In-situ Feature-based Objects Tracking for Large-Scale Scientific Simulations

Fan Zhang, Solomon Lasluisa, Tong Jin, Ivan Rodero, Hoang Bui, Manish Parashar
NSF Center for Cloud and Autonomic Computing
Rutgers University, Piscataway NJ, USA
{zhangfan, lasluisa, tjin, irodero, hbui, parashar}@cac.rutgers.edu

***Abstract*—Emerging scientific simulations on leadership class systems are generating huge amounts of data. However, the increasing gap between computation and disk IO speeds makes traditional data analytics pipelines based on post-processing cost prohibitive and often infeasible. In this paper, we investigate an alternate approach that aims to bring the analytics closer to the data using data staging and the in-situ execution of data analysis operations. Specifically, we present the design, implementation and evaluation of a framework that can support in-situ feature-based object tracking on distributed scientific datasets. Central to this framework is the scalable decentralized and online clustering (DOC) and cluster tracking algorithm, which executes in-situ (on different cores) and in parallel with the simulation processes, and retrieves data from the simulations directly via on-chip shared memory. The results from our experimental evaluation demonstrate that the in-situ approach significantly reduces the cost of data movement, that the presented framework can support scalable feature-based object tracking, and that it can be effectively used for in-situ analytics for large scale simulations.**

***Keywords*-Scientific data analysis, scalable in-situ data analytics, feature-based object tracking**

I. Introduction

Scientific simulations running at extreme scale on leadership class systems are generating unprecedented amount of data. To enable scientific discovery, the high amount of simulation data has to be analyzed and understood by domain scientists. However, the increasing gap between computation and disk IO speeds makes traditional data analytics pipelines based on post-processing cost prohibitive and often infeasible [1]. Storing entire datasets from the large scale systems running the simulation to storage servers is becoming increasingly expensive in terms of the time required as well as the energy costs associated with data movement. Moreover, the efficiency and scalability of subsequent analysis operations on the data are also hindered by the cost of disks-based data I/O. This trend of big simulation data is resulting significant challenges limiting the ability of scientists to translate this data into insights, and as a result, the impact of the simulations themselves. Clearly, this required rethinking the analytics pipelines to incorporate new approaches that are cost-effective and scalable. In-situ data processing has recently emerged as a promising approach which can effectively reduce data movement and data IO overheads by placing analysis operations at the simulation machines closer to where the data is being produced.

In this paper, we investigate this alternate approach that aims to bring the analytics closer to the data using the in-situ execution of data analysis operations. Specifically we explore in-situ feature-based objects tracking in distributed scientific datasets. In order to extract insightful information from the large datasets produced by simulations over thousands of time steps, scientists often need to follow data objects of interest (i.e., features) across the different time steps, such as tracking storm formation and movement in climate modeling simulation, or identify burning regions in combustion simulations. As a result, feature extraction and tracking is an important technique for analyzing and visualizing scientific datasets. However, most feature extraction and tracking techniques operate offline by post-processing data dumped from the simulation runs. Being able to perform such feature-based analytics in-situ, i.e., concurrent with a simulation run, can significantly increase the utility of these techniques and the productivity of the simulations, and can also lead to better utilization of expensive high-end resources.

However, performing in-situ feature tracking presents several research challenges. First, it requires a distributed feature extraction and tracking algorithm that operates on distributed data. Second, it requires a programming and runtime system that enables the mapping and execution of the simulation and the data analytics codes on co-located processor cores and asynchronously share data at runtime. Most existing in-situ data analysis implementations employ an inline approach, i.e., the data analysis operations are embedded into the execution path of main simulation process, for example, as function calls. One major drawback of this approach is that simulation would have to block and wait for the completion of the in-situ analytics routine, which impacts the execution and performance of the main simulations.

