

Data-Intensive Management and Analysis for Scientific Simulations

Randy Hudson¹

John Norris¹

Lynn B. Reid^{2,3}

G. Cal Jordan IV¹

Klaus Weide¹

Michael E. Papka^{1,4}

¹ Flash Center, University of Chicago, Chicago, IL 60622, USA,

² Western Australian Geothermal Centre of Excellence, CSIRO, Kensington, WA 6151,

³ School of Environmental Systems Engineering, University of Western Australia, Crawley, WA,

⁴ Computation Institute, Argonne National Laboratory / University of Chicago, Chicago, IL 60622, USA
Email: papka@anl.gov

Abstract

Scientific simulations can produce enormous amounts of data, making the analysis of results and management of files a difficult task for scientists. The simulation management and analysis system (*Smaash*) described here is designed to allow scientists to easily capture, store, organize, monitor, and analyze simulation results. The system is automatic, standardized, and secure. *Smaash* was built using open-source tools and modularized to be independent of the scientific simulation. The web-based front-end allows the scientist to easily interact with the data, and has proved its usefulness in improving the efficiency of a scientific team's workflow.

Keywords: Data-intensive, scientific workflow management, FLASH astrophysical code

1 Introduction

High performance parallel computing allows scientists to solve complex physical problems through computer simulation. However, the massive amounts of data generated and the complex computing environment can create additional complications. A recent review by Ludäscher et al.(2009) describes how scientific workflows can assist scientists in extracting knowledge from these data-intensive operations by automating components within pipelines. Within the fusion community, Klasky et al.(2008) and colleagues have developed a system that handles the storage management, data movement, metadata generation and management, and a means to analyze the results. In response to scientists' needs, a simulation management and analysis system (*Smaash*) was developed at the University of Chicago and Argonne National Laboratory (USA). *Smaash* provides an integrated way to monitor simulations and analyze computational results; catalog, store, and retrieve simulations; and prepare output for publications. The system is independent of the particular simulation code, accessible from many HPC and browser-based platforms, and built around open-source software tools. Data security and provenance is considered throughout. The

This work was part supported at the University of Chicago by the US Department of Energy under contract B523820 to the ASC Alliances Center for Astrophysical Nuclear Flashes.

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the Ninth Australasian Symposium on Parallel and Distributed Computing (AusPDC 2011), Perth, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 118, J. Chen and R. Ranjan, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

analysis components are hidden behind a web-based front end, enabling scientists to focus on their results and not get bogged down by information overload.

2 Typical Data Requirements – FLASH code

The FLASH multiphysics adaptive mesh refinement code developed at the University of Chicago (Dubey et al., 2009) provided prototype data, and the astrophysicists of the Flash Center provided essential feedback to *Smaash* developers about analysis needs and scientific workflows. Typical scientific applications of the code include weakly-compressible turbulent flows (Fisher et al., 2008) and detonation of Type 1A supernovae (Jordan IV et al., 2008). The initial application is a set of parameter studies of Type 1A supernovae explosions which were calculated on the unclassified Purple HPC system at LLNL and the IBM BlueGene/P HPC system at Argonne National Lab.

A typical simulation of this three-dimensional physical system requires eight linked runs, or restarts, of 12 wall-clock hours each, which progressively cover only a few seconds of simulation time. Calculated on thousands of cores, the output might have: 90 large result files at 34GB each; 600 smaller analysis files at 9 GB each; log files recording integrated physical results and computational progress; and affiliated processing and visualization results. Total storage for one run may require eight terabytes of disk space; the parameter study required 16 of these complete simulations. The management of the data produced in this scientific study could be overwhelming for researchers who are primarily interested in abstracting physical insights from the computational results. Moreover, because of the limited availability of these HPC systems, the computed data must be carefully preserved and provenance understood.

3 Smaash Components

Smaash consists of three main parts: a back-end to capture, store, verify, and monitor simulation; a front-end designed for the scientists' needs; and a database. The front- and back-ends have modular components, allowing easy extensibility. The database is designed to be independent of the simulation code output formats and research genre.

The first step in the *Smaash* back-end is to automatically capture the data being generated and store it securely in the database. The archiving pipeline starts with a *Collector* which tracks the simulation progress through the simulation code's log file, and launches dynamically-loaded tools to record information into the database about each individual output type being generated. Next, the *Archiver* automates the transfer of files on the local filesystem into long-term mass storage, additionally storing details about

the provenance in the database. The *Verifier* maintains the accuracy of the database, and the *Associator* keeps related analyses, such as post-processing results, connected to the original simulation results. The *Observer* monitors these back-end processes, and emails the user of significant changes. This program frees scientists from tending a lengthy simulation by being tied to a terminal, and allows efficient use of allocated computer processing time.

The back-end tools generally run on the same HPC system as the concurrent simulation, but on separate processors to avoid degrading simulation performance. They are robust and can recover from fatal conditions, and communicate securely.

3.1 User Interfaces

The control interface for Smaash is web-based and allows computational scientists to manage most post-processing and analysis from a web browser distant from the HPC system. Two primary front-end components are the *TreeView*, which hierarchically organises simulation sets, and the *GraphView* which shows visual details of simulation progress and provides easy access to data output. Figure 1 shows the TreeView in action, where multiple users can keep track of cascading simulations and their restarts. Figure 2 displays a user-definable concise window into the enormous quantity of data created by two simulations.

