A Visual Representation for Data and Workflow of HPC Applications

Johnatan Garcia[1], Chen Wang [2]
1. National Nuclear Security Administration
2. Lawrence Livermore National Laboratory
garcia323@llnl.gov, wang116@llnl.gov

Introduction

Real-world HPC applications often necessitate the collaboration of multiple nodes to exchange data to achieve specific objectives. These objectives may manifest through various means, such as machine learning (ML), high-scale scientific simulations, or workflows composed of multiple applications in conjunction to complete a common goal. As the complexity of the application escalates, new challenges and conflicts emerge in tracking the flow of data between nodes. While efforts have been made in the visualization of single HPC applications, the visualization of data flow across the entirety of the workflow remains an area with limited progress.

Methodology

In HPC systems, it is common to observe a hierarchical I/O stack acting as a bridge from high-level operations to low-level hardware interactions. Some abstraction is presented at each level, treating the stack as a black box. Our main contribution is the data flow visualization of a real-world HPC workflow, Montage [1]. Utilizing Recorder [2], we collected trace files of Montage running a Lustre File System. Recorder enables the tracking of multi-level I/O calls processed throughout the entire stack, facilitating the analysis of I/O behavior at each level. By leveraging Recorder we generated a multi-level trace that includes MPI and POSIX calls throughout the Montage workflow. Utilizing the recorder-viz tool, a Python library for visualization, we can create one visualization report (an HTML file) for each component in the workflow. Specifically, each report contains four main sections: Overall I/O performance, function statistics, access patterns, and I/O statistics.

