Combining In-situ and In-transit Processing to Enable Extreme-Scale Scientific Analysis

Janine C. Bennett[*], Hasan Abbasi[†], Peer-Timo Bremer[‡], Ray Grout[§], Attila Gyulassy[¶],
Tong Jin[||], Scott Klasky[†], Hemanth Kolla[*], Manish Parashar[||], Valerio Pascucci[¶],
Philippe Pebay[**], David Thompson[**], Hongfeng Yu[*], Fan Zhang[||], and Jacqueline Chen[*]
[*]Sandia National Laboratories, [†]Oakridge National Laboratory, [‡]Lawrence Livermore National Laboratory
[§]National Renewable Energy Laboratory, [¶]University of Utah, [||]Rutgers University, [**]Kitware

Abstract—**With the onset of extreme-scale computing, I/O constraints make it increasingly difficult for scientists to save a sufficient amount of raw simulation data to persistent storage. One potential solution is to change the data analysis pipeline from a post-process centric to a concurrent approach based on either in-situ or in-transit processing. In this context computations are considered in-situ if they utilize the primary compute resources, while in-transit processing refers to offloading computations to a set of secondary resources using asynchronous data transfers. In this paper we explore the design and implementation of three common analysis techniques typically performed on large-scale scientific simulations: topological analysis, descriptive statistics, and visualization. We summarize algorithmic developments, describe a resource scheduling system to coordinate the execution of various analysis workflows, and discuss our implementation using the DataSpaces and ADIOS frameworks that support efficient data movement between in-situ and in-transit computations. We demonstrate the efficiency of our lightweight, flexible framework by deploying it on the Jaguar XK6 to analyze data generated by S3D, a massively parallel turbulent combustion code. Our framework allows scientists dealing with the data deluge at extreme scale to perform analyses at increased temporal resolutions, mitigate I/O costs, and significantly improve the time to insight.**

I. Introduction

While the steady increase in available computing resources enables ever more detailed and sophisticated simulations, I/O constraints are beginning to impede their scientific impact. Even though the time scales resolved by modern simulations continue to decrease, the length between time steps saved to disk typically increases. For example, turbulent combustion direct numerical simulations (DNS) currently resolve intermittent phenomena that occur on the order of 10 simulation timesteps (e.g., the creation of ignition kernels). However, in order to maintain I/O overheads at a reasonable level, typically only every 400th timestep is saved to persistent storage for post-processing and, as a result, the data pertaining to these intermittent phenomena is lost. Figure 1 illustrates such subtle vortical structures identified in a large and complex flow field of turbulent combustion. This problem is widely predicted to become even more pressing on future architectures, motivating a fundamental shift away from a post-process centric data analysis paradigm.

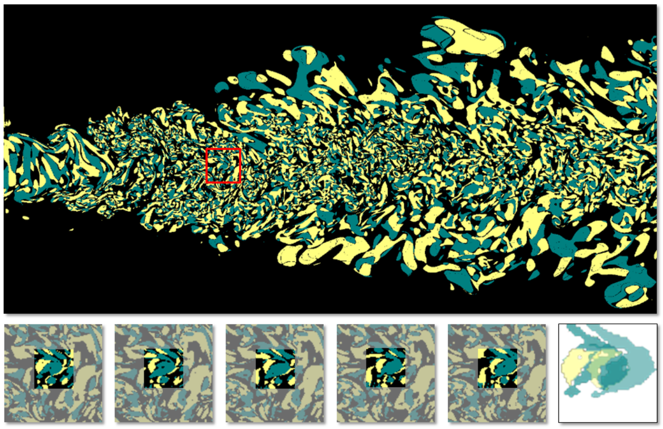

Fig. 1: Top: A small vortical structure in a turbulent flow field is highlighted in the red box. Bottom: The highlighted structure is tracked over time (left 5 images). The right-most image shows the overlap between the 1st and 5th images. Such connectivity indicators are lost with conventional post-processing when the temporal length-scale of features is shorter than the frequency at which data is written to disk.

One promising direction is to move towards a concurrent analysis framework in which raw simulation data is processed as it is computed, decoupling the analysis from file I/O. The two most commonly considered variants are in-situ and in-transit processing. Both are based on the idea of performing analyses as the simulation is running, storing only the results, which are usually several orders of magnitude smaller than the original, and thus mitigating the effects of limited disk bandwidth or capacity. Their difference lies in how and where the computation is performed. In-situ analysis typically shares the primary simulation compute resources. In contrast, when analyses are performed in-transit, some or all of the data is transferred to different processors, either on the same machine or on different computing resources all together.

Both of these approaches have inherent advantages and disadvantages. In principle, in-transit analysis minimally impacts the scientific simulation. By using asynchronous data transfers to offload computations to secondary resources, the simulation can resume operation much more quickly than if it were to wait for a set of in-situ analyses to complete. However, in practice, transferring even a subset of the raw data over the network may become prohibitive, and furthermore, the memory and/or

SC12, November 10-16, 2012, Salt Lake City, Utah, USA
978-1-4673-0806-9/12/$31.00 ©2012 IEEE

computing capabilities of the secondary resources can quickly be surpassed. In-situ analyses are not faced with the same resource limitations because the entirety of the simulation data is locally available. However, scientists will typically tolerate only a minimal impact on simulation performance, which places significant restrictions on the analysis. First, simulations are often memory bound and thus all analyses must operate within a very limited amount of scratch space. Second, the analysis is usually allotted only a short time window to execute. The latter restriction is particularly challenging as many data analysis algorithms are global in nature and few are capable of scaling satisfactorily.

