In Situ Generated Probability Distribution Functions for Interactive Post Hoc Visualization and Analysis

Yucong (Chris) Ye[*1], Tyson Neuroth[†1], Franz Sauer[‡1], Kwan-Liu Ma[§1],
Giulio Borghesi[¶2], Aditya Konduri[‖2], Hemanth Kolla[**2], and Jacqueline Chen[††2]

[1]University of California, Davis, CA
[2]Sandia National Laboratory, Livermore, CA

Abstract

The growing power and capacity of supercomputers enable scientific simulations at extreme scale, leading to not only more accurate modeling and greater predictive ability but also massive quantities of data to analyze. New approaches to data analysis and visualization are thus needed to support interactive exploration through selective data access for gaining insights into terabytes and petabytes of data. In this paper, we present an in situ data processing method for both generating probability distribution functions (PDFs) from field data and reorganizing particle data using a single spatial organization scheme. This coupling between PDFs and particles allows for the interactive post hoc exploration of both data types simultaneously. Scientists can explore trends in large-scale data through the PDFs and subsequently extract desired particle subsets for further analysis. We evaluate the usability of our in situ method using a petascale combustion simulation and demonstrate the increases in task efficiency and accuracy that the resulting workflow provides to scientists.

Index Terms: G.3 [Mathematics of Computing]: Probability and Statistics—Distribution Functions; H.3.2 [Information Systems]: Information Storage and Retrieval—Information Storage; I.6.6 [Computing Methodologies]: Simulation and Modeling—Simulation Output Analysis; J.2 [Computer Applications]: Physical Sciences and Engineering—Physics;

1 Introduction

The ever increasing computing power enables researchers to develop extreme scale scientific simulations. Consequently, massive volumes of output are generated, reaching terabytes or even petabytes of data. However, the speed of disk storage systems are not keeping pace with the dramatically increased computing power. The ever increasing gap between data generation and data accessibility not only creates a bottleneck in simulation output, but also poses a challenge to data analysis in post processing. Scientists need to analyze their data to generate new insights but the standard methods through which this is done are becoming too expensive.

Data preprocessing is critical for more efficient subsequent data visualization and analysis. Depending on the task, different data layouts and indexing schemes can result in a huge performance difference. For example, laying out the data into a contiguous buffer is beneficial for volume rendering [27], while indexing and sorting the data based on its attributes can accelerate query based analysis [19]. Due to the large size of data, loading the entire dataset is no longer feasible. As a result, preparing the data for high level exploration by summarization as well as for detailed analysis by selective data access is crucial for post hoc data exploration. Some of the preprocessing tasks are parallel in nature and can be performed on a supercomputer.

On the other hand, a common trend to deal with such overwhelming amounts of data is to deploy in situ visualization and analysis [28, 22, 23, 3, 24]. Moving visualization and analysis in situ is a promising strategy to harvest the difference between the supreme computing power and the lack of I/O capability. Since the results generated by visualization and analysis tasks are usually orders of magnitude smaller than the raw simulation output, moving such tasks in situ effectively mitigates the I/O bottleneck. The data generated by in situ techniques should be compact, thus imposing minimal I/O requirements to the simulations. Also, most of the computing power should be reserved for the simulations, as a result in situ techniques must have a minimal computation and memory overhead.

The goal of this work is to perform data preprocessing in situ in order to support more efficient post hoc visualization and analysis. More specifically, we present an in situ data processing method to reorganize the particle data according to a spatial organization scheme. We also generate probability distribution functions (PDFs) from scalar field data using the same spatial organization scheme. Therefore, the PDFs effectively provide an overview of the particle data. Utilizing such a coupling between PDFs and particles, a post hoc interactive visualization tool was implemented to explore both data types simultaneously. Scientists can then use the tool to selectively extract the desired particle subsets without the need to access the entire particle data. As a result, post hoc data exploration and selective particle access is made more efficient by the in situ generated PDFs and reorganized particles.

Our tools come in the form of a set of in situ processing routines as well as a post hoc visualization and analysis system. The in situ methods are evaluated with multiple simulation runs on Lawrence Berkeley National Laboratory's NERSC computing facility. To test our method, we integrate our tools into S3D, a large-scale direct numerical simulation (DNS) code that models the turbulent nature of combustion processes. For the post hoc visualization tool, we demonstrate the usability by showing the improved efficiency and accuracy of particle selection through a case study.

The remainder of the paper is organized as follows: We first present background information including the traditional scientists' workflow of particle selection in Section 2. Our proposed work-

[*]e-mail: chrisyeshi@gmail.com
[†]e-mail: taneuroth@ucdavis.edu
[‡]e-mail: fasauer@ucdavis.edu
[§]e-mail: klma@ucdavis.edu
[¶]e-mail: gborghe@sandia.gov
[‖]e-mail: akondur@sandia.gov
[**]e-mail: hnkolla@sandia.gov
[††]e-mail: jhchen@sandia.gov

flow is then presented in Section 3 along with the implementation details of both the in situ processing methods and the post hoc visualization tool. In Section 4, the in situ performance results and the usability case study are presented.

2 BACKGROUND

The massive amounts of data generated by extreme scale scientific applications present daunting challenges to researchers in a wide variety of fields. In this section we present related work on in situ processing and the use of PDFs for data analysis. In addition, we provide some background information on the S3D combustion simulation, its applications, and the scientists' prior workflow before adopting our new techniques.

2.1 Related Work

In situ processing is becoming a popular technique in overcoming the I/O barrier. Multiple overview papers of in situ visualization and analysis have been presented. An earlier work by Ma et al. [15] presented the challenges and opportunities of in situ visualization due to the growing power of supercomputers. Ahern et al. [1] later outlined a research roadmap for exascale computing and discussed the pros and cons of in situ processing. Klasky et al. [11], in the same year, gave an overview of the major problems with extreme scale simulations. They then introduced a multipoint approach to handle the presented problems, including service oriented architecture and in situ processing. Childs et al. [6] categorized in situ processing into co-processing, concurrent processing, and hybrid. Major design principles involved with the solutions to the different types of in situ processing are also described in detail. More specifically, co-processing means tightly coupled synchronous, which performs visualization and analysis in the same process as simulations. Concurrent processing means loosely coupled asynchronous staging, which moves processing tasks to staging nodes. Lastly, hybrid means the combination of co-processing and concurrent processing in the same scientific application. The in situ processing in this paper falls in the category of co-processing.

Multiple tools, libraries, and frameworks have been developed for in situ processing. Visualization applications such as ParaView and VisIt provides easy to adapt libraries for the purpose of in situ visualization, namely ParaView Catalyst [3] and VisIt Libsim [24]. For more general in situ processing needs, ADIOS [14] implements service oriented architecture principle for data management to ease the data communication between simulations and in situ processing tools; Damaris [7] also provides simple to use APIs for in situ processing and I/O by using dedicated cores or nodes.

One approach to hide the I/O bottleneck with in situ processing is to perform data compression before outputting. ISOBAR [18] was introduced as an asynchronous method by exploiting data access patterns for more efficient data compression and I/O. Lakshminarasimhan et al. presented ISABELA [12], an error bounded in situ compression algorithm for scientific data, which offers up to an 85% compression ratio by taking advantage of both spatial and temporal coherence of the data. In situ compression can be considered a less aggressive approach compared to other in situ visualization and analysis techniques because original data can be extracted from the compressed data.