In this paper we present the design, implementation and evaluations of a systems framework that can support in-situ feature-based objects tracking for large-scale parallel simulations. Central to this framework is the scalable decentralized and online clustering (DOC) [2] and cluster tracking algorithm, which executes in-situ, i.e., on different cores, and in parallel with the simulation processes, and retrieves data from the simulations directly and asynchronously via on-chip shared memory. The framework also provides programming support for composing in-situ "simulation plus analytics" workflows. The results from experimental evaluation on the Lonestar system at Texas Advanced Computing Center (TACC) demonstrate that the in-situ approach significantly reduces the cost

of data movement, that the presented framework can support scalable feature-based cluster tracking, and that it can be effectively used for in-situ analytics for large scale simulations.

The rest of the paper is structured as follows. Section II presents related work. Section III describes the system architecture and implementation of the feature-based tracking algorithm. Section IV presents an experimental evaluation of the prototype system using 3D time-varying CFD dataset. Section V concludes the paper.

II. Related Work

In-situ scientific data processing: The increasing performance gap between computing and I/O, and the cost of moving large volume of data to/from disks, motivates computation scientists to employ the in-situ data processing approach to perform analysis, visualization [3], indexing building [4], compression etc. The key idea is to move operations to data where the simulation is running. However, existing inline approach [5], [6] tightly integrate analysis or visualization libraries into simulation code. Our techniques provides a more a generic framework to compose and execute in-situ data operations in a flexible and customizable way.

Staging area based in-transit data analysis and IO: The data staging area, a set of additional compute nodes allocated by users when launching the parallel simulations, and the application of staging area has been investigated to add values to simulation's I/O pipeline in projects DataStager [7], PreDatA [8], JITStaging [9], ActiveSpace [10], Glean [11]. Our techniques focuses on scheduling and running analysis code in-situ to exploit increasing hardware parallelism and intra-node locality, and can be integrated with these in-transit approaches to perform hybrid data staging and analysis.

Features tracking of time-varying simulation datasets: Many techniques have been developed by computer vision community to extract, identify and track features. Silver et al. [12] presented a semi-automatic volume tracking algorithm to improve visualization of 3D time-varying CFD datasets. Chen et al. [13] developed a parallel algorithm to analyze and visualize in realtime the evolving features extracted from time-varying simulation datasets. Our technique represents features as clustered data points in a multi-dimensional information space, and identifies and tracks the data clusters of interest in a distributed and timely manner using DOC.

III. Design and Implementation

A. Overview of System Framework

The system architecture of the proposed distributed data analysis framework consists of three main components, the execution client, in-situ data staging daemon and the management server. Each execution client abstracts and represents a basic computation element such as a physical processor core. The management server acts as the rendezvous point to bootstrap execution clients, and manages the execution of various applications within the in-stiu data analysis workflow.

Figure 1 shows co-located execution of DOC workers with the simulation program, where the physical processor cores on the multi-core compute nodes are functionally partitioned. As shown in the figure, on-node computation resource is mostly used by the scientific simulation program, and two processor cores are used to execute in-situ data staging daemon and DOC worker. The in-situ data staging daemons across the distributed cluster nodes build a Co-located DataSpaces (CoDS) [14], which provides a virtual shared-space abstraction to support asynchronous and decoupled coordination and data sharing between simulation processes and the DOC worker. The capability of functional partitioning of node-level processor cores, and the programming model and runtime to support asynchronous communication between the intra-node cores, is critical to exploit the ever-increasing hardware parallelism on emerging supercomputers.

Our framework employs the data-centric scheduling approach to map processes from different applications onto physical processor cores so that a large portion of the inter-application data transfers can be performed using the on-node shared memory. More specifically, two mapping methods - both centralized server side and decentralized execution client side - are designed in the framework. The server side mapping is applied to a "bundle" of concurrently coupled component applications that are launched simultaneously, have regular inter-application communication patterns and do not need dynamic re-mapping after launch, which matches with the "simulation plus DOC" scenario described in this paper. The execution client side approach schedules the newly launched workflow application that have dependency on the distributed data generated by preceding application. More details about the data-centric mapping is discussed in [14].

B. Feature-based Objects Tracking through DOC

1) Decentralized Online Clustering (DOC) and Feature Extraction: One method of identifying objects in scientific simulations is through data point clustering. Clusters can represent subsets of interest in a dataset, for example a flame or region of chemical reaction in a combustion simulation.

