Milestone 4547

Data Co-Processing for Extreme Scale Analysis

SAND# 2013-1427 P

Executive Summary of Milestone Report

March 5, 2013

David Rogers, Ron Oldfield, Kenneth Moreland and Nathan Fabian

Sandia National Laboratories

Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Summary

- Milestone 4745 "Data Co-Processing for Extreme Scale Analysis" was successfully completed on time, and demonstrated against the letter and spirit of stated Milestone.
- The Milestone Team completed over 10.5 million cpu hours of Cielo tests on both *in situ* and *in transit* analysis capabilities on a problem provided by a Sandia analyst.
- The results of these experiments have been detailed in a SAND report, which is published as an unclassified unlimited release document, available to the entire mod/sim community

The path to Exascale

Milestone 4745 is an important step in capability development, customer engagement, and scalability development on the path to exascale. It represents significant work on the development of both *Catalyst*, an open source *in situ* analysis capability, and *Nessie*, an open source data services capability.

This Milestone is part of an integrated R&D roadmap aimed at characterizing, understanding, and promoting solutions for complex analysis problems on advanced architectures.

It is an important foundation step in developing cross-cutting capabilities.

Milestone 4745

SC calculations produce complex datasets that are increasingly difficult to explore and understand using traditional post-processing workflows. To advance understanding of underlying physics, uncertainties, and results of ASC codes, SNL must gather as much relevant data as possible from large simulations. This drives SNL to couple data analysis and visualization capability with a running simulation, so that high fidelity data can be extracted and written to disk. This Milestone evaluates two approaches for providing such a coupling:

- In-situ processing provides ``tightly-coupled'' analysis capabilities through libraries linked directly with the simulation. SNL has collaborated on developing an in-situ capability designed for this purpose.
- In-transit processing provides ``loosely-coupled'' analysis capabilities by performing the analysis on separate processing resources. SNL provides this capability through a ``data services'' capability designed for this purpose.

SNL will engineer, test and evaluate customer-driven operations on large-scale data created by a running simulation. The data operations will be performed by instrumented versions of both the in-situ and in-transit solutions, with the resulting performance data published and made available to the ASC community.

A program review will be conducted, and its results documented. A report will be submitted as a record of milestone completion.

Motivation

SC calculations produce complex datasets that are increasingly difficult to explore and understand using traditional post-processing workflows. To advance understanding of underlying physics, uncertainties, and results of ASC codes, SNL must gather as much relevant data as possible from large simulations. This drives SNL to couple data analysis and visualization capability with a running simulation, so that high fidelity data can be extracted and written to disk.

- *Note: ASC program will benefit from a detailed understanding of the relationship between analyst tasks, analysis operations, and disk I/O performance.*

In situ and *in transit* workflows

- *In situ* processing provides ``tightly-coupled'' analysis capabilities through libraries linked directly with the simulation. SNL has collaborated on developing an *in situ* capability designed for this purpose.

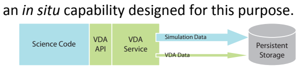


Diagram of *in situ* workflow, accomplished in this Milestone through the use of *Catalyst*, an open source, VTK-based analysis library.

- *In transit* processing provides ``loosely-coupled'' analysis capabilities by performing the analysis on separate processing resources. SNL provides this capability through a ``data services'' capability designed for this purpose.

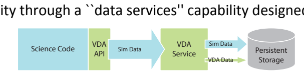


Diagram of *in transit* workflow, in which the science code communicates with data services nodes to perform analysis operations. This is accomplished in this Milestone through the use of *Nessie*, an open source data services library.

Milestone 4745, completion criteria

SC calculations produce complex datasets that are increasingly difficult to explore and understand using traditional post-processing workflows. To advance understanding of underlying physics, uncertainties, and results of ASC codes, SNL must gather as much relevant data as possible from large simulations. This drives SNL to couple data analysis and visualization capability with a running simulation, so that high fidelity data can be extracted and written to disk. This Milestone evaluates two approaches for providing such a coupling:

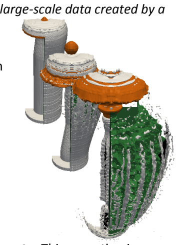
- In-situ processing provides ``tightly-coupled'' analysis capabilities through libraries linked directly with the simulation. SNL has collaborated on developing an in-situ capability designed for this purpose.
- In-transit processing provides ``loosely-coupled'' analysis capabilities by performing the analysis on separate processing resources. SNL provides this capability through a ``data services'' capability designed for this purpose.

