Accelerating In-situ Feature Extraction of Large-Scale Combustion simulation with Subsampling

Sidharth Kumar, Steve Petruzza, Duong Hoang, Valerio Pascucci
Scientific Computing and Imaging Institute, University of Utah
{sidharth,spetruzza,hoang,pascucci}@sci.utah.edu

Abstract

The increasing gap between available compute power and I/O capabilities is driving more users to adopt in-situ processing capabilities. However, making the transition from *post-processing* to *in-situ* mode is challenging, as parallel analysis algorithms do not always scale well with the actual simulation code. This is observed because these algorithms either involve global reduction operations with significant communication overheads or have to perform an overwhelmingly large amount of computation. In this work we propose a software stack, designed specially to expedite parallel analysis algorithms, thus making it possible for them to work in in-situ mode. Our software stack comprises of data-reduction techniques, sub-sampling and compression, used to reduce the computation load of the analysis algorithm. For better approximation of data, we can also use wavelets instead of sub sampling. We demonstrate the efficacy of our pipeline using two analysis algorithms, parallel merge tree and isosurface rendering. We also study the trade-off between the I/O gains and the corresponding error induced by the data reduction techniques. We present results with KARFS simulation framework run on the Tesla cluster of the Scientific computing and Imaging institute at the University of Utah.

1 Software stack for In-situ analysis

Besides being scalable, the time to completion for any in-situ analysis algorithm should only be a small fraction of the actual simulation time. This hard constraint makes it difficult for parallel analysis algorithms to transcend from post-processing to in-situ mode. With out software stack (see Figure 1), we hope that any parallel analysis algorithm can make this transition. Depending on the needs, algorithms can select which component of the software stack to use.

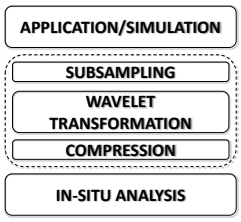


Figure 1. Software stack for in-situ analysis.

1.1 Subsampling

We use Hierarchical Z [4] order space-filling curve to sample the data domain. The space filling curve traverses the grid in such an order that a coarse representation of the grid is first obtained, and is further refined as one moves along the curve. HZ order has been previously used for creating multi-resolution representations of scientific data [1]. At every iteration (resolution level) we downsample in all three dimensions (first in X, then in Y and then in Z) leading to a data reduction in size by a factor of 8. This is similar to octree-style decomposition, where the 3D domain is decomposed into eight octants at a time. Note that with a sampling rate *s*, we select one sample every s samples in each dimension. Figure 2 shows an 16^2 dataset sampled at rate of 4 and 2.

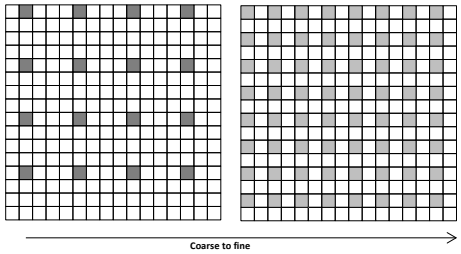


Figure 2. Sub-sampling a 16^2 data with sub-sampling rates 4 (left) and 2 (right) (both X and Y dimensions).

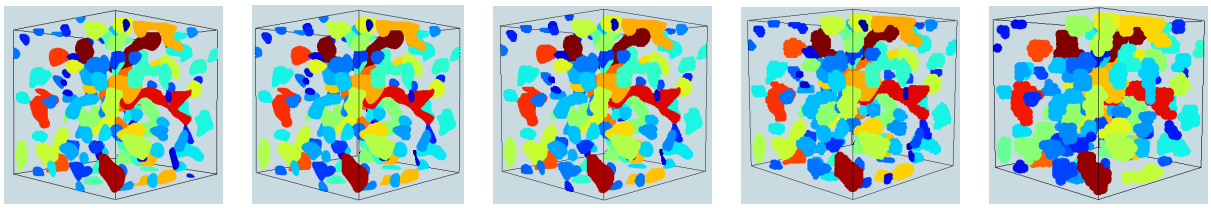
1.2 Wavelets and compression

Subsampling is inexpensive but is often not satisfactory for some analysis tasks such as visualization, as these tasks are sensitive to aliasing and often benefit from a low-pass filter on the original data. To this end, one of the future directions for our work is to enable discrete wavelet transform (DWT) of the dataset that can be used to make better approximations. We also intend to add a compression component in our software stack that can be used to significantly reduce the amount of data being moved during the communication phase of any parallel analysis algorithm.