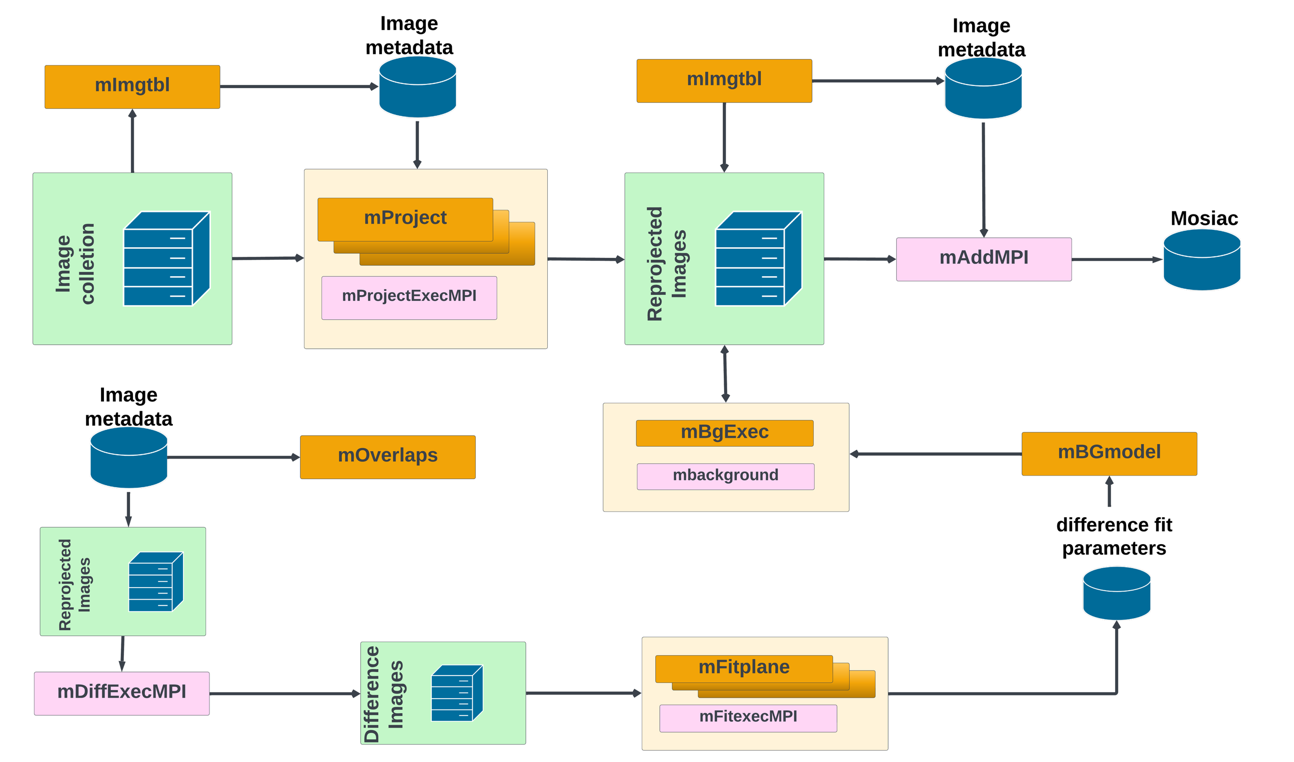

Figure 1: Montage is a software toolkit for constructing mosaics. A Montage workflow is composed of multiple MPI components (pink) and non-MPI components (orange). This figure is modified from Figure 1 of [3].

Results

Montage is a toolkit for assembling FITS (flexible image transport systems) images into custom mosaics. Figure 1 depicts the employed workflow. In this process, multiple components (e.g., mprojExec, mAddMPI, etc.) are executed in a sequential manner to compose the final image. The following figures represent the I/O patterns captured using Recorder, running on one node with 24 processes. We focus specifically on the following components: mProjExecMPI, mAddMPI, mDiffExecMPI, and mFitExecMPI.