SNL will engineer, test and evaluate customer-driven operations on large-scale data created by a running simulation. The data operations will be performed by instrumented versions of both the in-situ and in-transit solutions, with the resulting performance data published and made available to the ASC community.

A program review will be conducted, and its results documented. A report will be submitted as a record of milestone completion.

Experiment Driver

Milestone focused on *"customer-driven operations on large-scale data created by a running simulation"*



Customer driver use case: characterize fragments in an explosion simulation, an analysis step critical for understanding shock physics

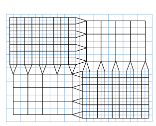
- Partner: Jason Wilke
- Critical steps
  - Find fragments (multiple operations required)
  - Characterize fragments (mass, velocity, etc.)
  - Extract useful information

Milestone experiments focused on identifying the fragments. This operation is a significantly complex part the analysis, so it serves as a useful way to characterize the operations in the driver use case.

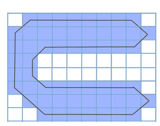
Full range of data experiments run at 32k cores on Cielo. Partial experiments done at 64k cores on Cielo. This report presents results from the 32k runs.

Fragment detection

- Operations required for fragment detection (requires a watertight surface)
  1. Find block neighbors
  2. Build a conforming mesh over AMR boundaries
  3. Identify boundaries of fragments



Step 2



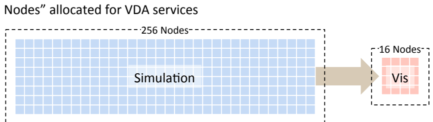
Step 3

Implemented Workflows

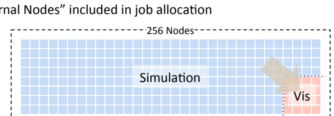
- ***In situ***: A CTH job that directly runs *in situ* data analysis
  - **Baseline**: Basic algorithm with somewhat redundant step of global communication to find AMR block neighbors
  - **Refined**: Improved algorithm that gets AMR block neighbors from CTH
- ***In transit***: CTH transfers data to separate server job
  - **Extra nodes**: CTH job size same as other runs, extra nodes are used to allocate the VDA service
  - **Internal nodes**: CTH job given fewer nodes that are assigned to VDA service so that together both jobs use the same nodes as other runs
- **Post-processing**: Write Spyplot files from CTH, then post process analysis by reading back in and batch processing in ParaView.

In Transit Allocations

"Extra Nodes" allocated for VDA services



"Internal Nodes" included in job allocation



Experiment Configurations

- All experiments performed on Cielo supercomputer at LANL, jointly managed by Los Alamos National Laboratory and Sandia National Laboratories
  - 8,944 node Cray XE6
  - Node: 2 AMD Opteron 6136 (Magny-Cours) 8-way processor chips
    - Total of 16 cores/node
    - 2.4 GHz peak computation speed per core
  - Peak of 1.37 Petaflops
  - 32 GB memory/node

Experiment, cont'd

- All applications complete 500 cycles (i.e., timestep calculations) of the CTH code.
- The first four applications execute an analysis operation once every 10 cycles
- Spyplot file application outputs spyplot data at a fixed interval in simulated time, calculated so that the application executed the same number of analysis operations performed by the *in situ* and *in transit* applications
  - Total number of analysis operations is the same
- Data captured was from instrumented code and HPCToolkit

Experiment, cont'd

- For each application, we ran strong scaling experiments for three different datasets.
  - Each data set comes from the same initial conditions but with a different maximum level of refinement
  - Measurements of different job sizes with different data set sizes provides a weak scaling overview.

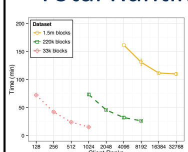
CTH				*In transit* Server			
Host		*In transit* Internal		Extra Nodes		Internal Nodes	
Cores	Nodes	Cores	Nodes	Cores	Nodes	Cores	Nodes
33K Blocks — 5 levels							
128	8	96	6	16	2	16	2
256	16	224	14	16	2	16	2
512	32	480	30	16	2	16	2
1,024	64	992	62	16	2	16	2
220K Blocks — 6 levels							
1,024	64	768	48	128	16	128	16
2,048	128	1,792	112	128	16	128	16
4,096	256	3,840	240	128	16	128	16
8,192	512	7,936	496	128	16	128	16
1.5M Blocks — 7 levels							
4,096	256	2,496	156	1,024	128	800	100
8,192	512	6,592	412	1,024	128	800	100
16,384	1,024	14,784	924	1,024	128	800	100
32,768	2,048	31,168	1,948	1,024	128	800	100
65,536	4,096	63,936	3,996	1,024	128	800	100

Table shows the range of core sizes used for the various experiments. For every application we used the maximum 16 cores-per-node for the CTH client, since CTH is primarily bound by computation and scales very well.