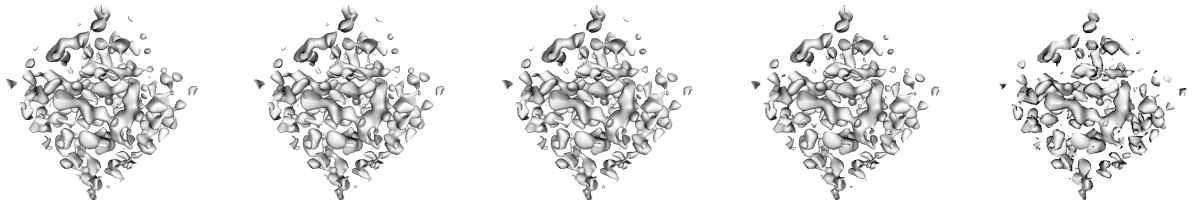
2 Parallel merge tree

We demonstrate the efficacy of our sampling scheme on a parallel implementation of the merge tree topological analysis algorithm [2]. The merge tree computation is a good representative of the feature detection algorithms. The merge tree encodes the evolution of connected components of the super-level sets of a given scalar function defined on the given domain, where the superlevel set is the region of the domain above a certain function value. The geometric descriptions of the super-level sets are often needed for analysis, for example, to track features, to determine their volumes and shapes, and for visualization.

The distributed merge tree computation involves three stages (Figure 4). In the first stage every process computes its local merge tree. The second stage joins subsets of the local trees to form the merge trees of the joined blocks. This resultant tree is then given to the participating local trees in the third stage, so that they can correct themselves based on the new information received. All three stages benefit from data-reduction offered by sub-sampling.



(a) Full resolution (512^3) **(b)** sampling rate 2 (256^3) **(c)** sampling rate 4 (128^3) **(d)** sampling rate 8 (64^3) **(e)** sampling rate 16 (32^3)



(f) Full resolution (512^3) **(g)** sampling rate 2 (256^3) **(h)** sampling rate 4 (128^3) **(i)** sampling rate 8 (64^3) **(j)** sampling rate 16 (32^3)

Figure 3. Top row shows the features extracted from the HCCI dataset at varying sampling rates. These features, in the simulation, represent ignition regions. Bottom row shows the isosurfaced extracted for the same dataset at different sampling rates.

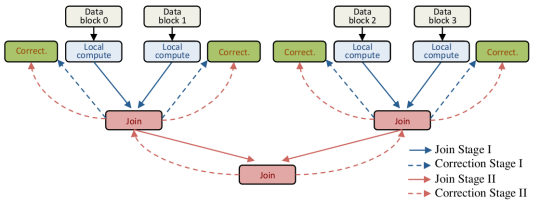


Figure 4. Schematic diagram showing the Parallel merge tree.

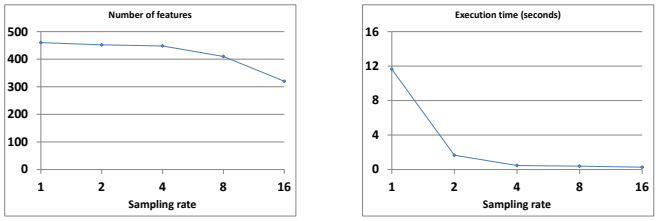


Figure 5. (left) Number of features and (right) PMT time to completion for the HCCI dataset.

We ran the parallel merge tree (PMT) analysis algorithm on dataset of resolution 512^3 consisting of single precision floats. The dataset has been generated with the KAUST Adaptive Reacting Flow Solvers (KARFS) [3] simulator on Shaheen II supercomputer and represents an autoignition process in a Homogeneous-Charge Compression Ignition (HCCI) engine. We ran the analysis runs on Tesla cluster (512 Xeon X5550 2.67GHz Processors) at the SCI institute of the University of Utah. We fixed the core count to 64, while changing the sampling rate from 1 to 16 (1, 2, 4, 8, 16), hence varying the overall resolution from 512^3 (no sampling) to 32^3 (sampling rate of 16). The PMT analysis algorithm generates the merge tree of the dataset, which is then used to perform segmentation. Segmentation as a result produces the features of interest for our dataset. We have presented the extracted features for all runs in Figure 3. We also present the total time for each of these cases in Figure 5 (right). From Figure 3, we observe that visually there is almost no loss in feature when down sampling from 512^3 to 128^3, it is only after that resolution we start to see a discernible loss of features. For example at sampling rate of 8 and 16 we can see fewer features at top left corner of the dataset (see Figure 3d and 3e) . We also observe a significant improvement (close to a factor of 8) in execution time as we go from full resolution (512^3) to 1/8th resolution (256^3). The rate of improvement slows down with higher sampling rate, mainly because total time starts to get dominated by the communication phase. We also computed the actual number of features at all sampling rates (see Figure 5 (left)). It can be seem that we incur a total loss of 140 features (460 at full data vs. 320 at sampling rate of 16) while going down in execution time from 11.5 seconds to 0.27 seconds. Overall, we observe that the time to completion for the PMT algorithm reduces significantly with sub-sampling, hence, also making it possible for them to be run in in-situ mode. Similar to PMT, we also observe a small describable differences among the iso-surface extracted at different sampling rates (see Figure 3 (bottom row)). One of the future work is to come up with heuristics to quantify the differences in output of the analysis tasks.