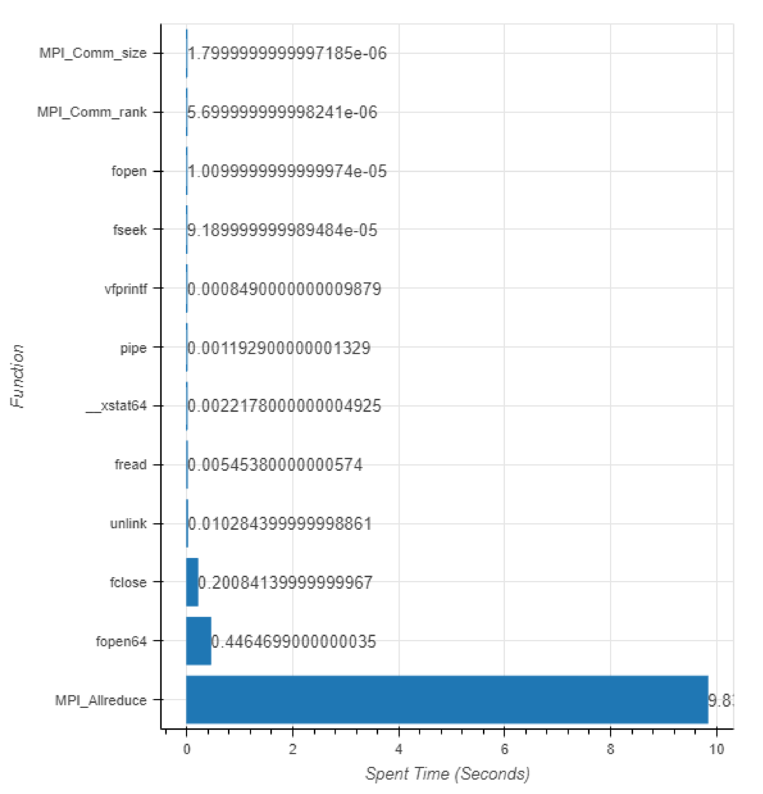

Figure 2: Total time in seconds spent utilizing each function

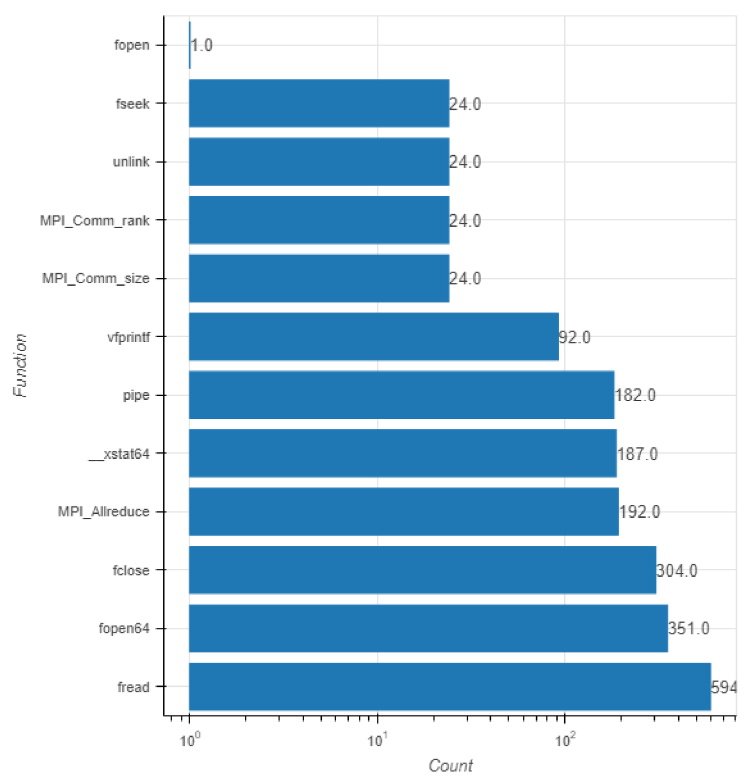

Figure 3: Functions utilized by the mProjExecMPI component along with the count. I/O is performed using POSIX calls.