To address these challenges, this paper proposes a hybrid approach based on decomposing analysis algorithms into two stages: a highly efficient and massively parallel in-situ stage, and a small-scale parallel or serial in-transit stage connected via a transparent data staging framework. The key insight is that many analysis algorithms can be formulated to perform various amounts of filtering and aggregation, resulting in a set of intermediate data that is often orders of magnitude smaller than the raw simulation output. Asynchronously transferring these partial results, we are able to both minimize simulation impact and reduce in-transit data transfer costs. We demonstrate our framework using three common post-processing tasks: visualization, descriptive statistical summaries, and a sophisticated topological analysis. All three approaches perform an entirely local set of in-situ computations and transfer their intermediate results asynchronously to a staging area where computations are completed in-transit. The staging framework automatically pipelines in-transit computations using different processes for successive time steps via a pull-based scheduling model to manage execution heterogeneity. This almost entirely decouples the time necessary to complete the analysis of a time step from the time required to advance the simulation. In particular, we demonstrate how our framework enables analysis and visualization of a large-scale combustion simulation at temporal frequencies infeasible for traditional post-processing approaches, while minimizing impact on the primary simulation. Our contributions in detail are:

- A new formulation of three common analysis approaches into a massively parallel in-situ and a small-scale or serial in-transit stage;
- A flexible data staging and coordination framework to transparently transfer intermediate data from the primary to a set of secondary computing resources;
- A temporally multiplexed approach to decouple the performance of the analysis from that of the simulation; and
- A case study demonstrating a wide range of analyses applied to a large-scale turbulent combustion simulation at unprecedented temporal frequencies.

Overall, our framework represents a crucial first step towards a practical approach for the concurrent analysis of massively parallel simulations. Our approach is flexible, extensible to a wide range of analysis algorithms, applicable to virtually all high performance computing environments, and promises to significantly improve the time to insight for modern scientific simulations.

II. Related Work

In-situ and In-transit processing: The increasing performance gap between compute and I/O capabilities has motivated recent developments in both in-situ and in-transit data processing paradigms. Largely data-parallel operations, including visualization [1]–[5], and statistical compression and queries [6], have been directly integrated into simulation routines, enabling them to operate on in-memory simulation data. Another approach, used by FP [7] and CoDS [8], performs in-situ data operations on-chip using separate dedicated processor cores on multi/many-core nodes.

The use of a data staging area, i.e., a set of additional compute nodes allocated by users when launching parallel simulations, has been investigated in projects such as DataStager [9], PreDatA [10], JITStaging [11], DataSpaces [12]/ActiveSpaces [13], and Glean [14]. Most of these existing data staging solutions primarily focus on fast and asynchronous data movement off simulation nodes to lessen the impact of expensive I/O operations. They typically support limited data operations within the staging area, such as pre-processing, and transformations, often resulting in under-utilization of the staging nodes' compute power. In contrast, we present a hybrid in-situ/in-transit processing framework in which a multi-stage pipeline supporting various simultaneous analyses fully utilizes both the data buffering and computation capabilities of staging nodes.

Analytics: Visualization is a largely data-parallel operation that has been the focus of many of the existing in-situ analytics efforts. Among the earlier work are several parallel run-time visualizations whose problem and system scales were fairly small [15]–[17]. One of the primary advantages of simulation-time visualization is the ability it grants scientists to visually monitor their simulation while it is running. For example, SCIRun [18] provides a computational steering environment that supports run-time simulation tracking. Tu et al. [1] were the first to demonstrate how to effectively monitor a terascale earthquake simulation running on thousands of processors of a supercomputer. Over a wide-area network, they were able to interactively change visualization parameters used to visually monitor simulation runs [2]. Yu et al. [3] demonstrate in-situ visualization of particle and scalar field data from a large-scale combustion simulation, creating a scalable solution in which in-situ visualization only accounts for a small fraction of overall simulation time. Recent efforts also allow for the coupling of simulation codes with popular visualization tools, VisIt [4] and ParaView [5]. Both works aim to reduce integration efforts required by the user and minimize performance impact to the simulation.

Descriptive statistics are a common tool used by scientists to provide succinct summaries of their data. The R [19] software package contains a subset of algorithms which have been fully parallelized [20]. The work of [6] provides a framework for performing statistical queries on massive data. This paper describes the in-situ and in-transit deployment of scalable parallel statistics algorithms [21]–[23], in the VTK library [24], whose use had previously been reported for post-processing purposes.

Topology-based techniques have proven useful in the analysis of a wide variety of simulation data due to their efficient representation of the feature space of a scalar function [25]–[30]. Reeb graphs [31] and their variants, contour trees and merge trees [32], encode the level set behavior of a function, while Morse- and Morse-Smale complexes [33]–[35] represent gradient flow information. Both sets of approaches provide multi-scale, condensed representations of relevant features. However, their construction is inherently not data-parallel and existing algorithms do not scale such that they can be deployed entirely in-situ. Techniques to compute topological structures for large-scale data fall into two categories: 1) streaming out-of-core approaches, such as for of Reeb-graphs [36] and merge trees [37], and 2) divide-and-conquer parallel approaches, such as for contour trees [38] and Morse-Smale complexes [39], [40] that rely on k-nary merging of regions of the domain.

III. Hybrid in-situ/in-transit Analytics

Central to our framework is the notion of decomposing analysis algorithms into separate in-situ and in-transit stages. Ideally, the first in-situ stage should be entirely data-parallel, using only data already available on the local compute nodes. Furthermore, in-situ computations should use a limited amount of memory, execute sufficiently fast relative to the performance of the simulation itself, and most importantly should significantly reduce the data sent to the in-transit stage. This second stage must be able to execute solely using the data transferred from the first stage and must operate within the memory and processing constraints of available secondary compute resources. In practice, the fastest sustainable analysis frequency is limited by memory and processing constraints on the secondary system. This hybrid formulation naturally optimizes the use of system resources, and provides scientists a mechanism for elaborate prioritization to ensure accomplishment of time-critical tasks.

Fortunately, a large class of analysis and visualization algorithms can be rewritten according to these guidelines. In particular, there has been an increased focus on online, streaming algorithms [21], [41], [42] which naturally – and in some cases unaltered – can be reframed to satisfy this model. The following section provides a description of three commonly used post-processing algorithms and discusses how they have been adapted to our framework. These range from highly data-parallel descriptive statistics and visualization, to an advanced topological feature extraction technique with complex, global communication requirements.