Visualization techniques were the first to be adopted into in situ processing [28]. Applications that generate static images in situ are commonly used [3, 24]. A priori knowledge is usually required for generating meaningful visualizations with such in situ applications. To overcome these limitations, several approaches have been proposed to enable more explorability of the in situ generated visualizations [22, 9, 2, 8, 4]. Such in situ visualization techniques output additional information along with static images, therefore allowing some limited exploration into the original data.

Feature extraction and tracking is a common analysis technique to generate insights from massive scientific data. In situ feature extraction and tracking has been studied in several works. Landge et al. [13] used segmented merge trees to implement in situ feature extraction.Ye et al. [26] introduced in situ depth maps to support post hoc image based feature extraction and tracking. Woodring et al. [25] performed in situ eddy analysis by thresholding the Okubo-Weiss field for MPAS-Ocean climate simulation.

Data preprocessing, such as indexing, sorting, and statistical sampling, has been performed in situ in multiple works. Kim et al. [10] experimented with in situ bitmap indexing and pointed out the challenges and opportunities of parallel index creation and parallel query operations. Su et al. [20] later extended in situ bitmaps generation for scientific simulation data. They demonstrated the usability of bitmaps with online time step selections using multiple metrics (earth mover's distance and conditional entropy) and offline correlation mining. Zheng et al. [29] presented a concurrent processing approach to prepare data before it reaches storage. They described the importance of data preprocessing and demonstrated their technique with bitmap indexing, sorting, and 1D/2D histogram (PDF) generation. Our approach uses co-processing instead of concurrent processing, and we generate regional PDFs (instead of global histograms) to support post hoc visualization and analysis.

The use of histograms has also been studied in various areas. Novotny et al. [17] utilized histograms to generate parallel coordinates in real time for interactive exploration of large datasets. Thompson et al. [21] used hixels (1D histograms) to represent either a collection of values in a block of data or collections of values at a location in ensemble data. They demonstrated that topological analysis and uncertainty visualization can be performed with hixels. Neuroth et al. [16] generated spatially organized velocity histograms both on-the-fly and in situ for interactive visualization and exploration of the underlying data. Our technique generates multidimensional PDFs (histograms). We compare the accuracy gained from generating PDFs from scalar field data instead of particle data. Furthermore, we demonstrate the usability of our technique by showing the improved efficiency and accuracy of particle filtering and selecting through a practical case study.

2.2 The S3D Simulation and Current Workflow

While the techniques presented in this paper can apply to large variety of large-scale simulations, we primarily evaluate our system using case studies in the field of combustion. Specifically, we integrate our tools into S3D [5], a petascale combustion simulation developed by researchers at Sandia National Laboratories. While this simulation computes information both as a field and as tracer particles, only the particle data tends to be saved to disk since the field data is too large. As a result, in situ visualization tools that can represent information from the high resolution field data would be greatly beneficial to S3D. Furthermore, the manner through which particles are saved and accessed during post processing limits the amount of interactivity currently available in the scientists' workflow.

The original workflow used by the scientists is shown in Figure 1a. Particles are output from the simulation by each of the computing nodes in no particular ordering. In terms of selecting particles, scientists filter based on the scalar values associated with them, such as when the temperature is within a desired range. However, this filtering process must be done by iterating through all of the simulation particles to check if a particle matches the filtering criteria. The filtered particles are then visualized in a 3D scatter plot based on their positions. Lastly, groups of particles are isolated into different features based on spatial proximity to one another within local neighborhoods throughout the domain.

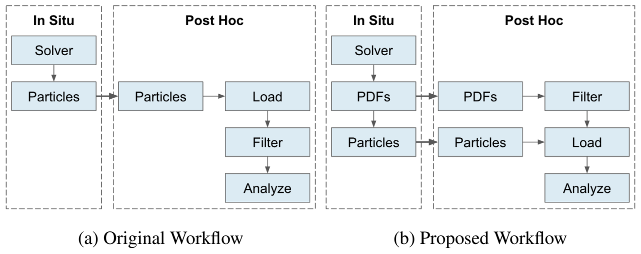


Figure 1: The scientists' workflow to select and analyze simulation outputs. In the original workflow, the entire set of particles is filtered by iterating through each particle, which is time consuming. The proposed workflow is to apply filters to in situ generated probability distribution functions (PDFs) and only load the particles which match the filtering criteria, thus allowing interactive particle selections.

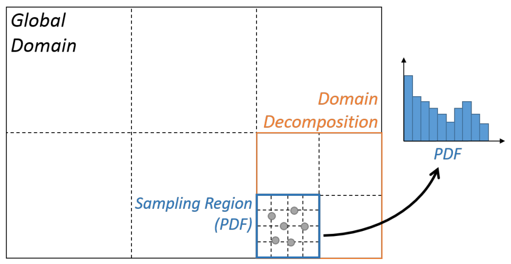


Figure 2: The hierarchical sub-division of domain regions as well as the terminology we use to indicate each throughout the paper.

3 Methodology

The proposed workflow, as shown in Figure 1b, is to utilize in situ generated probability distribution functions (PDFs) to aid post hoc particle selection and analysis. PDFs are generated in situ during the simulation and saved in a compressed format. In addition, particles are sorted according to the same spatial organization scheme of the PDFs before being written to high performance storage. In post processing, instead of filtering the particles directly (a time consuming operation), the scientists can interactively apply filters to the PDFs in an exploratory fashion. Particles are then loaded from disk according to the selected PDFs. Since both the particles and the PDFs use the same spatial organization scheme, our post hoc visualization tool is able to quickly pinpoint the memory locations of the particles on disk without scanning through the entire dataset. An important design goal is to make all interactions on PDFs to be interactive in order to verify scientists' ideas quickly. The particle loading operation based on the filtered PDFs depends on the amount of particles that need to be loaded into memory. However, since the PDF selections already subsample the domain, the amount tends to be small enough to achieve a usable level of interactivity.

3.1 In Situ Components

Both the PDF generation and particle sorting follow the same spatial organization scheme which we insert as an additional level into the simulation spatial hierarchy. At the highest level lies the simulation's *global domain*. This is then split into a number of *domain decompositions* which are each handled by a distributed computing node. We further subdivide each *domain decomposition* into a number of *sampling regions*, which each is used to generate a PDF. These *sampling regions* could be used to sample either particles or field data. Figure 2 depicts an image of these hierarchical levels.

In this study, the number of grid points per dimension per sampling domain is restricted to be an evenly divisible factor of the number of grid points in the corresponding dimension in each domain decomposition. For example, if a domain decomposition consists of 20x20x20 grid points, a possible sampling domain can consist of a 5x5x5 grid points resulting in 4x4x4 sampling regions per domain decomposition. Such a restriction ensures that a sampling domain will never span across the boundary of a domain decomposition, thus allowing our technique to remain embarrassingly parallel.

3.1.1 Probability Distribution Function Generation

PDFs can be generated from either scalar field data or particle data. However, PDFs generated from scalar fields tend to be better suited to the scientists' needs. When PDFs are generated from particle data, two sources of error can arise. One source of error comes from the total number of available particles while the other source of error comes from the distribution of the particle locations. If the number of available particles is small, then the distributions in the PDFs will not be statistically significant. On the other hand, if there are regions in the domain which contain very few or very many particles, the PDF data will be skewed. We conducted a study to analyze errors when using particle data to generate PDFs which can be found in Section 4.2.3. We therefore choose to use the scalar field data as input into the in situ PDF generation routines since it better represents trends in the data throughout the domain.