The nature of data creation in scientific applications, as well as other applications, require specialized clustering algorithms.In this paper we use Decentralized Online Clustering (DOC) which was created to provide online and decentralized data analysis using the collective computing resources in

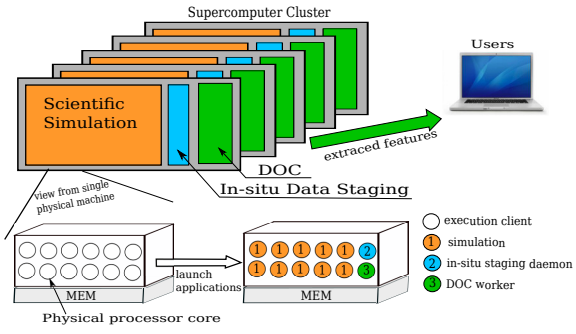


Fig. 1. Architecture of the in-situ feature extraction and tracking system.

distributed systems. DOC's decentralized programming can reduce overall data transfer cost by its novel clustering workflow, where individual nodes are assigned a specified region of space to analyze. A complete explanation of DOC is done by Quiroz [15]. To summarize DOC workers run on each computational node and identify local clusters These workers then communicate with neighboring workers and merge clusters iteratively. DOC can maintain data locality in certain configuration and thereby remove the need to aggregate data.

As part of our work DOC has been modified to extract cluster feature that can later be used to uniquely identify objects across time. Identified clusters in a dataset contain metadata or cluster features which can be extracted in order to describe the cluster. These features describe the data points which form the cluster (e.g., location, density, etc.). Cluster feature are collected within DOC itself leveraging the information used to cluster the data. As will be explained in section III-B2 cluster features can be used to uniquely identify a cluster.

2) Cluster Tracking Using Recursive Clustering: When a collection of cluster features have been gathered over time, the clusters they describe can be tracked by comparing the features to one another. By using a clustering algorithm as the mechanism to compare cluster features to another another it is possible to reuse DOC and create a two step clustering process to identify objects and tracks them. This is possible if cluster features are sufficiently similar from one point in time to the next that they can be assumed to correspond to the same cluster. As an example Figure 2 is provided where three clusters at different points in time have their centroids collected and subsequently had their centroids clustered through DOC. As can be seen the three centroids cluster with one another and conclude that based on this information the three clusters are in fact the same cluster at different times. As explained by Lasluisa [16], clusters can be identified and tracked by their features through a comparison of one cluster's features to another cluster's feature at different points in time.

C. Composing in-situ data analysis workflow

One important idea of the proposed framework is to support building in-situ data analysis as a tightly-coupled workflow. Two problems need to be solved regarding the programmability: First, how to specify the control flow; Second, how to program the coordination and data communication between the interacting workflow applications.

The tightly-coupled application workflow is expressed as a DAG, where each vertex in the DAG represents a parallel program. Our DAG representation extends traditional DAG representation such as DAGMan used in the workflow engine Pegasus, with the concept of a "bundle" which represents a group of parallel programs that need to be launched simultaneously, for example, the simulation and DOC programs of our in-situ feature extraction and tracking scenario. The edges of the DAG represent the control flow. The DAG as well as the bundles are explicitly defined by users. Figure 3 presents the DAG representation for our simulation plus DOC application workflow, and the user-generated DAG description file which is then parsed by the management server. Each parallel program in the DAG is identified by a unique application id in the description file.