Visualization: In this paper we compare the behavior of two visualization algorithms. The first of these is an entirely in-situ volume rendering approach, whose design is similar to our previous work [3], that renders full-resolution data on shared compute nodes with the simulation. This parallel rendering approach is very efficient and generates high-quality images that visually convey the results of large-scale simulations in great detail. However, we observe that for monitoring and verification purposes, lower-resolution images are sufficient. Such images can be generated using secondary compute resources, minimizing direct impact on the simulation. Thus, the second algorithm is a hybrid in-situ/in-transit approach that first down-samples the full-resolution combustion data in-situ using predefined or user-specified sampling rates. Then, the down-sampled data is transferred to a staging area for completion in-transit. A single, serial in-transit node receives all blocks of down-sampled data and generates a look-up table that records the upper and lower bounds of each block to encode their spatial relationship. We use this small look-up table to identify voxel positions during the ray casting process, avoiding expensive visibility sorting or volume reconstruction steps. Figure 2 compares the images resulting from these two visualization algorithms. We note that the two algorithms are not exclusive to each other. Multiple instances of each visualization mode can be dynamically created in-situ and/or in-transit on demand, enabling scientists to explore different aspects of simulation and analysis data in linked-views.

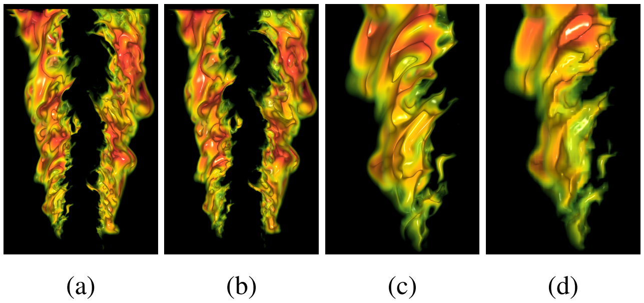

Fig. 2: Visualization of the temperature variable of combustion simulation data. (a) and (c) show an overview and a zoom-in view generated using the in-situ algorithm. (b) and (d) show the same views generated via a hybrid algorithm that down-samples data (at every 8th grid point) in-situ and volume renders the down-sampled data in-transit.

Topology: As discussed in Section II, topological feature extraction techniques have been highly successful in a broad range of applications. In particular, *merge trees*, which encode an ensemble of threshold-based segmentations, have been used extensively in the analysis of large-scale simulations [30], [43]–[45]. The merge tree of a function f encodes the merging of contours – connected component of level sets – as an isovalue is swept top to bottom through the range of f. Each time a new contour appears, at a local maximum, a node is created in the tree. As the isovalue is swept downward, the contour evolves, represented as a lengthening arc in the tree. When two separate contours merge with each other at a saddle in the function, the associated arcs in the tree are merged, as shown in Fig. 3. When combined with topological simplification and filtering, the resulting merge tree encodes a family of segmentations with many analysis uses. For example, the regions around local maxima can be used to describe features such as burning regions [43], extinction events [30],

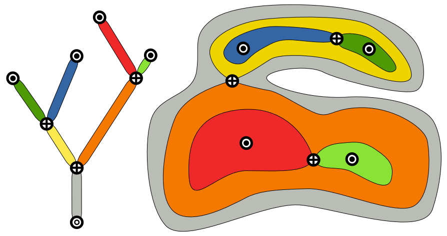

Fig. 3: Merge trees represent merging in the evolution of contours as the isovalue is lowered through the range of a function. The color coding in this 2-dimensional example indicates the correspondence between branches in the tree and their associated region in the domain.

[44], or eddies in ocean currents [46].

In a distributed setting, the computation of a merge tree is inherently not data-parallel due to complex communication costs. However, by adapting and combining two different existing merge tree algorithms, we have created a novel hybrid in-situ/in-transit solution in which we compute subtrees in-situ and then combine the subtrees into the final merge tree using a streaming algorithm in-transit.

To compute the subtrees we adapt a low-overhead, in-core algorithm [32] that works well in-situ but that requires a sort operation, making it ill-suited for a global solution. Special care must be taken to include additional boundary vertices to guarantee that neighboring subtrees can be glued appropriately. A detailed description is beyond the scope of this paper and we refer the reader to [47] for a discussion of the relevant theory. In practice, one must include the boundary components that are the topological equivalent of simulation ghost-cells (these include the 8 corners of the sub-domain and all maxima restricted to boundary components).

The final tree is computed by aggregating subtrees in-transit on a single serial process. We adapt a streaming algorithm for unstructured data [43] that maintains a low memory footprint. Unlike the in-situ algorithm, a global sort is not required, however additional logic must be performed to process subtree vertices in any order. To maintain a low memory footprint, a subtree vertex must be processed before any subtree edge that contains it, and is considered *finalized* once its last incident edge has been processed. As subtree elements are processed the algorithm maintains a merge tree of all elements seen thus far, and writes those vertices and edges to disk that have been finalized, removing them from memory.

Descriptive Statistics: Scientists have long been using descriptive statistics, including first through fourth order moments, to provide concise summaries of trends in their data. In [21]–[23] we describe formulas for robust, single-pass computation of moment-based statistics, and present details on an open source parallel statistics framework that was built using these formulas as part of the VTK library. The statistics algorithms employ a design pattern specifically targeted for distributed-memory parallelism comprising 4 stages, see Figure 4.

The learn stage calculates a primary statistical model from an input data set. Derive calculates a more detailed statistical model from a minimal model. The assess stage annotates each observation with a number of quantities relative to a given statistical model, and the test stage calculates test statistic(s) for hypothesis testing purposes given a model and input data set. From the parallelism standpoint, this partitioning reduces learn to a special case of the map-reduce design pattern [48]. By construction, learn is the only operation which always requires inter-process communication; for instance, in the case of descriptive statistics, cardinality, external values, and centered aggregates up to the fourth order must be exchanged and updated to assemble a global model.

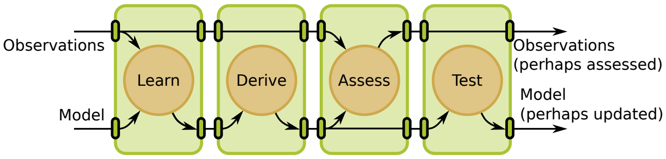

Fig. 4: The 4 operations of statistical analysis and their interactions with input observations and models. The learn stage is the only stage that requires inter-process communication by design.

In this paper we compare two methods for computing descriptive statistics: 1) a fully in-situ approach where learn and derive are performed on shared compute resources; and 2) a hybrid approach where learn is performed in-situ and derive is performed in-transit. In the former, all computations are performed in-situ and an all-to-all communication is required to guarantee a consistent model is computed across all processors. In the latter, all partial models computed on individual processors are communicated to a single serial in-transit process that aggregates the final result.