In order to easily incorporate this into the S3D simulation and maximize its usability, we provide multiple subroutines for generating PDFs with different settings along with a configuration reading subroutine to populate the PDF settings from file. Specifically, three subroutines are used to handle generating 1D, 2D, and 3D PDFs which allow users to compare the distribution of up to three variables at once. These subroutines are designed to be run in each domain decomposition (MPI process), and also handle dividing the domain decomposition into sampling regions according to the desired resolution. The inputs required for the PDF generation subroutines are: the spatial subdivision of the domain decomposition into sampling regions, the values of each scalar field location that need to be sampled into the PDF (note that we rely on the simulation to take out any ghost cells that might disrupt statistics), as well as other specific PDF parameters such as the number of bins and value ranges to use in each dimension of the PDF. These parameters can be determined *a priori* (via a configuration file) or changed dynamically throughout the course of a simulation. In terms of inter processe communication, our algorithm is embarrassingly parallel due to the carefully specified sampling region size. Field data and particle data communication is better handled by the simulation because different simulations might have different communication patterns.

PDFs are first generated using the dense matrix representation. More specifically, each PDF is represented as an array where each array element represents a bin of the PDF. The subroutine loops over each grid point in the domain decomposition. The position of the grid point is divided by the sampling region size to get the sampling region index which the grid point belongs to. Then we map the scalar values of the grid point to the corresponding PDF bin by discretizing the scalar values of a grid point according to the PDF bin ranges. For example, when the grid point has the normalized scalar values (temperature: 0.35, pressure: 0.42, density: 0.87) and there are 10 bins per dimension of the 3D PDF with range from 0.0 to 1.0, the grid point is mapped to bin (3, 4, 8). With the corresponding PDF bin calculated, we then increment its frequency by 1. Note that this procedure only requires one loop through the grid points in a given domain decomposition. Depending on the portion of empty bins in each PDF, it may be inefficient to store the PDFs in this dense matrix representation. As a result, the algorithm chooses whether to convert a PDF to its sparse matrix representation depending on which representation results in a smaller storage requirement. The dense matrix representation stores the frequency of each bin by one value, while the sparse matrix representation stores only the frequency of each nonempty bin by two values (frequency and location). Therefore, when the number of nonempty bins is less than half of the total number of bins in a PDF, it is more efficient to store the PDF in its sparse matrix representation. A flag is also stored per PDF to indicate which representation it is using.

Helper arrays are necessary for indexing the PDFs when using the sparse matrix representation. When outputting PDFs in sparse matrix representation, the storage size of each PDF depends on the number of nonempty bins. Since we are concatenating multiple PDFs into a single file in order to reduce the number of files to a manageable amount, we need a way to determine where the information corresponding to a PDF is in the concatenated file. As a result, we also output a helper array per concatenated file to index the individual PDFs. This array requires only one number per PDF in the concatenated file. Our scheme currently concatenates all the PDFs in the same domain decomposition. If that still generates an unmanageable amount of small files, it is also possible to concatenate PDFs from multiple domain decompositions. More investigation in optimizing this process will be explored in future work.

3.1.2 Particle Sorting/Indexing

An in situ subroutine is also provided to sort and index the particles according to the spatial organization of the sampling regions. This ensures more efficient selective access of the particles in post processing. Since the PDFs and the particles both follow the same spatial organization scheme, scientists are able to first perform filtering operations on PDFs in order to reduce the amount of particles being accessed. Otherwise, the entire particle data needs to be loaded for filtering which can be a rather slow operation. Since scientists tend to need to perform many filtering operations in an exploratory fashion to explore new datasets, sorting and indexing particles in situ is essential for efficient analysis of the particle data.

The common spatial organization scheme ensures the particles within the same sampling region of a PDF are contiguous in storage and easy to locate. We provide an additional subroutine to sort and save the particle data which is designed to run independently in each domain decomposition. The algorithm begins by looping over each particle and mapping its spatial location to an associated sampling region. A sampling region ID array is created indicating which sampling region each particle belongs to with one ID per particle. While looping through the particles, the number of particles per sampling region is also calculated into a particle count array. Prefix sums are then calculated from the particle count array to get an sampling region offset array, which indicating where the particles in each sampling region should be in the sorted array. Finally, by utilizing the sampling region ID array and the sampling region offset array, each particle is copied to its unique location in the sorted array. Note that an incrementing index per sampling region is also used here to indicate the particle index within a sampling region segment in the sorted array. This particle sorting algorithm is capable to be GPU accelerated, but the current implementation is only in CPU.

Since the particles are stored contiguous in a single file with multiple sampling regions (which vary in particle count). A helper array is required to efficiently access the selected particles. The helper array denotes where to access the particles within the same sampling region within the file. Specifically, we use the sampling region offset array mentioned above to act as the helper array.

3.2 Post Simulation Analysis and Exploration Tool

The in situ generated PDFs as well as the sorted particle data are used to support efficient analysis of the simulation in post processing. Due to the exploratory nature of the analysis tasks required by the scientists, a high level of interactivity is necessary. As a result, we design and implement an intuitive visualization and analysis tool that can be used to explore the computed PDFs. PDFs, which contain desired data trends, can be selected and used to interactively query the associated particle data for further analysis. An image showing the general interface of our visualization tool can be seen in Figure 3.

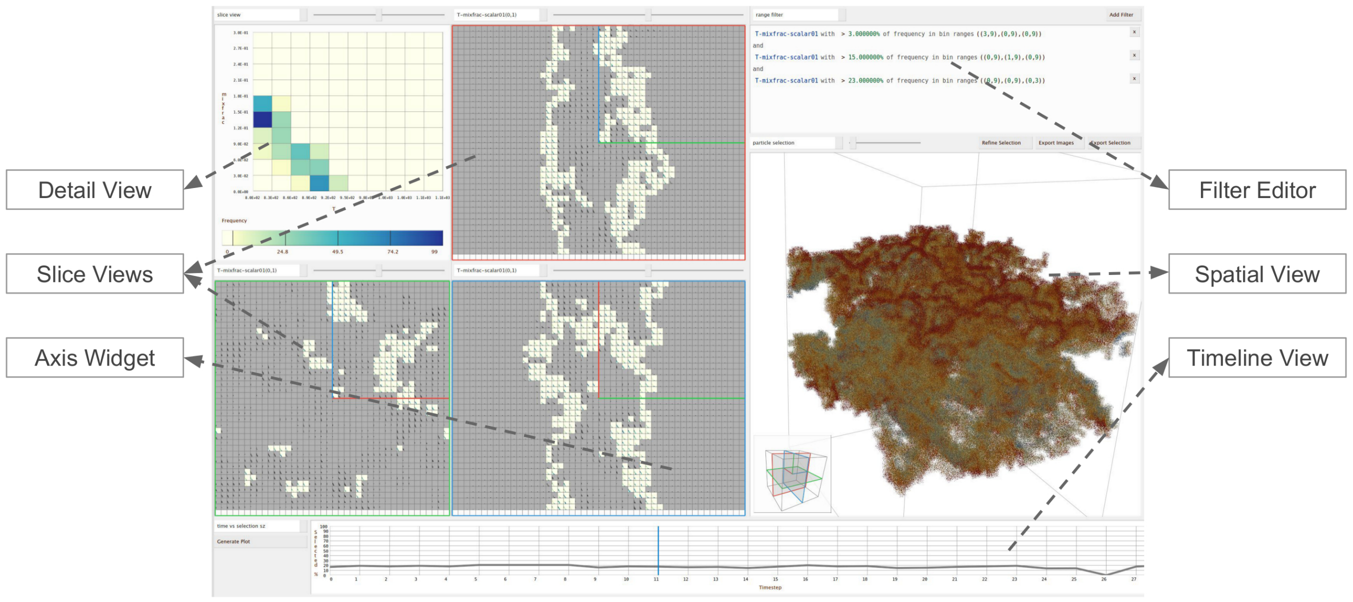
The design of the post hoc visualization and analysis tool closely follows the proposed workflow, which was described in Section 3. First, the in situ generated PDFs per simulation run are loaded to the post hoc tool. The settings of the PDFs and the simulation, such as *sampling regions* and *domain decompositions* are automatically populated from the in situ saved configuration files. The post hoc tool facilitates two ways to browse the loaded PDFs: 1) The PDFs are grouped into slices in order to better show the physical location of a particular PDF, and 2) The timeline widget allows the scientists to view PDFs among different time steps of the simulation. Next, the PDFs are filtered using range-based queries (i.e., whether a desired proportion of samples lie within a certain range). Finally, the particles corresponding to the selected PDFs are loaded and visualized for subsequent particle analysis. Currently the selected subset of particles is exported to other tools in the scientists' workflow. However, future work will focus on integrating desired particle-based analysis techniques directly into our visualization system.