References

[1] S. Kumar, V. Vishwanath, P. Carns, J.A Levine, R. Latham, G. Scorzelli, H. Kolla, R. Grout, R. Ross, M.E. Papka, J. Chen, and V. Pascucci. 2012. Efficient data restructuring and aggregation for I/O acceleration in PIDX. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*. 1–11. DOI: https://doi.org/10.1109/SC.2012.54

[2] Aaditya G Landge, Valerio Pascucci, Attila Gyulassy, Janine C Bennett, Hemanth Kolla, Jacqueline Chen, and Peer-Timo Bremer. 2014. In-situ feature extraction of large scale combustion simulations using segmented merge trees. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 1020–1031.

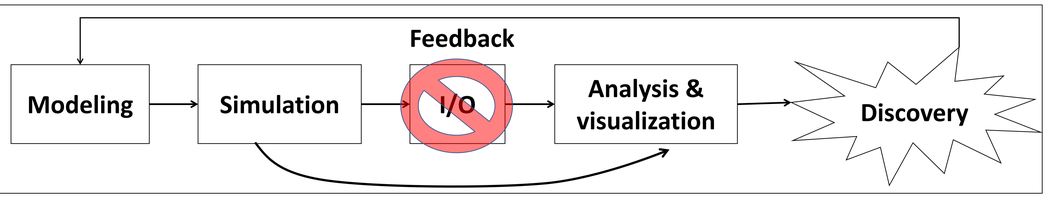
[3] Bok Jik Lee, Xu Xiao, Francisco E. Hernandez Perez, Hong G. Im, and Ramanan Sankaran. 2012. KARFS: A Combustion DNS solver for Hybrid Computing Architectures. (2012). Poster presented at 36th International Symposium on Combustion, Seoul, South Korea, 2016.

[4] Valerio Pascucci and Randall J. Frank. 2001. Global Static Indexing for Real-time Exploration of Very Large Regular Grids. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (SC '01)*. ACM, 2–2. DOI: https://doi.org/10.1145/582034.582036