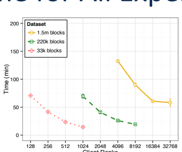
CTH				*In transit* Server			
Host		*In transit* Internal		Extra Nodes		Internal Nodes	
Cores	Nodes	Cores	Nodes	Cores	Nodes	Cores	Nodes
33K Blocks — 5 levels							
128	8	96	6	16	2	16	2
256	16	224	14	16	2	16	2
512	32	480	30	16	2	16	2
1,024	64	992	62	16	2	16	2
220K Blocks — 6 levels							
1,024	64	768	48	128	16	128	16
2,048	128	1,792	112	128	16	128	16
4,096	256	3,840	240	128	16	128	16
8,192	512	7,936	496	128	16	128	16
1.5M Blocks — 7 levels							
4,096	256	2,496	156	1,024	128	800	100
8,192	512	6,592	412	1,024	128	800	100
16,384	1,024	14,784	924	1,024	128	800	100
32,768	2,048	31,168	1,948	1,024	128	800	100
65,536	4,096	63,936	3,996	1,024	128	800	100

Results

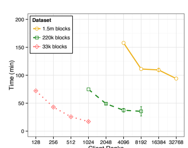
Total Runtime for All Experiments



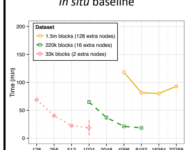
In situ baseline



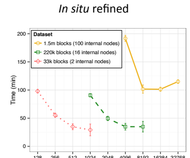
In situ refined



Disk-based post-processing



In transit extra nodes



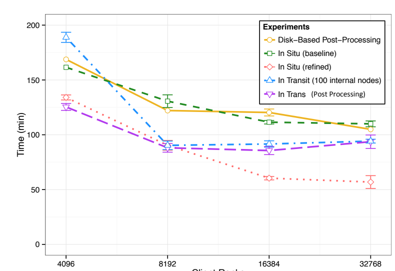
In transit internal nodes

5 applications
3 datasets

Strong scaling for each dataset

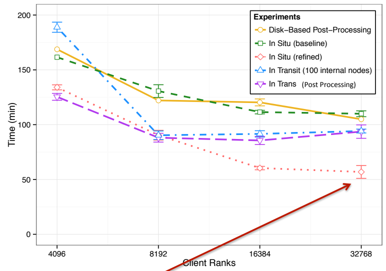
Error bars show variance

Pipeline Summary Timing (1.5m blocks)



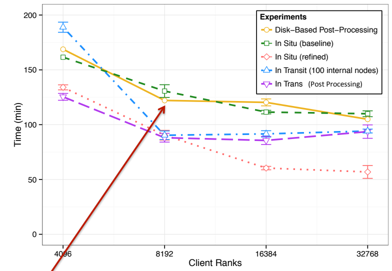
Acceptable scaling performance, with the exception of the baseline algorithm.

Pipeline Summary Timing (1.5m blocks)



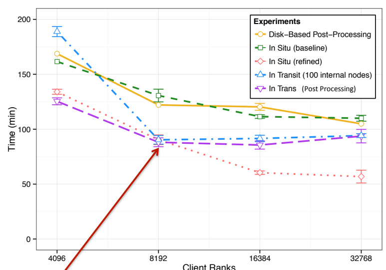
No significant improvement at 32K cores. Probably insufficient work for analysis (only 45 blocks per process).

Pipeline Summary Timing (1.5m blocks)



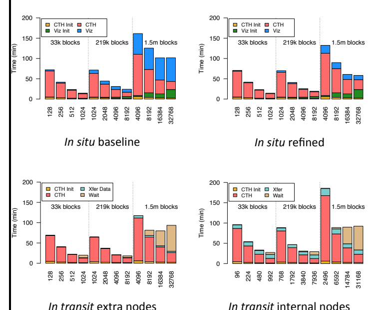
Writing files surprisingly fast. Although slower than most alternatives, still a viable option.