Multiple design decisions are made while designing the user interface. Users are able to choose which dimension to show for each PDF. The PDFs generated in situ can be three dimensional which impose certain difficulty for visualization. Visualizing a 3D PDF directly is often confusing to the users. As a result, we provide controls for users to select which dimensions to plot for a 3D PDF. When only one dimension is chosen, the 3D PDF collapses into a 1D PDF which is then plot as a bar chart. When two dimensions are chosen, the 3D PDF collapses the unchosen dimension and plots the 2D PDF as a heatmap. This way, the users are able to analyze a vairable closely using the 1D PDF and also they are able to explore the correlation between two variables using the 2D PDF.

Grouping PDFs into axis aligned slices in the physical space helps organizing the PDFs in a meaningful way and identifying patterns in the data. The in situ process generates a large amount of PDFs. If we simply show all the PDFs in a list, the users can easily get lost in the ocean of PDFs and potentially identify misleading patterns. For example, if a user wants to examine the spatial neighbors of a specific PDF, the user will have to look through the whole list of PDFs since the neighboring PDFs are not necessarily nearby in the list. With the slice views, the users are able to locate the PDF associated with a specific spatial location. Also, the patterns shown in the slice views are now meaningful in the physical space.

The timeline view aims to help identifying interesting time steps given a specific filtering criteria. After range filters are applied to the current time step, the next dimension to explore is usually the temporal dimension. It is often interesting to see which time steps have the minimum/maximum amount of selected regions. The line chart in the timeline view is specifically aimed for this purpose. The line chart shows the number of selected regions for each time step. As a result, users can directly jump to time steps that are more interesting. Note that generating the timeline line chart is often time consuming because it performs the filtering process to all the time steps. As a result, the line chart is only generated on demand when the user clicks the generate button.

Loading the particles of the selected PDFs can be done interactively since the particle data and the PDFs share the same spatial organization scheme. Given the selected PDF ID, we are able to pinpoint the memory location of the group of particles using the in situ generated helper arrays for the particles. Thus, we are able to efficiently load the particles the scientists are interested in without scanning through the entire particle data. While selecting all the



```
T-mixfrac-scalar01 with  > 3.000000% of frequency in bin ranges ((3,9),(0,9),(0,9))
and
T-mixfrac-scalar01 with  > 15.000000% of frequency in bin ranges ((0,9),(1,9),(0,9))
and
T-mixfrac-scalar01 with  > 23.000000% of frequency in bin ranges ((0,9),(0,9),(0,3))
```

Figure 3: The interface of the post hoc visualization tool is composited of multiple linked views. The detail view shows a visualization of a single PDF selected by hovering the mouse over the slice views. Darker colors represent higher frequency counts in the PDF. Each slice view shows a slice of PDFs on orthogonal cutting planes in the volume. The location of each slice is shown in the axis widget. The X, Y, and Z axes are correspondingly represented with red, green, and blue. The border color of a slice view matches with the border color of the slice in the axis widget in order to show which axis the slice view corresponds to. There are also two colored lines in each slice view to show their orientation. The axis widget also shows the orientation of the spatial view. The filter editor allows scientists to add, edit, and delete range-based filtering criteria. The spatial view shows the particles after filtering. An immediate feedback is given to the scientists in this view and it is important because the filtering process is usually done in an exploratory fashion. Also the greyed out PDFs in the slice views represent the filtering result. Lastly, the timeline view shows the number of filtered PDFs in all time steps and helps the scientists to identify time steps of interest.

PDFs will result in loading the full particle data, in practice, only a small subset of PDFs and particles are of interest, allowing our tool to remain interactive.

4 Results

We implement our techniques directly into the S3D simulation and demonstrate its usability through a combustion research case study and extensive performance evaluations using Lawrence Berkeley National Laboratory's NERSC computing facility.

4.1 Case Study

We used our system on an existing simulation of a temporally evolving mixing layer between n-dodecane and diluted air undergoing ignition. This simulation was performed using the massively parallel DNS code S3D [5] to provide new insights on the physics of combustion processes occurring at conditions relevant to diesel engines. A volume visualization of the simulation is shown in Figure 4. The global domain of the simulation is 1400x1500x1000, while each domain decomposition is responsible for 30x30x25 grid points. $80,000$ computing processors ($5,000$ computing nodes) in Titan were used. Each time step of the field data occupies $564GB$ of storage while each time step of the particle data is $50GB$.

In this case study, a 3D probability distribution function (PDF) is generated from the temperature, mixture fraction, and scalar dissipation of each grid point per domain decomposition. This generates 40x50x40 PDFs for the entire domain. Each PDF uses 10 bins per dimension with a value range of 800–1100 for temperature, 0.0–0.3 for mixture fraction, and 0.0–0.3 for scalar dissipation. Each time step of the PDF data only uses $1.2GB$ of data storage, which is orders of magnitudes smaller than the field data. Due to the much reduced data size, the generated PDFs can be visualized and analyzed on a local workstation. Furthermore, since the particle data is sorted according to the spatial organization scheme, the system can quickly locate and access particles representing specific features of interest.

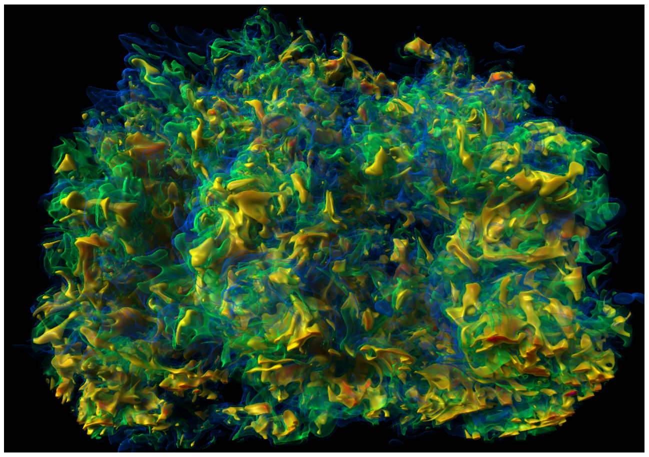


Figure 4: Direct volume rendering of the AirDodecane simulation showing low (green) and high (yellow) regions of the pressure field.