ACCELERATING IN-SITU ANALYSIS AND VISUALIZATION WITH SUBSAMPLING

Sidharth Kumar, Steve Petruzza , Duong Hoang, Valerio Pascucci

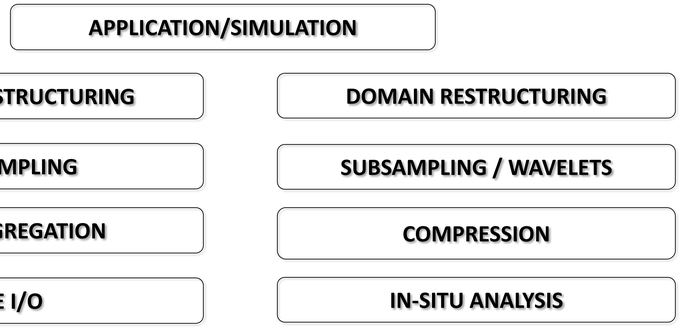
IN-SITU ANALYSIS FRAMEWORK



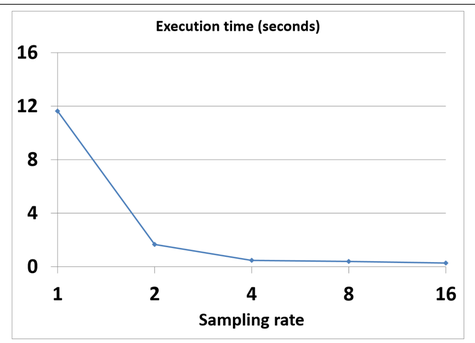
MOTIVATION AND CONTRIBUTION

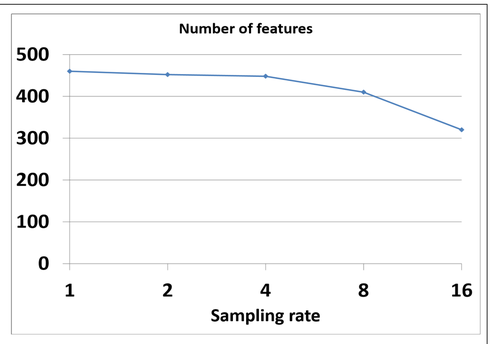
- The increasing gap between available compute power and I/O capabilities is driving more users to adopt in-situ processing capabilities.
- Making the transition from post-processing to in-situ mode is challenging, as parallel analysis algorithms do not always scale well with the actual simulation code. This is observed because these algorithms either involve global reduction operations with significant communication overheads or have to perform an overwhelmingly large amount of computation.
- In this work we propose a software stack, designed specially to expedite parallel analysis algorithms, thus making it possible for them to work in in-situ mode. The software stack (part of the PIDX I/O framework) comprises of data-reduction techniques, sub-sampling and compression, used to reduce the computation load of the analysis algorithm.
- We demonstrate the efficacy of our pipeline using the parallel merge tree analysis algorithm. We also study the trade-off between the I/O gains and the corresponding error induced by the data reduction techniques.

DATA MOVEMENT PIPELINE FOR PIDX IN I/O MODE AND IN-SITU MODE



PIDX IN-SITU PERFORMANCE RESULTS

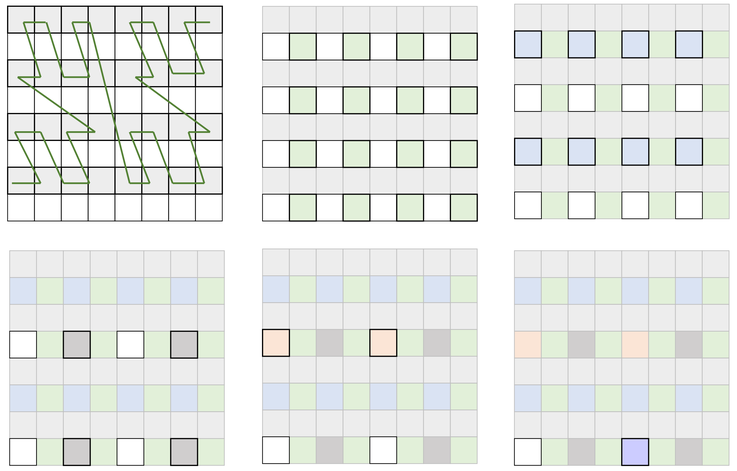


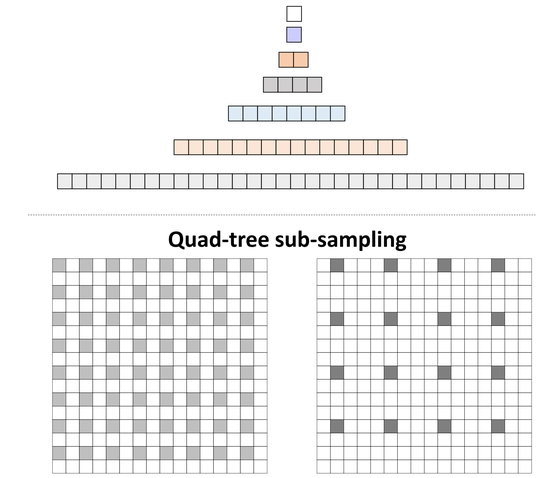


- Parallel merge tree analysis algorithm on dataset of resolution 512x512x512 consisting of single precision floats, at core count 64, and sampling rate varied from 1 to 16 (1,2,4,8,16).
- Resolution varied from 512x512x512 (no sampling) to 32x32x32 (sampling rate of 16).
- The rate of improvement slows down with higher sampling rate, because total time starts to get dominated by the communication phase
- Incur a total loss of 140 features (460 at full data vs. 320 at sampling rate of 16) while going down in execution time from 11.5 seconds to 0.27 seconds

SUB-SAMPLING TECHNIQUES

Hierarchical Z orders pace-filling curve to sample the data domain

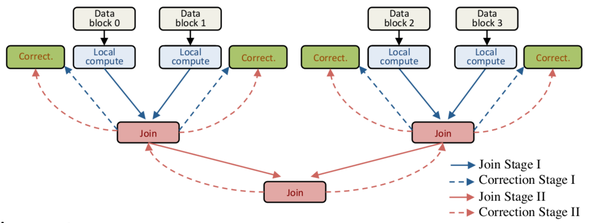




Quad-tree sub-sampling

Sub-sampling a 16x16 data with sub-sampling rates 4 and 2

PARALLEL MERGE TREE (ANALYSIS ALGORITHM)

