations. In *Proceedings of 2008 International Symposium on Collaborative Technologies and Systems (CTS 2008)*, 2008.

[8] Jay Lofstead, Fang Zheng, Scott Klasky, and Karsten Schwan. Adaptable, metadata rich io methods for portable high performance io. In *In Proceedings of IPDPS'09*, 2009.

[9] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH 1987*, 21(4):163–169, 1987.

[10] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience*, 2005.

[11] B. Ludascher, I. Altintas, S. Bowers, J. Cummings, T. Critchlow, E. Deelman, D. D. Roure, J. Freire, C. Goble, M. Jones, S. Klasky, T. McPhillips, N. Podhorszki, C. Silva, I. Taylor, and M. Vouk. Scientific process automation and workflow management. In A. Shoshani and D. Rotem, editors, *Scientific Data Management: Challenges, Existing Technology, and Deployment*, Computational Science Series, chapter 13. Chapman & Hall/CRC, 2009.

[12] John Milnor. *Morse Theory*. Princeton University Press, 1963.

[13] Pierre Mouallem, Mladen Vouk, Scott Klasky, Norbert Podhorszki, and Roselyne Barreto. Tracking Files Using the Kepler Provenance Framework. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management (SSDBM'09)*, LNCS 5566, pages 273–282, 2009.

[14] Tim Oreilly. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *O'Reily Media. Communications & Strategies*, (1):17, First Quarter 2007.

[15] Georges Reeb. Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *Comptes-rendus de l'Académie des Sciences*, 222:847–849, 1946.

[16] Craig Upson, Jr. Thomas Faulhaber, David Kamins, David H. Laidlaw, David Schlegel, Jefrey Vroom, Robert Gurwitz, and Andries van Dam. The Application Visualization System: A Computational Environment for Scientific Visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, 1989.

[17] M van Kreveld, R van Oostrum, L Bajaj, C, V Pascucci, and R Shcikore, D. Contour trees and small seed sets for isosurface traversal. In *ACM Symposium on Computational Geometry*, pages 212–220, 1997.

[18] G H Weber, Scott E Dillard, H Carr, Valerio Pascucci, and Bernd Hamann. Topology-controlled volume rendering. *IEEE Visualization*, 13:330–341, 2007.