Our visualization tool is able to achieve fluid interactive slicing and filtering on a local workstation due to the reduced data size. By interactively inspecting each PDFs in detail by mousing over them, we are able to quickly scan the PDFs for interesting features and

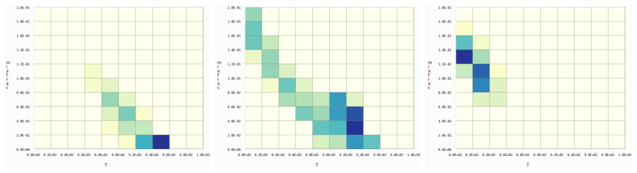


Figure 5: Temperature (horizontal axis) vs. mixture fraction (vertical axis): dark blue represents high frequency while light yellow represents zero frequency. The left PDF is found in the outer region of the volume, which shows that most contents in the sampling region are air, hence low mixture fraction. The right PDF is found in the inner region of the volume, which shows that most contents here are fuel, hence high mixture fraction. These two PDFs confirms the expectations from the scientists. The middle PDF is believed to have some burning contents at low values of mixture fraction due to the bins that deviate from the inverse linear correlation.

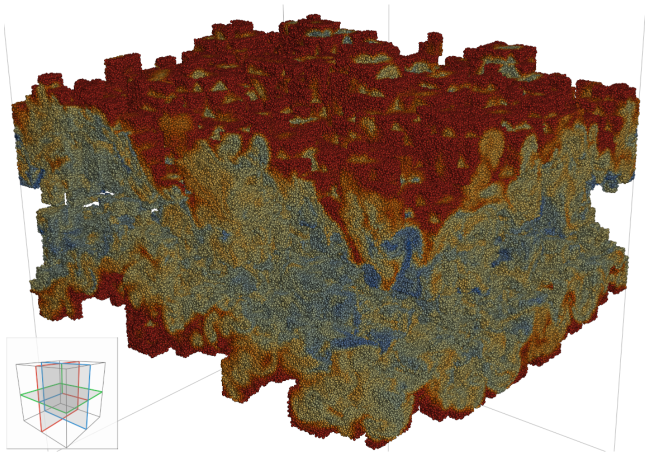


Figure 6: Particles selected by combining multiple range-based filtering criteria. The color represents the temperature of the particles from low (blue) to high (red). Based on knowledge from domain scientists, we used the following criteria: 1) 5% of the samples are within temperature range 920–960, 2) 6% of the samples are within mixture fraction range 0.04–0.2, and 3) 30% of the samples are within scalar dissipation range 0.0–0.14.

trends. Figure 5 shows how the PDFs are able to verify the expectations of the scientists. With advice from the scientists with domain knowledge about the dataset, we are able to setup range-based filtering criteria to precisely select interesting regions. Figure 6 shows a rendering of the user selected particles. In problems where a non-premixed flow undergoes ignition, high values of scalar dissipation are responsible for delaying the ignition event. By selecting particles based on different scalar dissipation ranges, scientists can assess how different levels of mixing rate have a different impact on chemical reactions. Furthermore, they can also use this data as the starting point for more sophisticated time-varying analysis of their features of interest.

4.2 Performance Results

The benefit of our in situ analysis algorithm is that it prepares the data for more accurate and efficient analysis in post processing. However, by doing in situ analysis, our algorithm imposes a certain overhead in both computation and storage. In this section, we demonstrate the usability of such in situ analysis by comparing the in situ overhead with post processing benefit.

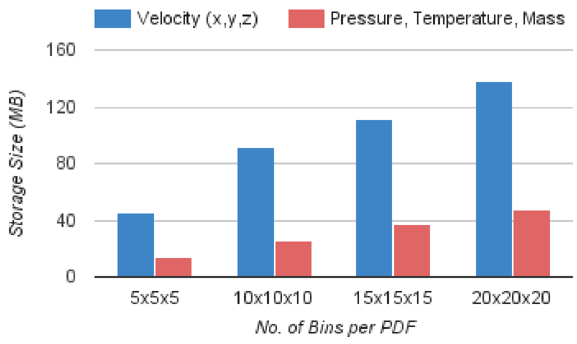


Figure 7: The storage sizes of in situ generated 3D PDFs with different number of bins per PDF and different variables. For all eight simulation runs, the domain decomposition size is set to 20x20x20 and the global domain size is 400x400x400. The number of PDFs per domain decomposition is 4x4x4. The horizontal axis represents the number of bins per PDF while the vertical axis represents the actual output file size of the PDFs. With a larger amount of bins in a PDF, it is more likely to have more bins which contain information. As a result, the size of the sparse matrix representation of the PDF increases. The blue bars represent PDFs generated using the three components of the velocity field while the red bars represent PDFs generated using pressure, temperature, and mass as variables. Using different variables creates different data distribution affecting the size of the generated PDFs. For reference, the storage size of the 6,435,078 particles per output step in this S3D test run is about 2GB, which is magnitudes larger than the storage size of the PDFs.

4.2.1 Probability Distribution Function Overhead

The storage requirement of the probability distribution functions (PDFs) depends on the number of bins per sampling region. In the case where we store the PDFs in the dense matrix representation, each bin of a PDF is stored as an element in the array that represents the PDF, thus giving us a well defined storage size. The storage size can be computed with nm, where n is the number of bins per PDF and m is the number of PDFs that are generated. However, after converting the PDFs to the sparse matrix representation, the storage size is no longer constant anymore.

With sparse matrix representation, instead of only depending on the number of bins per PDF, the storage requirement also depends on the scalar value distribution within sampling regions. Sparse matrix representation means only the non-zero bins are recorded. When the scalar values are uniformly distributed within the sampling regions, they are likely to land in different bins of the PDF. As a result more bins are likely to be non-zero leading to a higher storage size. In contrast, if the scalar values are highly skewed within the sampling region, only a small fraction of the bins will be non-zero leading to a lower storage size. As a result, the worst case of the storage size is $O(nm)$, where n is the number of bins per PDF and m is the number of PDFs that are generated. Practically, the worst case storage size is very unlikely to happen. Although the correlation between storage size and number of bins per PDF is no longer linear, the number of bins does still affect the storage size because the increased granularity of the PDF leads to more bins with information. Figure 7 shows the storage size of the probability distribution functions in four runs of the S3D simulation with different number of bins per PDF. We can see that different number of bins and different data distribution affects the storage size as described above. Even with 3D probability distribution functions, the storage size is still minimal compared to the particle output.

Our test simulation runs also show that the computational over-

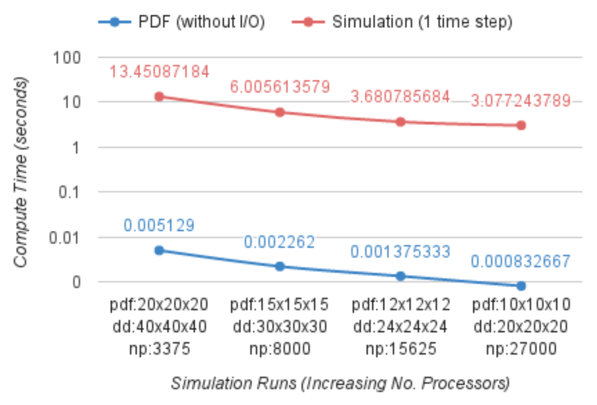


Figure 8: A strong scaling test of the compute time spent on in situ PDF generation vs. a standard simulation time step. Four simulation runs are performed with the same global domain but different domain decomposition size and different number of bins in PDFs. The global domain size is set to 600x600x600, and the number of PDFs per domain decomposition is 2x2x2. In both figures, *pdf* represents the number of bins per PDF, *dd* represents the domain decomposition size, and *np* represents the number of processors. The PDF resolution decreases accordingly with the domain decomposition size. The compute time of generating PDFs is measured without I/O. We can see the PDF generation time is lower than the simulation time by orders of magnitude even for a single time step.