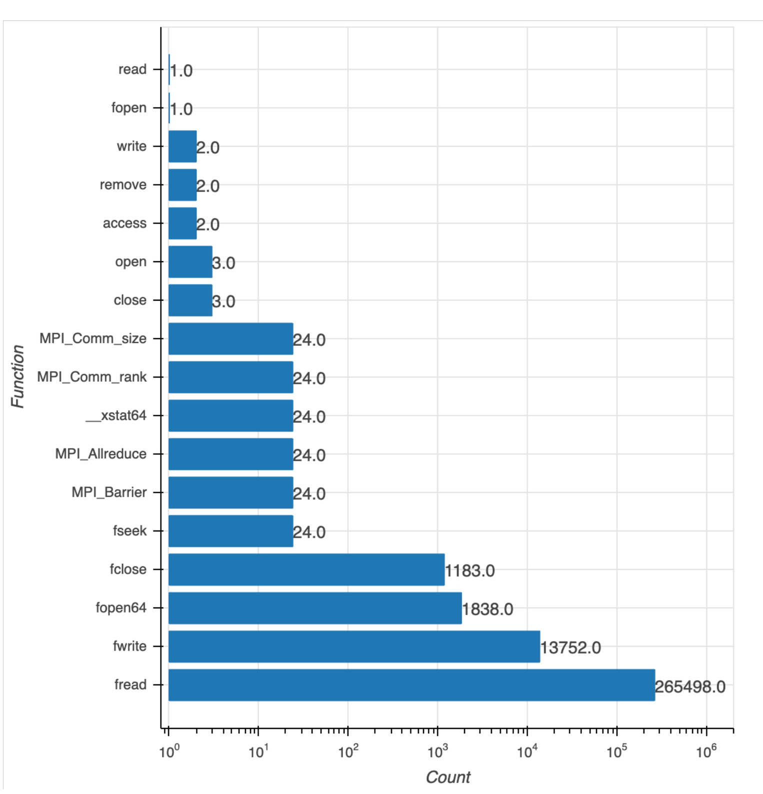

Figure 4: Functions utilized by the mAddMPI Component

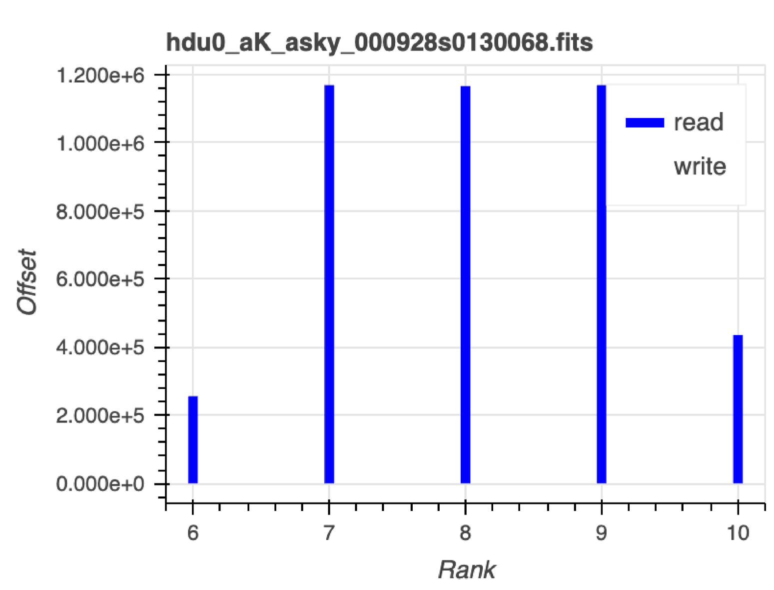

Figure 5: mAddMPI reading the fits files that have been re-projected

The first component we will look at is mProjExec, which runs mProject on the images inside the metadata. mAddMPI reads a set of flux images re-projected by the mProject component onto the same pixel space. Figure 2 and Figure 3 demonstrate that even though `fread` was the most called it was not the most expensive function called. Figure 4. mAddMPI utilized 17 types of function calls, with the majority being POSIX and only 4 being MPI. Among these, `fread` emerges as the most frequently called function, executed 265,498 times. We have scrutinized certain I/O behaviors that may contribute to potential bottlenecks. However, a conclusive investigation is pending.

Figure 6 and Figure 7 illustrate overlapping write and read operations carried out by ranks, including the associated file offsets. Observing this pattern may pave the way for future work in optimizing this I/O behavior to enhance the application's performance and mitigate potential bottlenecks. Following mOverlaps we generated traces on the mDiffExec component which runs mDiff on the pairs gathered by mOverlap. Since we have gathered pairs from mOverlap, mDiffExec only reads from one file that being the Ktemplate header file. The mFitExecMPI component is executed after and aims to write to multiple fit files which will be used by the mBgModel to apply corrections to each image to find the best fit.

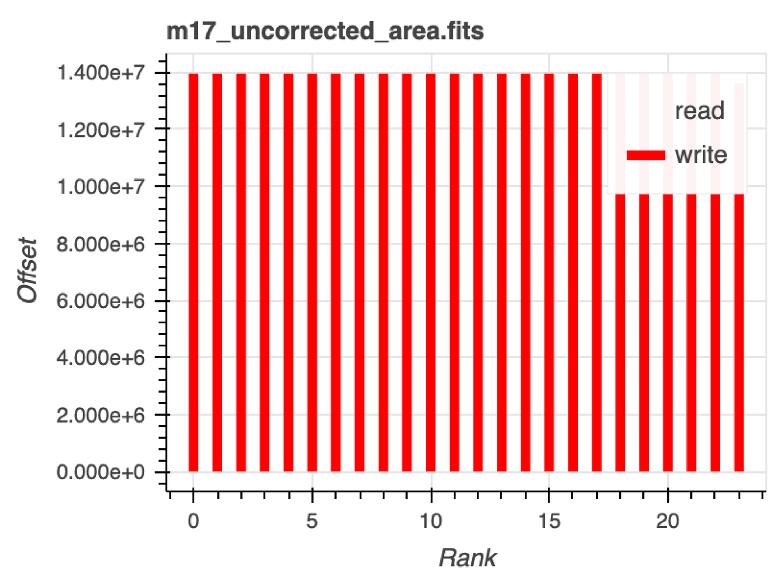

Figure 6: Overlapping writing on the FITS file done by each rank on the mAddMPI component

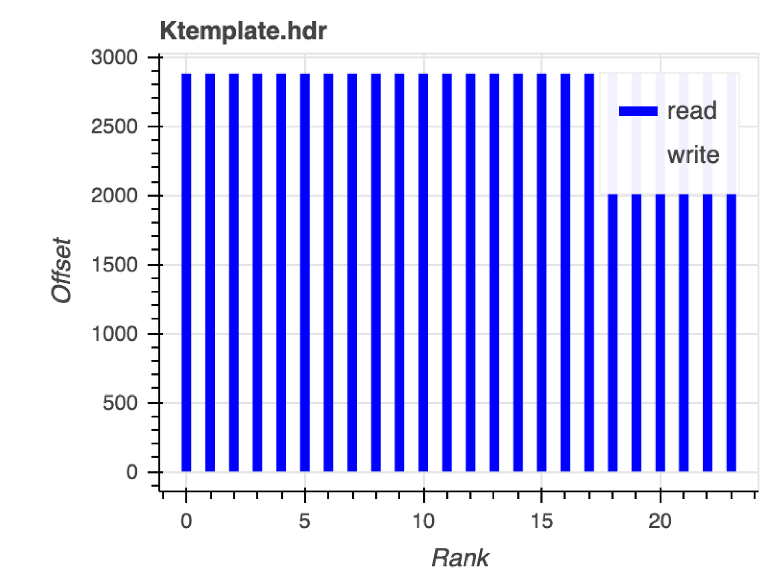

Figure 7: Overlapping reading on the header file done by each rank on mAddMPI component