Pipeline Summary Timing (1.5m blocks)



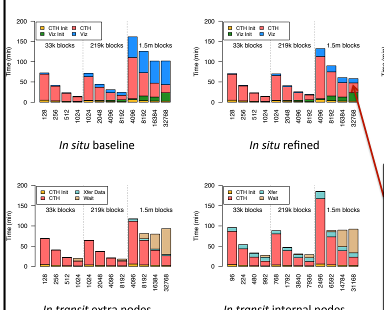
"Sweet spot" at 8K cores: *in transit* with unrefined algorithm equal to *in situ* with refined algorithm.

Timing Per Task



- CTH scales well.
- Baseline algorithm does not scale
- Disk I/O not bad

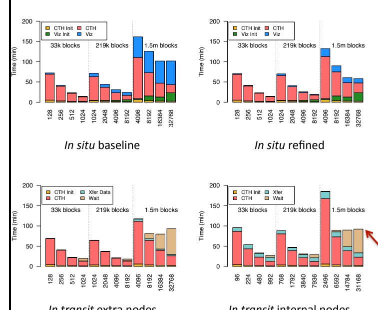
Timing Per Task



Refined analysis has much lower overhead, but initialization is problematic.

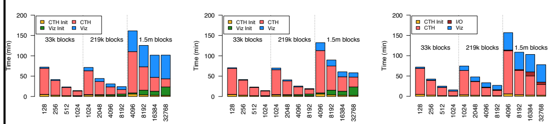
Refined algorithm requires additional data to be passed. Not done for *in transit* experiments.

Timing Per Task

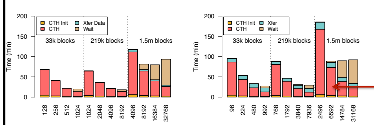


Service is a fixed size (100 nodes), the wait time should be independent of the number of cores on the client.

Timing Per Task



In situ baseline | *In situ* refined | Disk-based post-processing

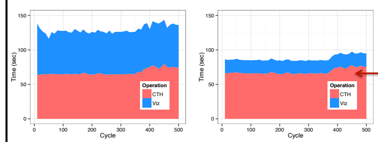


In transit extra nodes | *In transit* internal nodes

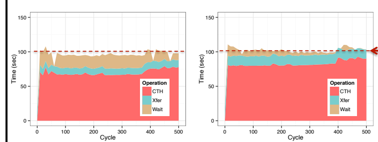
"sweet spot", where compute and analysis are balanced (equal)

Time-Series Analysis (8k cores)

10-cycle increments



In situ baseline | *In situ* refined



In transit extra nodes | *In transit* internal nodes

Although number of blocks changes very little, CTH runtime gets longer as simulation progresses.

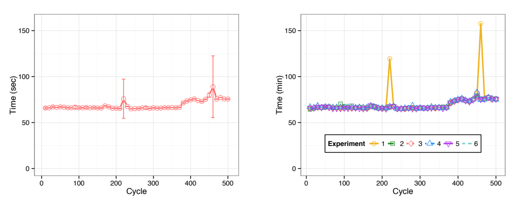
Vis time is roughly constant.

In transit will "win" when xfer+wait is less than viz.

In transit can flatten the runtime as long as extra simulation time consumes only wait time.

Time-Series Analysis: Variance

10-cycle increments

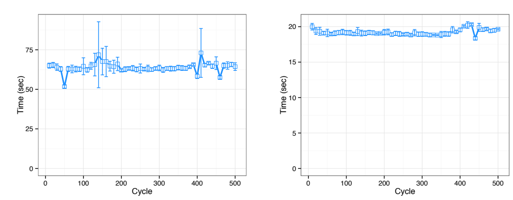


CTH (mean+std) | CTH (overlaid)

Anomalies that cause large variance are clearly identified in overlay plot.
Not sure the true cause of the outliers.

Time-Series Analysis: Variance

10-cycle increments

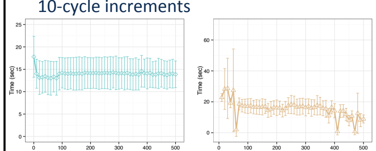


In situ baseline | *In situ* refined

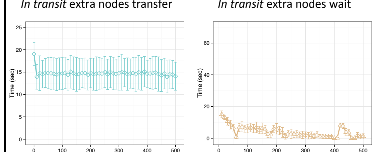
Refined algorithm has much less communication, resulting in less variance.
Scales are different in the two plots.