IV. Hybrid in-situ/in-transit Data Movement

The hybrid in-situ/in-transit framework comprises primary and secondary compute resources. The primary resources execute the main simulation and in-situ computations, while the secondary resources contain a task scheduler and a staging area which is a set of dedicated nodes, whose cores act as "staging buckets" where in-transit operations can be scheduled. In order to minimize impact on the simulation, Fig. 5 shows a messaging scheme that only requests data from the primary resources when secondary resources are available for processing. The task scheduler manages the scheduling and execution of the in-transit computations and is composed of two layers: 1) a communication and data movement layer, and 2) a scheduling and coordination layer. These two layers are built on DART and DataSpaces respectively, which are open-source [49], distributed with ADIOS, and available on Cray machines that have support for Portals or uGNI API. Ports for DART and DataSpaces to IBM BlueGene/P, InfiniBand, and TCP also exist.

Communication and Data Movement Layer: This layer builds on DART [50], which is an open-source asynchronous

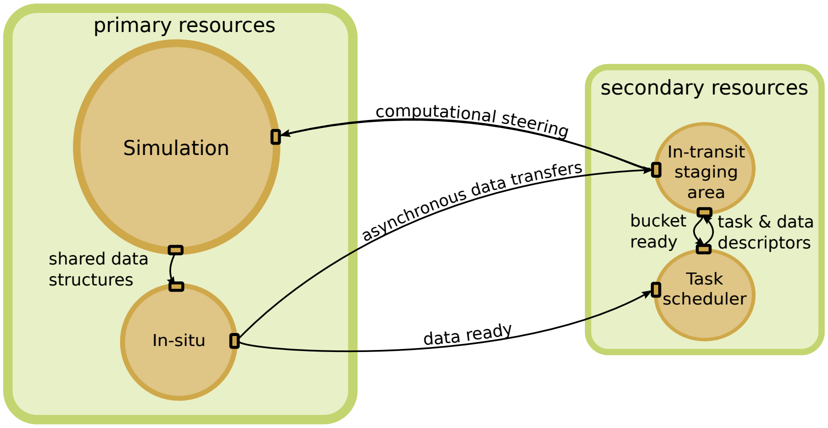

Fig. 5: An overview of the hybrid in-situ/in-transit analysis framework: Primary resources execute the main simulation and in-situ computations. Secondary resources (on the same or on another machine) contain 1) a staging area whose cores act as buckets for in-transit computations; and 2) a task scheduler, built on DART and DataSpaces, that manages the scheduling and execution of the in-transit computations. In-situ computations notify the scheduler when data is ready and an in-transit data and task descriptor is put into a scheduling queue. In-transit staging buckets that have notified the queue that they are available via a bucket ready request are assigned tasks in a first-come first-served manner and asynchronously pull data from the specified in-situ task.

communication and data transport substrate based on RDMA one-sided communication. DART enables asynchronous data extraction from parallel simulation machines, transporting data to the staging area for further processing. It provides services such as node registration/unregistration, data transfer, message passing, event notification and processing. A key contribution of this work is the implementation of the DART functionality on the Gemini network of the Cray's XE/XK systems.

Gemini provides the user Generic Network Interface (uGNI) as its low-level interface. User-space communication in uGNI is supported by a set of data transfer functions using the *Fast Memory Access* (FMA) and the *Block Transfer Engine* (BTE) mechanisms. To ensure efficiency and scalability, DART dynamically adapts which mechanism is used based on data size. For small message sizes, DART uses the GNI *Short Message* (SMSG) mechanism, which leverages FMA and allows for direct OS-bypass achieving the lowest latencies and highest message rates. For large data transfers, the BTE memory operations (RDMA Get and RDMA Put) are used to achieve better computation-communication overlap and lower performance overhead. The completion of an FMA or BTE transaction generates a corresponding event notification at both the source and destination of the data transfer, allowing DART to track the status of a transaction and schedule related data analysis operations.

Scheduling and Coordination Layer: This layer builds on DataSpaces [12], which is a distributed interaction and coordination service. DataSpaces implements a scalable, semantically specialized shared space abstraction that is accessible by all components and services in an application workflow. It can be used to coordinate the execution of these components/services as well as support dynamic and asynchronous interactions among them. It also supports operations such as flexible data querying, filtering, data redistribution, and, building on the asynchronous, low-overhead, memory-to-memory data transport provided by the communication layer, it allows applications to overlap interactions and data transfers with computation, and to reduce the I/O overheads by offloading data operations to the staging area. In this research, we use DataSpaces to enable end-to-end workflows between the multiple interacting in-situ and in-transit processes. This includes the scheduling of in-transit tasks, the coordination of their execution, and the management of data transfers between the in-situ and in-transit processes.

In-transit task scheduling is triggered by two events, *data-ready* and *bucket-ready*. In-situ computations notify DataSpaces of a *data-ready* event by inserting descriptors for RDMA-enabled data blocks containing the intermediate results of in-situ operations. A corresponding in-transit task is also created to process the associated data blocks, and is pushed into the DataSpaces task queue that caches the in-transit tasks and their data descriptors. Staging area buckets notify DataSpaces when they are available for use via a *bucket-ready* event request, and then wait to be assigned tasks by the scheduler. A *free bucket list* within DataSpaces is used to keep track of all currently available staging buckets. Note that in-transit tasks are assigned to staging buckets in a first-come first-served manner. Such an asynchronous pull-based scheduling mechanism can effectively and scalably address the heterogeneity and dynamic nature of the analytics pipeline, and manage load-balancing within the staging area.

V. Results

Simulation Case Study: The hybrid in-situ/in-transit analysis approach is integrated with S3D [51], a massively parallel turbulent combustion code. S3D performs first principlesbased direct numerical simulations of turbulent combustion in which both turbulence and chemical kinetics associated with burning gas-phase hydrocarbon fuels introduce spatial and temporal scales spanning typically at least 5 decades. For production simulations the time steps taken to advance the solution are smaller than the smallest time scales. However, when analyzing the data in a post-processing mode the spatial fidelity is preserved while temporal fidelity is partially compromised since analyses are performed on solution states typically a few hundred time steps apart (see introduction). Two broad classes of turbulent combustion problems are simulated with S3D: statistically stationary and temporally evolving flows. For both of these problem types, capturing intermittent events requires analyses to be performed at a much higher frequency, which is enabled by the hybrid in-situ/in-transit approach.