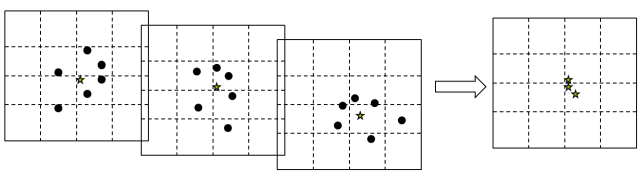


Fig. 2. Example of how cluster centroids (a cluster feature) can be used to track a cluster's movement

Components in the application workflow coordinate and communicate with each other through the abstraction of a shared data space - CoDS. In our in-situ feature extraction and tracking scenario, the DOC workers (data consumer) need to continuously access scientific data computed by the simulation (data producer). To implement this, each process of the simulation program specifies a descriptor as the key for its data, and inserts the data into CoDS using collective *put()* interface. Each DOC worker generates the query key, and uses the *get()* interface to retrieve data of interest. More specifically, DOC workers use the interface *get_local()* to only retrieve simulation data produced on local machine. One assumption for our framework is that the interacting workflow applications share the knowledge about the data, such as type of key, data name and format. Figure 4 shows a logical view of data interaction through shared data space.

IV. Experimental Evaluation

The prototype implementation of our framework was evaluated on the Lonestar linux cluster at Texas Advanced Computing Center (TACC). The Lonestar has 1,888 compute nodes, and each compute node contains two hex-core Intel Xeon processors, 24GB of memory and a QDR InfiniBand switch fabric that interconnects the nodes through a fat-tree topology. The system also supports a 1PB Lustre parallel file system.

A. Performance of Data Transfer

This section evaluates the end-to-end data transfer performance, and more specifically the time used to transfer data from simulation processes to DOC workers, for both our in-situ memory-to-memory and the disk IO approaches. In this

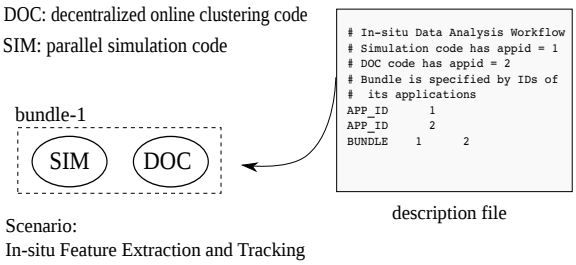


Fig. 3. Example of workflow DAG representation and description file.

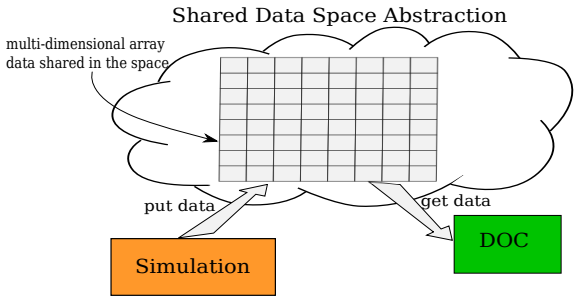


Fig. 4. Simulation and DOC workers share data through data space abstraction.

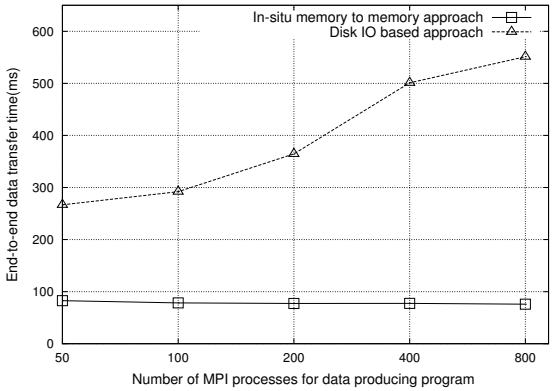


Fig. 5. End-to-end data transfer time in millisecond. The size of data produced per simulation process at each timestep is 16MB.

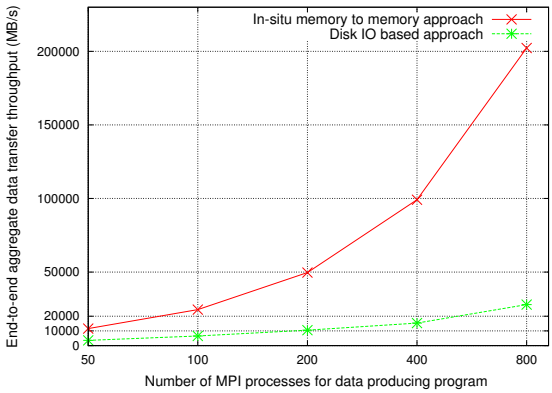


Fig. 6. Aggregate data transfer throughput. The size of data produced per simulation process at each timestep is 16MB.