Time-Series Analysis: Variance

10-cycle increments



In transit extra nodes transfer | *In transit* extra nodes wait



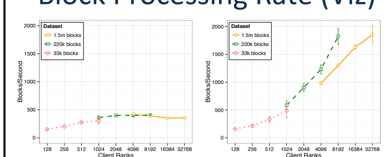
In transit internal nodes transfer | *In transit* internal nodes wait

In transit transfer times have noticeable variance.

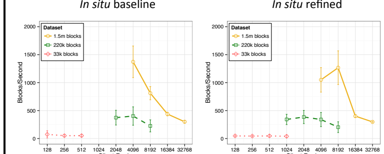
Possibly caused by placement issues.

Scheduler does not optimize allocations for transfers between jobs.

Block Processing Rate (Viz)



In situ baseline | *In situ* refined



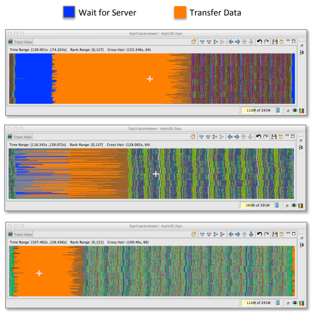
In transit extra (effective) | *In situ* inclusive (effective)

Refined algorithm is scalable, baseline algorithm is not. (Verifies previous work.)

In-transit plots show "effective" processing rate. Since viz time is flat, we expect the effective rate to decrease.

At 4k and 8k, effective processing rate of in transit outperforms in situ.

In Transit Node Scaling

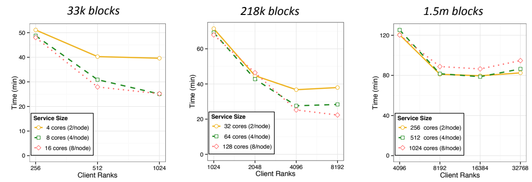


2 server cores: 64:1
- 10 cycles in 37 secs
- Client idle waiting for servers (also affects xfers)

4 server cores: 32:1
- 10 cycles in 23 secs

8 server cores: 16:1
- 10 cycles in 19 secs
- Less than 1% time waiting

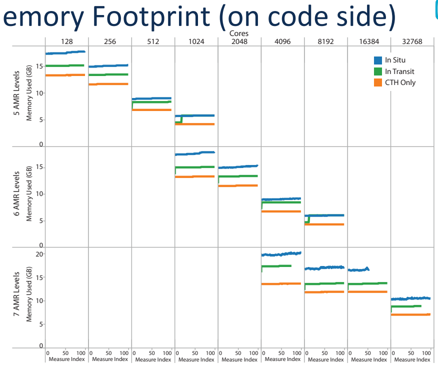
In Transit Node Scaling



For small datasets, there is clear benefit to using 4 and 8 cores/node (agreement previous slide)

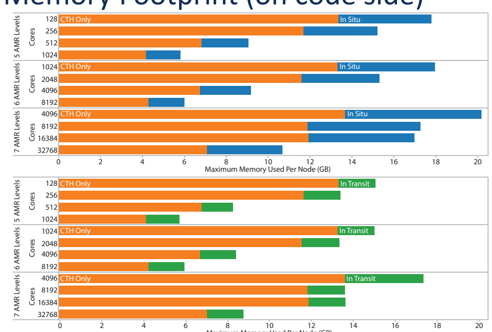
For 1.5m blocks datasets (at large scale), the opposite appears to be true. Needs further study.

Memory Footprint (on code side)



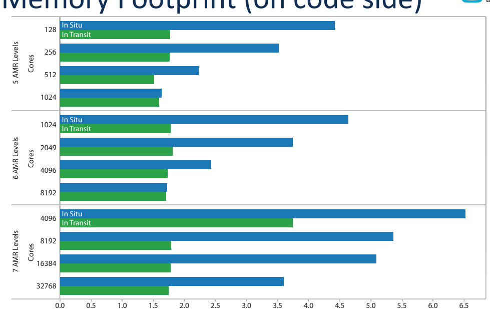
All memory measurements. Holds relatively steady for all workflows

Memory Footprint (on code side)



Memory overhead generally falls between 25% and 50%

Memory Footprint (on code side)



In transit generally has less memory overhead, but requires extra nodes allocated

Conclusions

Conclusions

In transit **can provide a performance improvement over *in situ* in some circumstances, but the window is narrower than we initially expected it would be.**