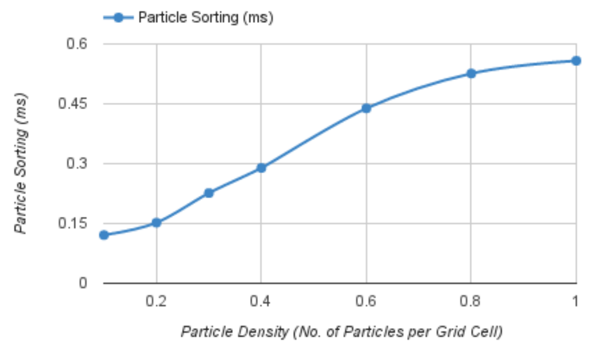


Figure 9: The time used for particle sorting with different particle densities. Particle density is defined here as the average number of particles per grid cell. The total number of grid cells is set to 400x400x400. The domain decomposition size is fixed to 20x20x20 and the number of PDFs per domain decomposition is fixed to 4x4x4. Even with particle density set to 1, which is 8000 particles per domain decomposition and 64 million total particles, particle sorting only takes 0.558 milliseconds. For reference, a time step of the simulation takes about 2 seconds. The particle sorting uses less than 0.02% of the simulation time.

head imposed by in situ PDF generation is only a small fraction of the overall simulation time. The algorithm contains two loops: one that loops through all the grid points to generate the PDFs, and another that loops through all the bins in all the probability distribution functions to compact them into sparse matrix representation. As a result, the time complexity is $O(n+mb)$, where n is the number of grid points per domain decomposition, m is the number of PDFs per domain decomposition, and b is the number of bins per probability distribution function. In practice, n and mb tend to be similar in magnitude. Figure 8 shows a strong scaling test on in situ PDF generation time and simulation time. We can see the PDF generation time only takes a small fraction of the simulation time.

4.2.2 Particle Sorting/Indexing Overhead

The storage overhead of particle processing is negligible compared to the standard output size of the simulation. The sorted particle file per domain decomposition is the same size as the unsorted particle file since it contains the same information, but in a different order. However, our algorithm outputs an additional index file per domain decomposition for quickly locating particles in the sorted particle file. The amount of storage required for the index file depends on the number of PDFs in the domain decomposition. Specifically, two integers per PDF are output to the index files. Since the number of PDFs is orders or magnitude smaller than the number of particles and since each particle can contain hundreds of double precision floating point values, the storage overhead of these additional index files is negligible in comparison.

The computational overhead of in situ particle processing arises from the particle sorting that is done per domain decomposition. The runtime cost for particle sorting depends on the number of particles and the number of PDFs. The time complexity is $O(n+m)$, where n is the number of particles and m is the number of probability distribution functions. In most cases, n is magnitudes larger than m. Figure 9 shows that the time spent sorting the particles is linearly related to particle density, which is defined here as the average number of particles per grid cell. When the particle density is 1, which translates to 64 million particles given a 400x400x400 grid, the particle sorting time is only 0.558 milliseconds. For reference, one time step of a typical run of the S3D simulation with a grid size of 400x400x400 takes 2 seconds. That is, the particle sorting algorithm only takes an extremely small fraction of the simulation time even if we choose to sort/output the particles for every time step.

4.2.3 In Situ vs. Post Hoc PDF Generation

In this section we justify the need to generate the probability distribution functions (PDFs) in situ from the available field data. While PDFs can be generated post hoc from the dumped particle data, generating them in situ is more efficient and more accurate. Note that field data is not generally produced by S3D since it is much too large to be output at a regular temporal frequency.

Sorting particles and generating PDFs in situ primarily saves time on I/O. The major steps to sort particles and generate PDFs post hoc are 1) load unsorted particles from disk, 2) sort particles, 3) generate PDFs from the sorted particles, and 4) output sorted particles and PDFs. There are two major differences from the in situ procedure. First, the PDFs are generated from particle data instead of field data. The post hoc procedure loops over each particle instead of each grid point, thus the runtime complexity becomes $O(p+mb)$, where p is the number of particles, m is the number of PDFs, and b is the number of bins per PDF. Usually, $p=Cn$, where C can be considered a constant particle density value and n is the number of grid points. As a result, the runtime complexity of the post hoc procedure is the same as the in situ procedure. Second, two extra I/O steps must take place to load the unsorted particles and save the sorted particles. This becomes less feasible as billions of particles can be output by current simulations. The time it takes to load and output the particles can easily become a new bottleneck. Figure 10 shows a weak scaling timing comparison between the in situ procedure and the post hoc procedure. We can see that the post hoc procedure is more time consuming due to the two extra I/O steps.

In situ generated PDFs are not only more efficient but also more accurate. The difference between the in situ and post hoc generated probability distribution functions is the data source: the in situ

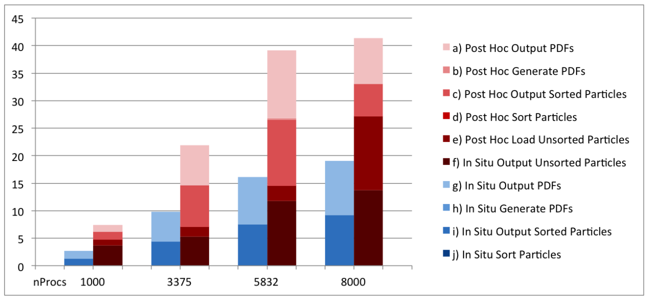


Figure 10: Weak scaling timing comparison between the in situ procedure and the post hoc procedure. The number of grid points per domain decomposition is set to 8000 (20x20x20) and the particle density per grid point is roughly 1.0. The times for sorting particles and generating PDFs (b, c, h, j) are small compared to the I/O times and are barely visible in the graph. We can see the time to output PDFs and sorted particles (a, c, g, i) are similar in both in situ and post processing. However, the post processing version requires two more I/O steps, outputting and loading unsorted particles (e, f).

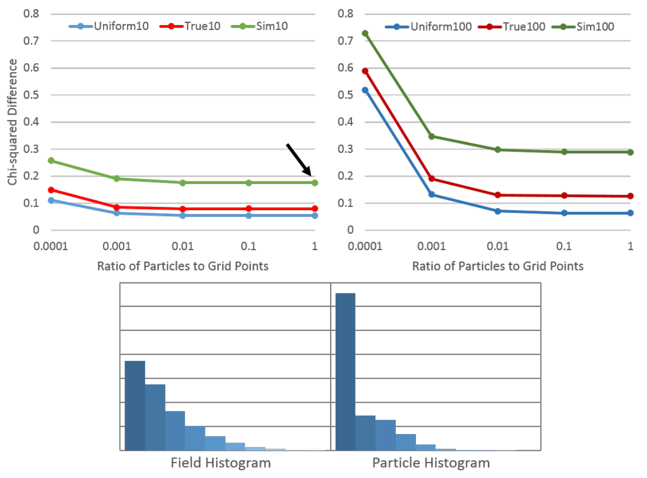


Figure 11: Top) Graphs showing the chi-squared difference between a 1D histogram (PDF) generated from the full resolution simulation grid (ground truth) to one generated by a particle subset. Three particle distributions were used: particles seeded using a uniform random distribution (blue), particles seeded using a true random distribution (red), and the real simulation particles (green). The plot on the left represents a 10 bin histogram while the plot on the right represents a 100 bin histogram. The uniform random seeding results in the smallest difference, whereas the real simulation particles results in the largest difference. Furthermore, reducing the number of available particles or increasing the number of bins also increases the difference. Bottom) A representation of the resulting histograms for the data point indicated by a black arrow. Note that all histograms we normalized to account for total frequency differences that arise from reducing the number of particles. These results demonstrate the need to generate histograms in situ using the simulation grid rather than post hoc using the particle data.