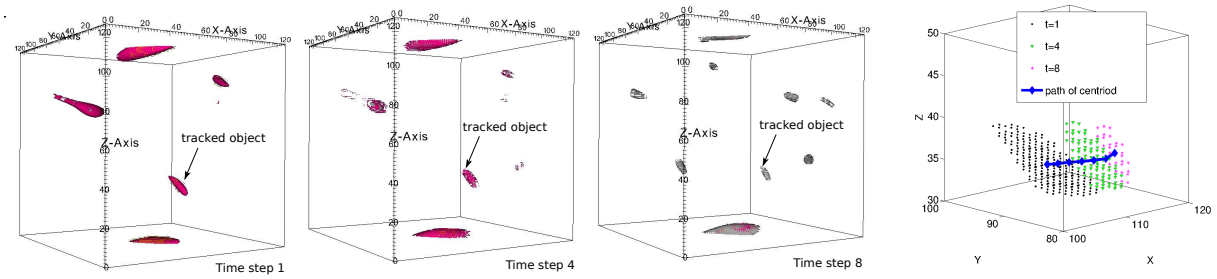
case, we use a testing MPI program as the parallel data producing simulation, which runs on a set of *m* processor cores. The parallel DOC workers runs on a separate set of *n* processor cores where the ratio of *m:n* is 10. In our in-situ data analysis approach, each DOC worker runs on a processor core co-located with 10 simulation cores of the same compute node, and retrieves data generated by the 10 intra-node simulation processes through CoDS *get_local()* interface. In the disk IO approach, simulation processes dump data to disk with the one file per process method using binary POSIX IO operations. Data files are then read by parallel DOC workers. For this evaluation, the number of simulation processes *m* is varied from 50 to 800, and the size of data produced per simulation process at each timestep is varied from 1MB to 64MB. The testing program is configured to run for 100 timesteps at each data output size.

Figure 5 and 6 compares the performance of the two evaluated end-to-end data transfer approaches. As shown in Figure 5, our in-situ memory-to-memory method is much faster than the disk IO approach, with average speedup of transfer performance as about 5. Also, the in-situ memory-to-memory method is scalable, and shows no performance degradation when the number of MPI processes in data producing program increases from 50 to 800. The reason accounts for this significant performance gain is that all data movement is intra-node, and performed through the on-node fast IO path - shared memory. But for the disk IO approach, both data producer and DOC worker processes have to use the off-node slow path - disk. Figure 6 illustrates the performance gain from another dimension - aggregate data transfer throughput. The fast intra-node shared memory approach enables much higher aggregate bandwidth for the data movement between simulation and DOC.

B. Effectiveness of the Feature-based Cluster Tracking Algorithm

This section evaluates the effectiveness and accuracy of our proposed feature tracking algorithm, using time-varying 3D dataset. The dataset is generated by simulation of coherent turbulent vortex structures with 128^3 resolution (vorticity magnitude) and 100 time steps. In this case, the data cluster or object of interest is defined as thresholded connected voxel regions. These regions evolve both in location and shape during the simulation. Though different time steps of the dataset can be visualized offline using visualization tool such as Visit, it is difficult to visually observe and accurately follow regions of interest. The tracking information from our algorithm is used to determine how the regions evolves, e.g., size, location, density, over the time steps.

In this experiment, we define the volume regions of interest as data points clusters that with vorticity values in the range of 9 to maximum. Figure 7(a) shows three selected time steps of the visualized dataset, and we demonstrate the effective tracking of the evolving volume region (or object as in DOC) pointed by black arrows. For this experiment we define *Tracking accuracy* as the ratio of data points encompassed by the tracked objects to total number of points in the experiment. This insure that high accuracy can only be achieved by identifying paths which pass through In each experiment to test accuracy 50 frames were used to identify the paths of the objects within these 50 frames. The tracking accuracy average across 47 tests was 92.28%, meaning only 7.72% of all vortex points were not associated to any trackable object in our experiments. In Figure 7(b) we present the tracking of a object (the same one as in visualized Figure 7(a)), as seen by DOC, at 3 different time steps. As can be seen this object is moving from left to right and shrinking in size.