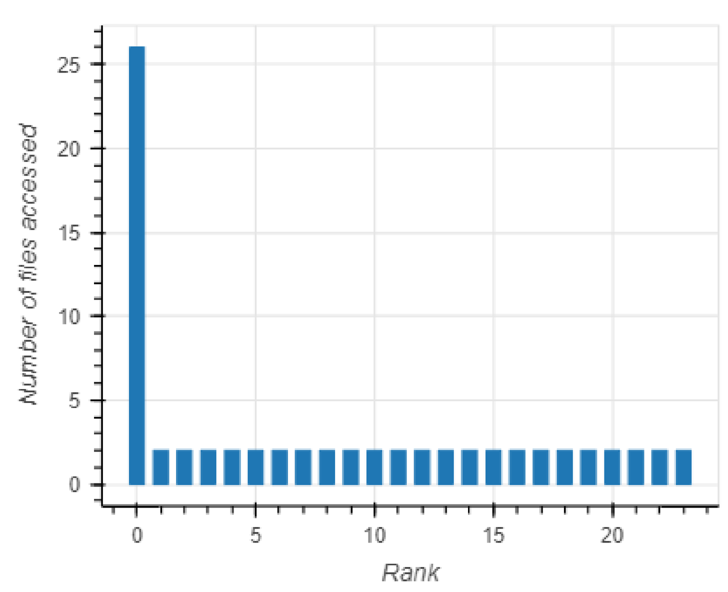

Figure 8: Number of files that each rank accessed by each rank in the mFitExecMPI Component

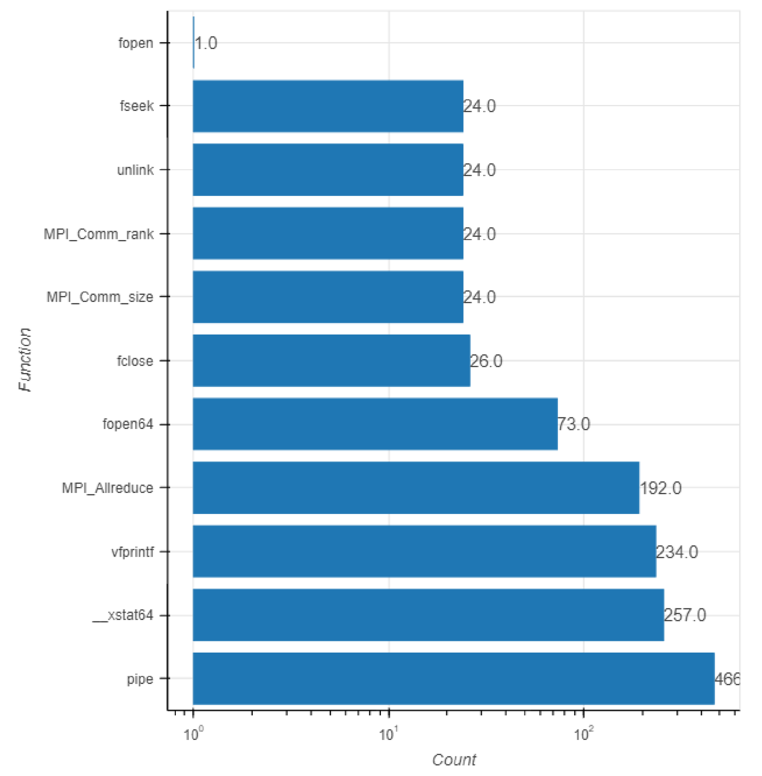

Figure 9: Functions utilized along with the number of times each call was used in the mFitExecMPI component