procedure uses the field data while the post hoc procedure uses the particle data. Generating from field data is more accurate because we can then select particles based on the high resolution field information. This accuracy difference depends on the purpose of the post hoc analyses. In our case, the scientists want to identify interesting regions in the field that matches certain criteria and then analyze the particles within these regions. As a result, the in situ generated PDFs are more accurate. Using the the field data allows one to take uniformly distributed samples. If the particles are uniformly distributed in the volume, the generated PDFs should be very similar. However, since the particles are rarely uniformly distributed, errors in PDFs generated post hoc can arise. Figure 11 shows the difference between a probability distribution function generated from the field data (which can only be done in situ) and those generated from particle data (which can be done post hoc) over the whole domain. We can see that using simulation particles results in a PDF that is very different and therefore noticeably less accurate from one that is generated directly from the field data.

4.2.4 Post Processing Timing

Lastly, we compare the overall time to extract a desired subset of particles using the PDFs against the original method of filtering from the particle data directly. The process consists with the filtering step and the particle extraction step. Filtering checks if a particle matches the specified criteria and then the particle is extracted/copied to the output array. In the original scheme, both steps are performed in a single for loop which iterates over the full particle data. As a result, the time to load and extract a particle subset remains constant. In our method, the filtering is no longer done per particle, instead it is performed using the in situ generated PDFs. Because all the PDFs have to be visited, the filtering process takes a constant time. After the PDFs are filtered, we have already identified the subset of particles that we want to extract. As a result, the time it takes to extract the particles reduces depending on how large the subset is. The timing results comparing the original scheme and our method can be seen in Figure 12. As long as the number of particles that needs to be extracted is smaller than (nearly) the full dataset, our scheme performs much faster than the original one, allowing scientists to interactively extract various particle subsets.

5 Discussion

The scientists we work with confirm that our approach provides a very quick and convenient way of exploring correlations in the simulation data. Our visualization tool really thrives at enabling them to quickly test their ideas, such as determining the dependencies among variables. When the result from our visualization looks encouraging, more complicated and time consuming analysis can then be carried out with higher confidence.

5.1 In Situ vs. Post Processing

One of the major driving motivations behind the development of this work was the inability of scientists to predict all the types of trends that interest them. As a result, a large majority of their exploratory investigation involves a trial and error approach which cannot be moved entirely into an in situ analysis. By incorporating a hybrid of in situ and post processing, we can accelerate the rate at which scientists perform their exploration of the data.

In our approach, the in situ processing (PDF generation and particle sorting) acts as both a data reduction technique and a means of significantly accelerating the post processing analysis (which must be done in a trial and error fashion). Furthermore, the full particle data must be saved to disk because the scientists do not have a priori knowledge of which particle subsets may be of interest to them. The in situ processing we perform allows the scientists to quickly work with the large particle data in an efficient manner.

Note our implementation also has the ability to generate in situ particle-based PDFs, however we focus primarily on examples using in situ generated PDFs from the field data because 1) it is simply not feasible to save the field data from the simulation at regular time intervals and 2) the larger number of equally spaced samples in the field data results in a more accurate representation of data trends.

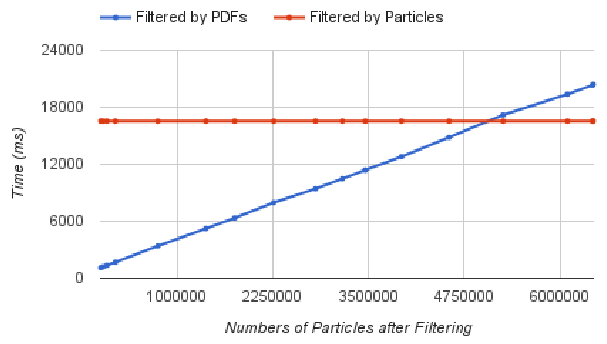


Figure 12: Filtering particles by iterating through each particle vs. iterating through each PDF. The dataset consists of 6.4 million particles before filtering, while there are only 0.5 million PDFs. The filtered by particles line includes the time to load the particles and to apply the filtering criteria to each particle. The time required to filter by particles remains constant because all the particles have to be visited exactly once. On the other hand, the filtered by PDFs line includes the time to load the PDFs, to filter by PDFs, and to load the filtered particles. Filtering by PDFs only requires visiting and loading the filtered particles, thus forming a linearly increasing line.

5.2 Temporal Analysis of Particle Subsets

The end result of this workflow is a subset of particles that match a specific set of potentially interesting simulation data trends. The next step involves a detailed analysis of the temporal properties of the subset. Currently the scientists have an alternate set of tools that they use to conduct this time varying analysis. Our system currently acts as a mediatory tool to accelerate the rate at which these subsets can be extracted from the full data.

In the future, we plan to incorporate these additional temporal analysis tasks directly into our visualization software. Currently, scientists are limited to exporting the particle selections to the other tools in their workflow and eliminating this step will streamline the scientific process. These additional tasks will include temporal analyses of particle trajectories in both physical and phase space and will allow scientists to gain a better insight into the evolution of extracted features of interest.

5.3 Usability

There are two major aspects which characterize usability of our approach. Firstly, interactivity is essential for post hoc visualization and analysis because the particle selection task requires an exploratory process. Since the probability distribution functions (PDFs) are generated in situ, we effectively prepare the massive simulation data for interactive post hoc exploration, as discussed in Section 4.2.4. Secondly, accuracy of the features represented by the selected PDFs and particles is much higher since the PDFs are generated using the full resolution field data, as demonstrated in Section 4.2.3.

Furthermore, since our in situ technique is embarrassingly parallel, the PDF generation procedures are local to a domain decomposition, which makes it really easy to adapt to any simulation. We also provide our routines in such a way that can process both field data and particle data, making our tools applicable to a wide variety of simulation types. The examples in this paper focus on the combustion simulation results of S3D as it was the motivation for developing this work. In the future, we plan to test how easily our system can be adapted to other case studies with little to no modification to the techniques themselves.

5.4 Limitations and Other Future Work

There are some limitations to our technique. First, it is possible that particles may not be uniformly distributed among the domain decompositions resulting in performance bottlenecks in sorting. However, such a case has not occurred in the simulation runs we performed with the S3D simulation. Secondly, there is a limit in the available resolution of the PDFs, both in physical and variable space. There will always be a point where the number of PDFs and bins becomes so large that it is more advantageous to simply dump the full resolution field data. Users will have to choose a balance between the spatial resolution of the PDF decomposition and the number of bins in each PDF.

Next, limits in the spatial resolution of the PDFs will limit the spatial detail through which particles can be selected. Future work will focus on using additional passes over the selected particle subsets to further refine the features of interest. For example, we can use the variables found in the particle data itself to filter out any unwanted particles bypassing the spatial resolution limits of each PDF. Such a procedure can still be interactive since we have already reduced the particle data in the initial filtering step and can exploit the embarrassingly parallel nature of such a task.