(a) A visualized view of the evolving volume regions (objects) tracked by our feature-based tracking algorithm.

(b) Illustration of tracked path for object evolves over multiple time steps.

Fig. 7. Experimental results

V. Conclusion and Future Work

This paper explored the in-situ execution of feature-based objects tracking of time-varying scientific simulation data. Specifically, we presented a feature-based cluster tracking algorithm which builds on DOC and uses it to process data in a distributed and timely manner. We also presented Co-located DataSpaces, a framework and its programming interface, to compose and run tightly coupled workflow applications in-situ. We evaluated the proposed approach on the Lonestar cluster at TACC using a number of experiments. The experiments measured the end-to-end data transfer performance as well as the effectiveness and accuracy of our cluster tracking algorithm.

Our direction for future work includes extending the decentralized in-situ cluster tracking system for other application areas such as online monitoring of resource utilization and anomaly detection in large scale data center. We will also explore the use of on-node NVRAM/SSD storage to support energy-efficient in-situ staging of large data sets which could not simply stored in current on-node memory.

Acknowledgement

The research presented in this work is supported in part by National Science Foundation (NSF) via grants number IIP 0758566 , by the Department of Energy ExaCT Combustion Co-Design Center via subcontract number 4000110839 from UT Battelle, by the Scalable Data Management, Analysis, and Visualization (SDAV) Institute via the grant numbers DE-SC0007455, the CPES Fusion Simulation Project via grant number DE-FG02-06ER54857, and by an IBM Faculty Award. The research and was conducted as part of the NSF Cloud and Autonomic Computing (CAC) Center at Rutgers University.

References

[1] H. Childs, "Architectural Challenges and Solutions for Petascale Post-processing," *Journal of Physics: Conf. Series*, vol. 78, no. 1, p. 12, 2007.

[2] A. Quiroz, N. Gnanasambandam, M. Parashar, and N. Sharma, "Robust clustering analysis for the management of self-monitoring distributed systems," *Cluster Computing*, vol. 12, no. 1, pp. 73–85, Mar. 2009.

[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. Kim, H. Abbasi, L. Chacon, C. Docan, S. Klasky, Q. Liu, N. Podhorszki, A. Shoshani, and K. Wu, "Parallel in situ indexing for data-intensive computing," in *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symp. on*, oct. 2011, pp. 65 –72.

[5] 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 Symp. on Parallel Graphics and Visualization (EGPGV'11)*, April 2011.

[6] 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 *Large Data Analysis and Visualization (LDAV), IEEE Symp. on*, oct. 2011, pp. 89 –96.

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

[8] 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 Intl. Parallel and Distributed Processing Symp. (IPDPS'10)*, April 2010.

[9] 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. 20th Intl. Symp. on High Performance Distributed Computing (HPDC'11)*, June 2011.

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

[11] V. Vishwanath, M. Hereld, and M. Papka, "Toward simulation-time data analysis and i/o acceleration on leadership-class systems," in *Large Data Analysis and Visualization (LDAV), IEEE Symp. on*, oct. 2011, pp. 9 –14.

[12] D. Silver and X. Wang, "Tracking and visualizing turbulent 3d features," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 129–141, Apr. 1997.

[13] J. Chen, D. Silver, and M. Parashar, "Real-time feature extraction and tracking in a computational steering environment," in *Proc. of Advanced Simulations Technologies Conf. (ASTC'03)*, 2003.

[14] 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 Intl. Parallel and Distributed Processing Symp. (IPDPS'12)*, 2012.

[15] A. Q. Hernandez, "Decentralized online clustering for supporting autonomic management of distributed systems," PhD in Electrical and Computer Engineering, Rutgers University, 2010.

[16] S. Lasluisa, "A cluster tracking algorithm for distributed data analytics," M.S. in Electrical and Computer Engineering, Rutgers University, 2012.