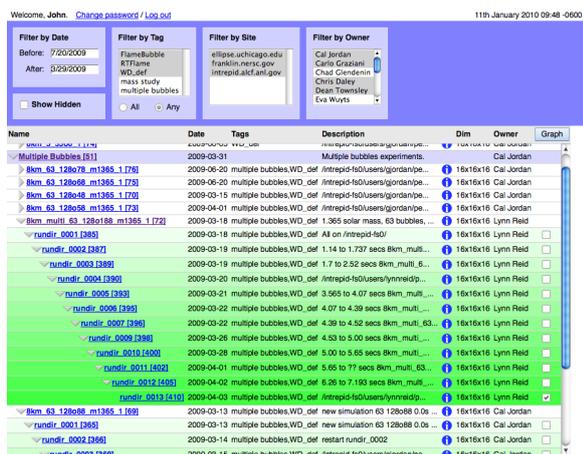


Figure 1: Web interface to the TreeView, showing multiple restarts in a single simulation.

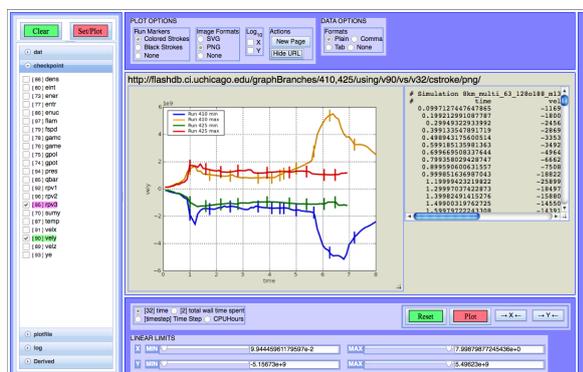


Figure 2: Web interface to the GraphView of two simulations, showing integral physical parameters plotted against simulation time. Curve coordinates are displayed on the right.

Other interfaces help the user keep track of simulation progress and do quick data analysis, such as the *Visualizer* pipeline, which provides graphical

snapshots of physical states over time. Summary web pages detail the accumulated information in the database and allow user annotations, while the robust URL feature of the graph pages allows a science group to share up-to-date notes through a wiki page.

3.2 Implementation

Smaash is designed to be easily adapted to a new scientific simulation framework by using modularity and standardization. Soft constraints encourage users to enter meaningful descriptions, and maintain data provenance. Off-the-shelf open-source tools such as MySQL, Django/Dojo/Dojango, and Matplotlib allow rapid development, extensibility, and provide well-considered security protocols. The Smaash development team is actively looking for new collaborations which would benefit from the integrated management and analysis tools described here.

4 Smaash in Action

The Smaash data management pipeline has greatly improved the efficiency of the scientific team's workflow. In one example, the front-end GraphView allowed easy amalgamation of multiple simulations into a clear picture showing differences in supernovae detonation times. Cross-referencing to the TreeView provided the means to pinpoint and extract crucial files for further analysis. In another computer science example, the FLASH programming team was able to spot a glaring inefficiency in CPU usage by plotting elapsed output file write times in the GraphView. Implementing a quick programming fix improved the use of precious allocated CPU time by forty percent. Smaash allows the scientist to shift focus from monitoring the simulation to analysing the results, while maintaining data integrity.

5 References

Dubey, A, Antypas, K, Ganapathy, MK, Reid, LB, Riley, K, Sheeler, D, Siegel, A & Weide, K (2009), 'Extensible component based architecture for FLASH, a massively parallel, multiphysics simulation code', *Parallel Computing* **35**, 512–522.

Fisher, RT, Kadanoff, LP, Lamb, DQ, Dubey, A, & 15 others 'Terascale turbulence computation using the FLASH3 application framework on the IBM Blue Gene/L system', *IBM Journal of Research and Development* **52** (1/2) 127–137, special issue on "Applications of Massively Parallel Systems".

Jordan IV, G, Fisher, R, Townsley, D, Calder, A, Graziani, C, Asida, S, Lamb, D & Truran, J. (2008), 'Three-dimensional simulations of the deflagration phase of the gravitationally confined detonation model of Type Ia supernovae', *The Astrophysical Journal* **681**, 1448–1457.

Klasky S, Barreto, R, Kahn, A, Parashar, M, Parker, S, Silver, D & Vouk, M (2008), 'Collaborative visualization spaces for petascale simulations', *International Symposium on Collaborative Technologies and Systems* 203-211.

Ludäscher, B, Altintas, I, Bowers, S, Cummings, J, Critchlow, T, Roure, EDDD, Freire, J, Goble, C, Jones, M, Klasky, S, McPhillips, T, Podhorszki, N, Silva, C, Taylor, I & Vouk, M (2009), Scientific process automation and workflow management, in A Shoshani & D Rotem, eds, 'Scientific Data Management: Challenges, Technology, and Deployment', Computational Science Series, Chapman and Hall/CRC, chapter 13.