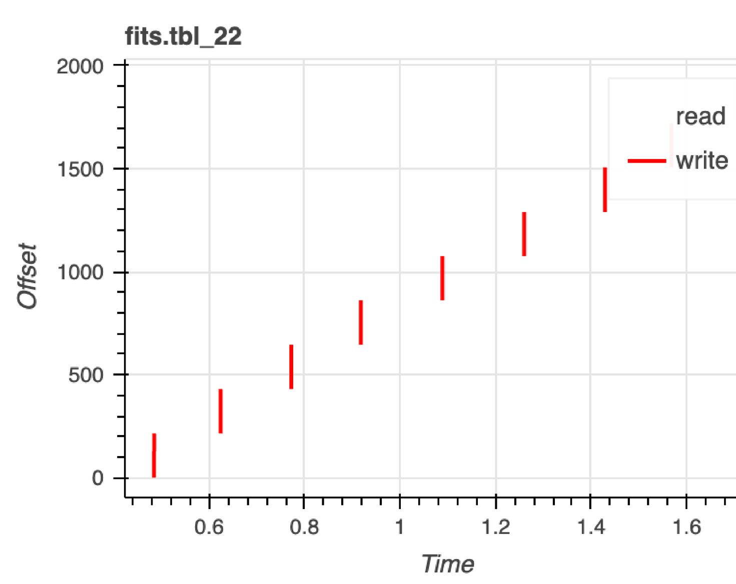

Figure 10: Writing pattern in the mFitExecMPI component for the fits.tbl_22 file with time stamp and offset

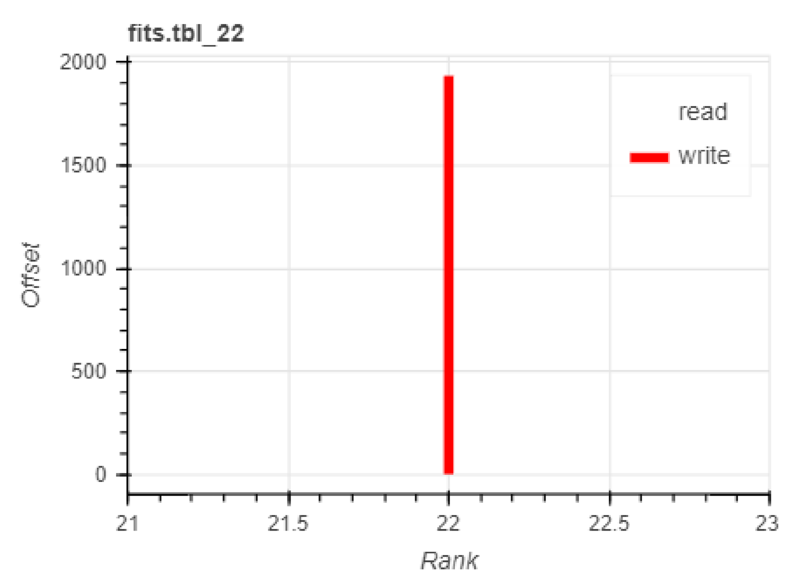

Figure 11: Rank and offset for rank 22 reading the fits.tbl_22 file

Conclusion and Future Work

In this work, we presented an approach to visualizing the workflow of real HPC applications using I/O traces that depict the entire I/O stack of each component. By generating a trace for each component, we can subsequently create a graph illustrating the data flow within our application.

With our current trace files, users can analyze the traces and create graphs that represent the flow of data throughout the application. By modifying Recorder to automate the process and create a directed acyclic graph (DAG) we can have a better representation of the data throughout the components and nodes. Further, our study only represented the visualization of the workflow, we would also want to direct our attention to using the graphs to detect potential I/O bottlenecks.

References

[1] Montage, http://montage.ipac.caltech.edu, 2024.
[2] Wang, Chen, Jinghan Sun, Marc Snir, Kathryn Mohror, and Elsa Gonsiorowski. "Recorder 2.0: Efficient parallel I/O tracing and analysis." In 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 1-8. IEEE, 2020.
[3] Jacob, Joseph C., Daniel S. Katz, G. Bruce Berriman, John C. Good, Anastasia Laity, Ewa Deelman, Carl Kesselman et al. "Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking." International Journal of Computational Science and Engineering 4, no. 2 (2009): 73-87.

ACKNOWLEDGEMENTS

his work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-POST-859767. This research was performed under an appointment to the Minority Serving Institutions Internship Program (MSIIP) administered by the Oak Ridge Institute for Science and Education (ORISE) for the National Nuclear Security Administration (NNSA) and U.S. Department of Energy (DOE). ORISE is managed by Oak Ridge Associated Universities (ORAU). All opinions expressed in this paper are the author's and do not necessarily reflect the policies and views of NNSA, DOE, ORISE or ORAU.