An example of this is flame stabilization by auto-ignition in a lifted hydrogen jet flame [52]. Ignition kernels form intermittently at the base of a lifted flame and are advected into the oncoming turbulent flow field and the temporal evolution of the balance between chemical kinetic generation, advection, and dissipation results in a stable lifted flame. Deeper insight into the flame stabilization mechanism requires tracking the inception, advection, and dissipation of the ignition kernels vis-a-vis the turbulent strain at a much higher temporal frequency than was hitherto done.

Experimental Results: We have tested our approach on Jaguar, the Cray XK6 at Oak Ridge National Laboratory's National Center for Computational Sciences. The system has 18,688 nodes connected through a Gemini internal interconnect, and each node has a single 16-core AMD 6200 series Opteron processor. The total system memory is 600 terabytes.

In our experimental study, we deployed our framework on a lifted Hydrogen combustion simulation of S3D with a grid domain size of $1600 \times 1372 \times 430$. We tested our framework using two different core counts: 4896 and 9440, with each core representing a portion of the spatial domain of size $100 \times 49 \times 43$ and $50 \times 49 \times 43$, respectively. The core configurations, the data region assignments, and the simulation and I/O times are listed in Table I. For the present experiments data read/write is done on a single-file-per-process basis, which achieves near peak I/O bandwidths over a wide range of core counts. The I/O bandwidths are limited by the number of Object Storage Targets (OSTs) on the lustre filesystem. Since the total data size is constant in the experiments the I/O read/write times do not depend noticeably on the number of cores used.

In our current system, primary and secondary resources are on a shared system and processing resources are managed by the application developers prescribing desired analysis frequencies. All simulation variables are double floating point values (8 bytes) and the in-situ algorithms access simulation variables by sharing the native simulation data structures. While it is possible to write the in-transit data to persistent storage for later processing, there are several advantages to a concurrent approach, including computational steering, on-the-fly visualization, and feature tracking.

	No. of cores	
	4896	9440
No. of simulation/in-situ cores	$16 \times 28 \times 10 = 4480$	$32 \times 28 \times 10 = 8960$
No. of DataSpaces-service cores	160	256
No. of in-transit cores	256	224
Volume size	$1600 \times 1372 \times 430$	$1600 \times 1372 \times 430$
No. of variables	14	14
Data size (GB)	98.5	98.5
Simulation time (sec.)	16.85	8.42
I/O read time (sec.)	6.56	6.56
I/O write time (sec.)	3.28	3.28

TABLE I: This table contains core-allocations, data sizes, and timing information for the two test scenarios: 4896 and 9440 cores. All measurements are per simulation time step.

Our framework covers the entire spectrum, from pure in-situ to pure in-transit analysis. For an entirely data-parallel problem the former will be optimal, while if little or no data-parallelism exists then transferring the data becomes more attractive. The optimal decoupling depends on the inherent scalability of the analysis algorithm in question and the available system resources. In general the in-situ algorithms require a fraction of the simulation data size, and, while we did not have any issues performing the in-situ analyses for our case study, we note that in extreme cases, this small overhead may prevent analysis. In this setting, one potential solution is to shift entirely to in-transit processing, incurring increased data transfer costs.

The scalability of the in-transit stage lies in three aspects. First, the scalability of our scheduling service is enabled by the distributed design of DataSpaces and the hashing used to balance the RPC messages (from in-situ or in-transit nodes) over multiple DataSpaces servers. Second, our scheduling multiplexes different in-transit operations (for each algorithm and each simulation time step), by mapping them to separate in-transit compute nodes, which are independent from each other and operate on different data, thus increasing both the processing parallelism and scalability. Third, in-transit operations pull the data they need directly from the memory of the in-situ nodes using RDMA, the scalability of which is limited only by underlying communication fabrics. Although in-transit computations for a given analysis and timestep are serial, we note that this can easily be made parallel as well.

Figure 6 shows the timing breakdown for in-situ, in-transit, and data movement for the simulation and analytics algorithms for a run of 4,896 cores, where all measurements are for one simulation time step. Among the three analytics tasks, we tested fully in-situ and hybrid in-situ/in-transit variants of both the visualization and descriptive statistics algorithms, while the topological analysis tests were strictly employed with a hybrid in-situ/in-transit algorithm. We can clearly see that the in-situ visualization and the in-situ descriptive statistics only account for a small fraction of the total simulation time. For example, if we perform in-situ visualization at each simulation time step for 4,896 cores, the visualization time is approximately 4.33 percent of the simulation time. Similarly, if we compute in-situ descriptive statistics at each simulation time step, the compute time is approximately 9.73 percent of the simulation time. In