In transit data analysis has an added overhead above embedded *in situ* data analysis involving transferring data between parallel jobs. Given a data analysis algorithm with perfect linear scalability, we suspect *in transit* workflows will always have an added cost, and our results support this. With a data analysis algorithm that does not scale perfectly, possibly due to communication overhead, it is theoretically possible for *in transit* to be faster by reducing the size of the data analysis job. This is one of the motivations for choosing a data analysis task that requires significant communication. In our results, we do find instances where *in transit* is faster, but by a smaller margin and for fewer configurations than we initially anticipated. So although *in transit* has several other positive features, we do not anticipate performance to be the main motivations for using it.

Conclusions

The efficiency of *in transit* relies on balancing the time spent in simulation and data analysis.

The significant overhead cost, apart from data transfer, in the *in transit* workflow is the idle time spent in the simulation waiting for the visualization and data analysis service to become ready or the idle time spent in the visualization and data analysis service waiting for the simulation to send more data. This idle waiting time is minimized when the simulation and data analysis spend the same amount of wall clock time between transfers. Although not demonstrated in this work, it is possible to "auto-balance" the work between simulation and data analysis by, at every iteration of the simulation, transfer data to the data analysis if and only if the data analysis service is ready to accept more work. The disadvantage of such an approach is that the idle process time could be replaced with unnecessary extra data analysis or less data analysis than necessary. However, we suspect that controlling the amount of visualization and data analysis performed through job allocation sizes fits well with users' rules of thumb about resource allocation.

Conclusions

Memory overhead will be an important trade-off space.

The baseline amount of memory added to the CTH job to perform *in situ* processing is roughly 100MB per core. Considering that our embedded *in situ* library is a fully featured visualization toolkit containing over 2 million lines of code and algorithms developed over almost 2 decades, this overhead is not unreasonable. Nevertheless, this footprint can be problematic for simulations already tight on memory. Because of this, efforts are already underway to improve our memory footprint by making finer modules and being more selective on the available algorithms. This, of course, requires a compromise between the size of the library and the algorithms that are dynamically available. We also note that our algorithm has the potential to generate sizable meshes of its own. Thus, it may be fruitful to pursue and support incremental algorithms where possible.

Conclusions

Initialization time matters

Our scaling efforts to date focus on the scalability of the algorithms invoked during the run of a simulation. The initialization cost, a one-time penalty, has yet to be seriously considered. However, based on our HPCToolkit measurements, initialization becomes a significant cost at high process counts.

Disk-based I/O is not dead . . . yet.

Our initial assumption was that it would not be feasible to output full results at a fine enough temporal resolution from CTH to disk storage to perform our high fidelity data analysis. However, our control workflow shows that although the overall time to write data to disk and then read back again incurs a large cost, it is still realistic to do so. Thus, users may still choose to incur the extra overhead to use a traditional offline post-processing visualization and data analysis workflow.

Conclusions

Better job scheduling is important

One of the more complicated parts of running an *in transit* workflow is scheduling the simulation job and service job to run in tandem. Frankly, the capabilities of the scheduler are inadequate for our needs. We cannot start and stop jobs independently and make reconnections dynamically. Another experiment we would like to do but is challenging to schedule is to allow simulation and service to share nodes. Since each node has 16 cores, perhaps we could get better transfer performance by allocating one core per node for service and the rest for simulation. A similar scheduling scheme will be important to take advantage of burst buffers in future architectures.

Future Work

- Algorithm comparison. Three similar algorithms with three different scaling behaviors
  - Contour algorithm (perfectly scalable)
  - Refined water tight contours (reasonably scalable)
  - Baseline water tight contours (not scalable)
- No-wait analysis (*in transit*)
  - Perform analysis if and only if the service is ready
- Investigate initialization cost of *in situ* vis
- Zero copy transfers (*in transit*)
- Additional apps at Cielo scale
- Improved OS and runtime support
  - Scheduling, placement, node sharing, specialized runtimes, …

Summary

- Milestone 4745 “Data Co-Processing for Extreme Scale Analysis” was successfully completed on time, and demonstrated against the letter and spirit of stated Milestone.
- The Milestone Team completed over 9 million node hours of Cielo tests on both *in situ* and *in transit* analysis capabilities on a problem provided by a Sandia analyst.
- The results of these experiments have been detailed in a SAND report, which is published as an unclassified unlimited release document, available to the entire mod/sim community