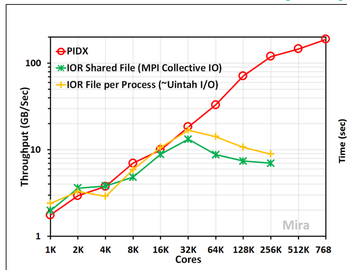


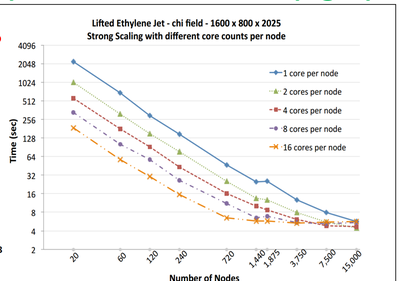
Three stages:

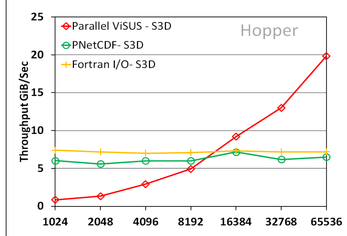
1. Every process computes its local merge tree.
2. Subsets of the local trees are joined to form the merge trees of the joined blocks.
3. The resultant tree is given to the participating local trees, they then correct themselves based on the new information.

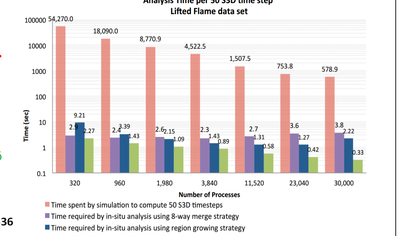
All stages benefit from data-reduction by sub-sampling.

PIDX I/O SCALING (left) AND PMT SCALING (right)

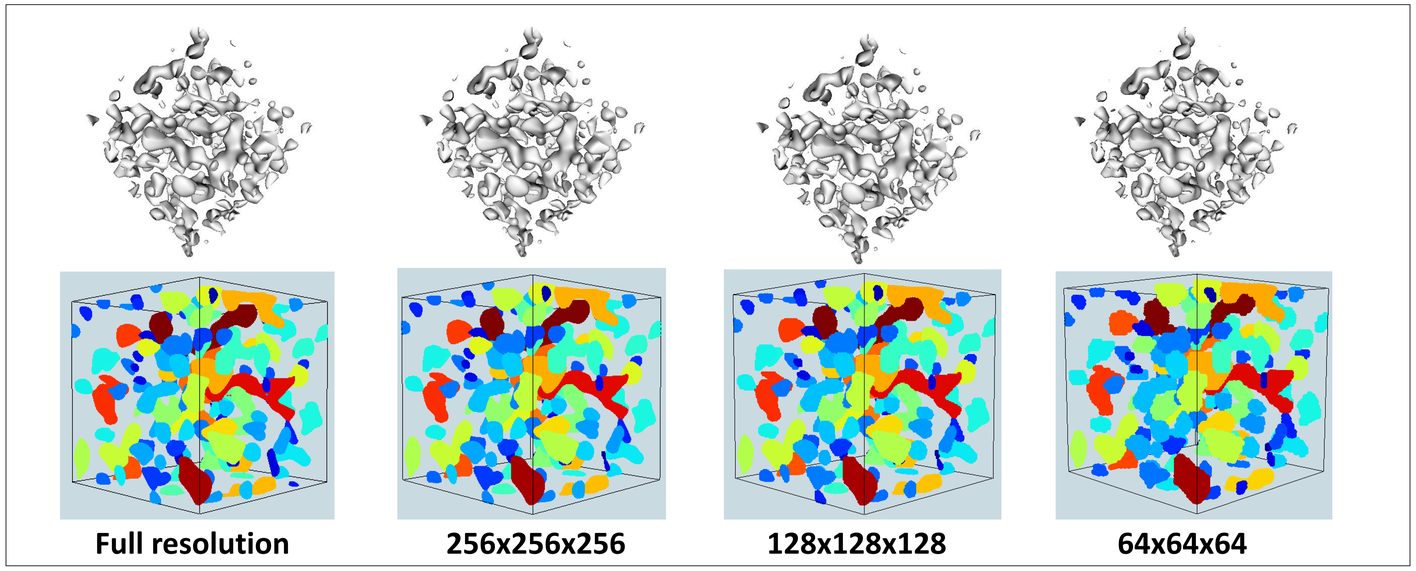








ISO-SURFACE AND FEATURE VISUALIZATION OF COMBUSTION DATA FOR VARYING SAMPLING RATES



Full resolution | **256x256x256** | **128x128x128** | **64x64x64**