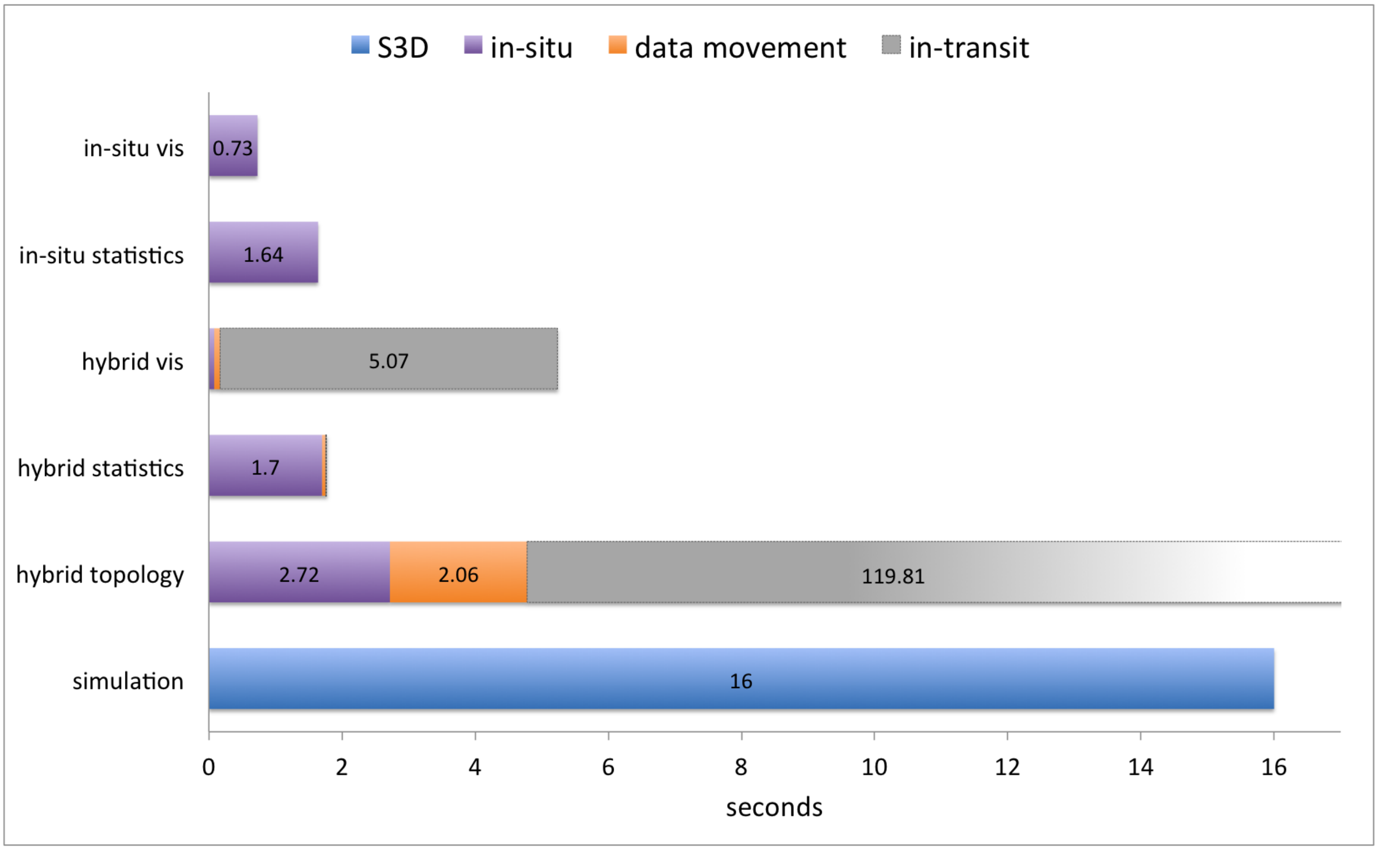

Fig. 6: The timing breakdown for in-situ, in-transit, and data movement for the simulation and the various analytics algorithms using 4896 cores. All measurements are per simulation time step.

practice, we usually perform in-situ processes less frequently (for example, every 10th time step), so the in-situ processing time can be two or three orders of magnitude less than the overall simulation time.

Moreover, we note that the hybrid in-situ/in-transit algorithms can further significantly reduce performance impact to the simulation. For example, recall that the hybrid in-situ/in-transit visualization algorithm first generates down-sampled data in-situ and then leverages DataSpaces to quickly transfer the reduced data for in-transit rendering on secondary compute resources. In the case of 4,896 cores, the down-sampling and data movement time for each time step is only about one percent of the simulation time. Although the time for in-transit rendering increases, we note that this is an asynchronous calculation performed outside of the simulation nodes, and thus has minimal impact on the simulation performance. With the in-situ and in-transit decoupling, we can perform analytics at the same timescale as simulation, while minimizing computation overhead to the simulation. The quantitative timing for different stages of analytics is reported in Table II.

In addition, the use of hybrid in-situ/in-transit algorithms inspires us to re-examine the parallel and serial aspects of analysis algorithms that are critically important but conventionally difficult to parallelize. As shown in Table II, we can see that the hybrid topology algorithm can efficiently compute subtrees directly from the simulation on 4,896 cores. The intermediate data is only about 87 MB, which can be transferred asynchronously and aggregated in-transit to compute the global tree. In this way, scientists are able to deploy algorithms that are not inherently data-parallel during simulation runs, making it possible to capture and track highly intermittent, transient phenomena.

VI. Conclusion and Future work

We have presented a novel hybrid in-situ/in-transit analysis framework that provides flexible data staging and coordination, allowing for transparent data transfers of intermediate data between primary and secondary computing resources. We have introduced a temporally multiplexed approach to decouple the performance of the analysis from that of the simulation, and have reformulated three common analysis algorithms with varying communication patterns into a massively parallel in-situ and a small-scale or serial in-transit stage. Finally, we have performed a case study, in which the analyses were applied to a large-scale turbulent combustion simulation at unprecedented temporal frequencies. Overall, our approach is extensible to a wide range of analyses, and promises to significantly improve the time to insight for modern scientific simulations.

	in-situ time (sec.)	data movement time (sec.)	data movement size (MB)	in-transit time (sec.)
in-situ visualization	0.73	–	–	–
in-situ descriptive statistics	1.64	–	–	–
hybrid in-situ/in-transit visualization	0.08 (down-sample)	0.092	49.19	5.06 (render)
hybrid in-situ/in-transit topology	2.72 (compute subtree)	2.06	87.02	119.81 (compute global tree)
hybrid in-situ/in-transit descriptive statistics	1.69 (learn)	0.06	13.30	0.01 (derive)

TABLE II: The timing and data movement costs for the various in-situ and hybrid in-situ/in-transit analytics algorithms using 4896 cores. All measurements are for a single simulation time step.

Nevertheless, a number of challenges and opportunities remain. First, the performance of the analysis algorithms can be highly data-dependent and it is likely that different in-situ processes finish at significantly different times. While our current system collects and buffers the in-transit data prior to processing, a more optimal approach would be to process in-transit data in a streaming fashion, starting as soon as the first data arrives. This has the potential to hide much of the in-transit computational costs and improve overall system utilization. Furthermore, the data transfer patterns exhibited by the analysis algorithms are significantly different from those common to check-pointing or other traditional file I/O operations. This presents a new challenge to optimize the data movement layer and extend it to the unstructured, data dependent, and highly adaptive output of data analysis tools. We have plans to use the current system as a test bed to experiment trade-offs between in-situ, in-transit, and post-processing algorithms. For example, we plan to develop a hybrid in-situ/in-transit auto-correlative statistical technique, in addition to combining the merge tree computation presented in this work with statistical analyses to enable the computation of feature-based statistics such as those present in the corresponding post-processing tools [30], [43]. Finally, to address more complex application scenarios, we aim to introduce alternative staging techniques that utilize a separate process co-hosted on the application node that executes asynchronously with the application.