As for other future work, we also plan to reduce the number of files necessary to represent the PDFs and particle data, which are saved separately per domain decomposition. This will be done by concatenating information from neighboring domain decompositions when the number of files becomes unmanageable. We also plan on using the in situ generated PDFs to perform additional tasks, such as entropy estimation, and more advanced analysis, such as importance driven time step selection. This can allow the simulation to intelligently choose certain temporal ranges at which to save information more frequently.

6 Conclusion

Overall, we present a hybrid in situ and post processing method which uses probability distribution functions (PDFs) and a reorganization of particle data to aid in simulation analysis. This coupling between PDFs and particles enables scientists to explore their simulation and accurately select the desired particle-based features in an interactive exploratory fashion. We also develop a visualization tool which can be used to perform each of these features in an intuitive manner. We evaluate the usability of our method using a real combustion simulation and demonstrate the increases in task efficiency and accuracy that the new workflow provides to scientists.

Acknowledgements

This work was supported in part by the U.S. Department of Energy under grants DE-FC02-12ER26072 and DE-SC0012610, and also by the U.S. National Science Foundation through grant IIS-1528203. The experimental study has used resources of the NERSC facility at Lawrence Berkelty National Laboratory and the OLCF facility at Oak Ridge National Laboratory.

References

[1] S. Ahern, A. Shoshani, K.-L. Ma, A. Choudhary, T. Critchlow, S. Klasky, V. Pascucci, J. Ahrens, E. W. Bethel, H. Childs, et al. Scientific discovery at the exascale. report from the DOE ASCR 2011 workshop on exascale data management. 2011.

[2] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *Proc. of SC*, pages 424–434, 2014.

[3] U. Ayachit, A. Bauer, B. Geveci, P. O'Leary, K. Moreland, N. Fabian, and J. Mauldin. Paraview catalyst: Enabling in situ data analysis and visualization. In *Proc. of ISAV*, pages 25–29. ACM, 2015.

[4] T. Biedert and C. Garth. Contour tree depth images for large data visualization. In *Proc. of EGPGV*, 2015.

[5] J. H. Chen, A. Choudhary, B. De Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. Liao, K.-L. Ma, J. Mellor-Crummey, N. Podhorszki, et al. Terascale direct numerical simulations of turbulent combustion using S3D. *Computational Science & Discovery*, 2(1), 2009.

[6] H. Childs, K.-L. Ma, H. Yu, B. Whitlock, J. Meredith, J. Favre, S. Klasky, N. Podhorszki, K. Schwan, M. Wolf, M. Parashar, and F. Zhang. In situ processing. In *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*, 2012.

[7] M. Dorier, R. R. Sisneros, T. Peterka, G. Antoniu, and D. B. Semeraro. Damaris/viz: a nonintrusive, adaptable and user-friendly in situ visualization framework. In *Proc. of LDAV*, Oct 2013.

[8] O. Fernandes, S. Frey, F. Sadlo, and T. Ertl. Space-time volumetric depth images for in-situ visualization. In *LDAV*, pages 59–65, 2014.

[9] A. Kageyama and T. Yamada. An approach to exascale visualization: Interactive viewing of in-situ visualization. *CPC*, pages 79 – 85, 2014.

[10] J. Kim, H. Abbasi, L. Chacn, C. Docan, S. Klasky, Q. Liu, N. Podhorszki, A. Shoshani, and K. Wu. Parallel in situ indexing for data-intensive computing. In *Proc. of LDAV*, pages 65–72, Oct 2011.

[11] S. Klasky, H. Abbasi, J. Logan, M. Parashar, K. Schwan, A. Shoshani, M. Wolf, S. Ahern, I. Altintas, W. Bethel, L. Chacon, C. Chang, J. Chen, H. Childs, J. Cummings, S. Ethier, R. Grout, Z. Lin, Q. Liu, X. Ma, K. Moreland, V. Pascucci, N. Podhorszki, N. Samatova, W. Schroeder, R. Tchoua, J. Wu, and W. Yu. In situ data processing for extreme scale computing. In *SciDAC*, 2011.

[12] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C. S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. Isabela for effective in situ compression of scientific data. *CCPE '13*, pages 524–540.

[13] A. G. Landge, V. Pascucci, A. Gyulassy, J. C. Bennett, H. Kolla, J. Chen, and P. T. Bremer. In-situ feature extraction of large scale combustion simulations using segmented merge trees. In *SC*, 2014.

[14] J. Lofstead, F. Zheng, S. Klasky, and K. Schwan. Adaptable, metadata rich io methods for portable high performance io. In *IPDPS*, 2009.

[15] K.-L. Ma, C. Wang, H. Yu, and A. Tikhonova. In-situ processing and visualization for ultrascale simulations. *JPCS*, 78(1):012043, 2007.

[16] T. Neuroth, F. Sauer, W. Wang, S. Ethier, and K.-L. Ma. Scalable visualization of discrete velocity decompositions using spatially organized histograms. In *Proc. of LDAV*, pages 65–72, Oct 2015.

[17] M. Novotny and H. Hauser. Outlier-preserving focus+context visualization in parallel coordinates. *TVCG*, pages 893–900, Sept 2006.

[18] E. R. Schendel, S. V. Pendse, J. Jenkins, D. A. Boyuka, II, Z. Gong, S. Lakshminarasimhan, Q. Liu, H. Kolla, J. Chen, S. Klasky, R. Ross, and N. F. Samatova. Isobar hybrid compression-I/O interleaving for large-scale parallel I/O optimization. In *Proc. of HPDC*, 2012.

[19] R. R. Sinha and M. Winslett. Multi-resolution bitmap indexes for scientific data. *ACM TODS*, 32(3), Aug. 2007.

[20] Y. Su, Y. Wang, and G. Agrawal. In-situ bitmaps generation and efficient data analysis based on bitmaps. In *HPDC*, pages 61–72, 2015.

[21] D. Thompson, J. A. Levine, J. C. Bennett, P. T. Bremer, A. Gyulassy, V. Pascucci, and P. P. Pbay. Analysis of large-scale scalar data using hixels. In *Proc. of LDAV*, pages 23–30, Oct 2011.

[22] A. Tikhonova, H. Yu, C. D. Correa, J. H. Chen, and K.-L. Ma. A preview and exploratory technique for large-scale scientific simulations. In *Proc. of EGPGV*, pages 111–120, 2011.

[23] V. Vishwanath, M. Hereld, and M. E. Papka. Toward simulation-time data analysis and i/o acceleration on leadership-class systems. In *Proc. of LDAV*, pages 9–14. IEEE, 2011.

[24] B. Whitlock, J. M. Favre, and J. S. Meredith. Parallel in situ coupling of simulation with a fully featured visualization system. In *EGPGV*, pages 101–109, 2011.

[25] J. Woodring, M. Petersen, A. Schmeisser, J. Patchett, J. Ahrens, and H. Hagen. In situ eddy analysis in a high-resolution ocean climate model. *IEEE TVCG*, 22(1):857–866, Jan 2016.

[26] Y. C. Ye, Y. Wang, R. Miller, K.-L. Ma, and K. Ono. In situ depth maps based feature extraction and tracking. In *LDAV*, pages 1–8, 2015.

[27] H. Yu and K.-L. Ma. A study of i/o methods for parallel visualization of large-scale data. *Parallel Computing*, 31(2):167 – 183, 2005.

[28] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K. L. Ma. In situ visualization for large-scale combustion simulations. *CG&A*, 2010.

[29] F. Zheng, H. Abbasi, C. Docan, J. Lofstead, Q. Liu, S. Klasky, M. Parashar, N. Podhorszki, K. Schwan, and M. Wolf. Predata - preparatory data analytics on peta-scale machines. In *IPDPS*, 2010.