Acknowledgment

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

This work was performed under the auspices of the US Department of Energy (DOE) by the Lawrence Livermore National Laboratory under Contract Nos. DE-AC52- 07NA27344, LLNL-JRNL-412904L.

References

[1] T. Tu, H. Yu, L. Ramirez-Guzmanz, J. Bielak, O. Ghattas, K.-L. Ma, and D. R. O'Hallaron, "From Mesh Generation to Scientific Visualization: An End-to-End Approach to Parallel Supercomputing," in *Proceedings of ACM/IEEE Supercomputing Conference*, 2006.

[2] H. Yu, T. Tu, J. Bielak, O. Ghattas, J. C. López, K.-L. Ma, D. R. O'Hallaron, L. Ramirez-Guzmanz, N. Stone, R. Taborda-Rios, and J. Urbanic, "Remote Runtime Steering of Integrated Terascale Simulation and Visualization," in *ACM/IEEE Supercomputing Conference HPC Analytics Challenge*, 2006.

[3] H. Yu, C. Wang, R. Grout, J. Chen, and K.-L. Ma, "In Situ Visualization for Large-Scale Combustion Simulations," *IEEE Computer Graphics and Applications*, vol. 30, no. 3, pp. 45–57, 2010.

[4] J.-M. F. Brad Whitlock and J. S. Meredith, "Parallel In Situ Coupling of Simulation with a Fully Featured Visualization System," in *Proc. of 11th Eurographics Symposium on Parallel Graphics and Visualization (EGPGV'11)*, April 2011.

[5] N. Fabian, K. Moreland, D. Thompson, A. Bauer, P. Marion, B. Gevecik, M. Rasquin, and K. Jansen, "The paraview coprocessing library: A scalable, general purpose in situ visualization library," in *Proc. of IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, October 2011, pp. 89 –96.

[6] S. Lakshminarasimhan, J. Jenkins, I. Arkatkar, Z. Gong, H. Kolla, S.-H. Ku, S. Ethier, J. Chen, C. Chang, S. Klasky, R. Latham, R. Ross, and N. Samatova, "Isabela-qa: Query-driven analytics with isabela-compressed extreme-scale scientific data," in *Proc. of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, November 2011, pp. 1 –11.

[7] M. Li, S. S. Vazhkudai, A. R. Butt, F. Meng, X. Ma, Y. Kim, C. Engelmann, and G. Shipman, "Functional partitioning to optimize end-to-end performance on many-core architectures," in *Proc. of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, November 2010, pp. 1–12.

[8] F. Zhang, C. Docan, M. Parashar, S. Klasky, N. Podhorszki, and H. Abbasi, "Enabling in-situ execution of coupled scientific workflow on multi-core platform," in *Proc. 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS'12)*, 2012.

[9] H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, and F. Zheng, "Datastager: scalable data staging services for petascale applications," in *Proc. of 18th International Symposium on High Performance Distributed Computing (HPDC'09)*, 2009.

[10] F. Zheng, H. Abbasi, C. Docan, J. Lofstead, S. Klasky, Q. Liu, M. Parashar, N. Podhorszki, K. Schwan, and M. Wolf, "PreDatA - preparatory data analytics on peta-scale machines," in *Proc. of 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS'10)*, April 2010.

[11] H. Abbasi, G. Eisenhauer, M. Wolf, K. Schwan, and S. Klasky, "Just In Time: Adding Value to The IO Pipelines of High Performance Applications with JITStaging," in *Proc. of 20th International Symposium on High Performance Distributed Computing (HPDC'11)*, June 2011.

[12] C. Docan, M. Parashar, and S. Klasky, "DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows," in *Proc. of 19th International Symposium on High Performance and Distributed Computing (HPDC'10)*, June 2010.

[13] C. Docan, M. Parashar, J. Cummings, and S. Klasky, "Moving the Code to the Data - Dynamic Code Deployment Using ActiveSpaces," in *Proc. of 25th IEEE International Parallel and Distributed Processing Symposium (IPDPS'11)*, May 2011.

[14] V. Vishwanath, M. Hereld, and M. Papka, "Toward simulation-time data analysis and i/o acceleration on leadership-class systems," in *Proc. of IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, October 2011.

[15] A. Globus, "A Software Model for Visualization of Time Dependent 3-D Computational Fluid Dynamics Results," NAS Applied Research, NASA Ames Research Center, Tech. Rep. RNR 92-031, 1992.

[16] K.-L. Ma, "Runtime Volume Visualization of Parallel CFD," in *Proceedings of Parallel CFD Conference*, 1995, pp. 307–314.

[17] J. Rowlan, E. Lent, N. Gokhale, and S. Bradshaw, "A Distributed, Parallel, Interactive Volume Rendering Package," in *Proceedings of IEEE Visualization Conference*, 1994, pp. 21–30.

[18] S. G. Parker and C. R. Johnson, "SCIRun: A Scientific Programming

Environment for Computational Steering,” in *Proceedings of ACM/IEEE Supercomputing Conference*, 1995.

[19] “The R project for statistical computing,” http://www.r-project.org/.

[20] M. Schmidberger, M. Morgan, D. Eddelbuettel, H. Yu, L. Tierney, and U. Mansmann, “State-of-the-art in parallel computing with R,” Department of Statistics, University of Munich, Tech. Rep. 47, 2009.

[21] J. Bennett, P. Pébay, D. Roe, and D. Thompson, “Numerically stable, single-pass, parallel statistics algorithms,” in *Proc. 2009 IEEE International Conference on Cluster Computing*, New Orleans, LA, Aug. 2009.

[22] P. P. Pébay, D. C. Thompson, and J. Bennett, “Computing contingency statistics in parallel: Design trade-offs and limiting cases,” in *CLUSTER*. IEEE, 2010, pp. 156–165. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5599992

[23] P. P. Pébay, D. C. Thompson, J. Bennett, and A. Mascarenhas, “Design and performance of a scalable, parallel statistics toolkit,” in *IPDPS Workshops*. IEEE, 2011, pp. 1475–1484. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6008655

[24] “VTK Doxygen documentation,” http://www.vtk.org/doc/nightly/html.

[25] A. Mascarenhas, R. W. Grout, P.-T. Bremer, E. R. Hawkes, V. Pascucci, and J. H. Chen, “Topological feature extraction for comparison of terascale combustion simulation data,” in *Topological Methods in Data Analysis and Visualization*, ser. Mathematics and Visualization, V. Pascucci, X. Tricoche, H. Hagen, and J. Tierny, Eds. Springer Berlin Heidelberg, 2011, pp. 229–240.

[26] A. Mascarenhas and J. Snoeyink, “Isocontour based visualization of time-varying scalar fields,” in *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration*, ser. Mathematics and Visualization. Springer Berlin Heidelberg, 2009, pp. 41–68.

[27] P.-T. Bremer, G. H. Weber, V. Pascucci, M. S. Day, and J. B. Bell, “Analyzing and tracking burning structures in lean premixed hydrogen flames.” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 2, pp. 248–260, 2010.

[28] D. Laney, P. T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci, “Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1053–1060, Sep. 2006.

[29] A. Gyulassy, M. Duchaineau, V. Natarajan, V. Pascucci, E. Bringa, A. Higginbotham, and B. Hamann, “Topologically clean distance fields,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1432–1439, 2007.

[30] J. Bennett, V. Krishnamoorthy, S. Liu, R. Grout, E. R. Hawkes, J. H. Chen, J. Shepherd, V. Pascucci, and P.-T. Bremer, “Feature-based statistical analysis of combustion simulation data,” *IEEE Trans. Vis. Comp. Graph.*, vol. 17, no. 12, pp. 1822–1831, 2011.

[31] G. Reeb, “Sur les points singuliers d’une forme de pfaff completement intergrable ou d’une fonction numerique [on the singular points of a complete integral pfaff form or of a numerical function],” *Comptes Rendus Acad.Science Paris*, vol. 222, pp. 847–849, 1946.

[32] H. Carr, J. Snoeyink, and U. Axen, “Computing contour trees in all dimensions,” in *Proc. of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM. New York, NY, USA: ACM Press, Jan. 2000, pp. 918–926.

[33] H. Edelsbrunner, J. Harer, and A. Zomorodian, “Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds,” *Discrete Computational Geometry*, vol. 30, pp. 173–192, 2003.

[34] P.-T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci, “Topological hierarchy for functions on triangulated surfaces,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, pp. 385–396, 2004.

[35] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann, “A topological approach to simplification of three-dimensional scalar functions,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 474–484, 2006.

[36] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas, “Robust on-line computation of Reeb graphs: simplicity and speed,” *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007.

[37] P.-T. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. S. Day, and J. B. Bell, “Interactive exploration and analysis of large-scale simulations using topology-based data segmentation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, pp. 1307–1324, 2011.

[38] V. Pascucci and K. Cole-McLaughlin, “Parallel computation of the topology of level sets,” *Algorithmica*, vol. 38, pp. 249–268, 2003.

[39] A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci, “A practical approach to Morse-Smale complex computation: scalability and generality,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1619–1626, 2008.

[40] A. Gyulassy, T. Peterka, R. Ross, and V. Pascucci, “The parallel computation of Morse-Smale complexes,” *IEEE International Parallel and Distributed Processing Symposium, to appear*, 2012.

[41] P.-T. Bremer, G. Weber, V. Pascucci, M. Day, and J. Bell, “Analyzing and tracking burning structures in lean premixed hydrogen flames,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 2, pp. 248–260, 2010.

[42] V. Pascucci, G. Scorzelli, P.-T. Bremer, and A. Mascarenhas, “Robust on-line computation of Reeb graphs: Simplicity and speed,” *ACM Transactions on Graphics*, vol. 26, no. 3, pp. 58.1–58.9, 2007, proceedings of SIGGRAPH 2007.

[43] P.-T. Bremer, G. Weber, J. Tierny, V. Pascucci, M. Day, and J. B. Bell, “Interactive exploration and analysis of large scale simulations using topology-based data segmentation,” *IEEE Trans. on Visualization and Computer Graphics*, vol. 17, no. 99, 2010.

[44] A. Mascarenhas, R. W. Grout, P.-T. Bremer, E. R. Hawkes, V. Pascucci, and J. Chen, *Topological feature extraction for comparison of terascale combustion simulation data*, ser. Mathematics and Visualization. Springer, 2011, pp. 229–240.

[45] P.-T. Bremer, E. Brings, M. Duchaineau, A. Gyulassy, D. Laney, A. Mascarenhas, and V. Pascucci, “Topological feature extraction and tracking,” *Proceedings of SciDAC 2007 - Scientific Discovery Through Advanced Computing*, vol. 78, pp. 012 032 (5pp), Journal of Physics Conference Series, 2007.

[46] S. Williams, M. Petersen, P.-T. Bremer, M. Hecht, V. Pascucci, J. Ahrens, M. Hlawitschka, and B. Hamann, “Adaptive extraction and quantification of atmospheric and oceanic vortices,” *IEEE Trans. Vis. Comp. Graph.*, vol. 17, no. 12, pp. 2088–2095, 2011.

[47] V. Pascucci and K. Cole-McLaughlin, “Parallel computation of the topology of level sets,” *Algorithmica*, vol. 38, no. 1, pp. 249–268, Oct. 2003.

[48] J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” in *OSDI’04: Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, Dec. 2004.

[49] “Dataspaces project,” http://www.dataspaces.org/.

[50] C. Docan, M. Parashar, and S. Klasky, “Dart: a substrate for high speed asynchronous data io,” in *Proc. of 17th International Symposium on High Performance Distributed Computing (HPDC’08)*, 2008.

[51] 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. Podhorski, R. Sankaran, S. Shende, and C. S. Yoo, “Terascale direct numerical simulations of turbulent combustion using s3d,” *Computational Science and Discovery*, vol. 2, pp. 1–31, 2009.

[52] C. S. Yoo, R. Sankaran, and J. H. Chen, “Three-dimensional direct numerical simulation of a turbulent lifted hydrogen jet flame in heated coflow: Flame stabilization and structure,” *Journal of Fluid Mechanics*, vol. 640, pp. 453–481, 2009.