RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

RDI²

Addressing Data Resiliency for Staging Based Scientific Workflows

Shaohua Duan, Pradeep Subedi, Philip E. Davis, Manish Parashar

Rutgers Discovery Informatics Institute

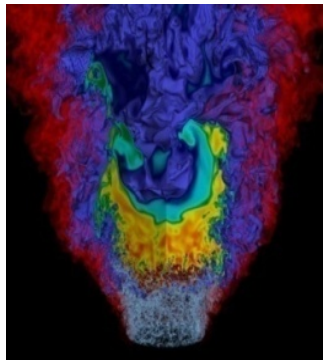
Rutgers, The State University of New Jersey, USA

Staging Based Workflows

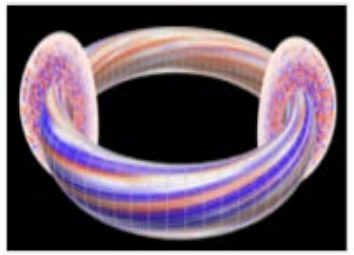
Data staging techniques have emerged as effective solutions for addressing data related challenges such as I/O storage challenge and data movement challenge at extreme scales scientific workflow.

❑ Characteristics

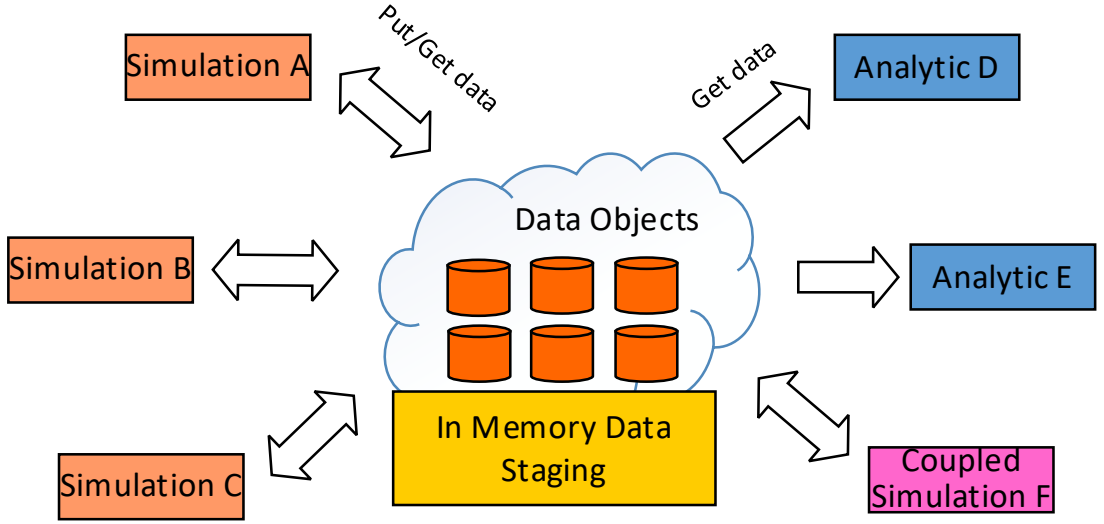
- ✓ In-memory storage distributed across set of cores/nodes
- ✓ Support runtime data processing, sharing and exchange



Combustion



Plasma Fusion



Silent Errors

❑A **Silent Error** (also known as **Silent Data Corruption**) is an unintentional change to bits (1 -> 0 or 0 -> 1) in memory which can impact correctness and performance of both applications and workflows.

❑For Current Systems

Jaguar: silent errors have been observed once per day (2010).
Hopper: encounters ~32 FITs (failures per billion hours of operation) per DRAM device (2015).

❑For Extreme Scale Systems, the estimated MTBE would be in minutes.

Data based on available public records in:

V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi. "*Memory errors in modern systems: The good, the bad, and the ugly*". In Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'15), March 2015.

Silent Error Challenge Faced by Workflows

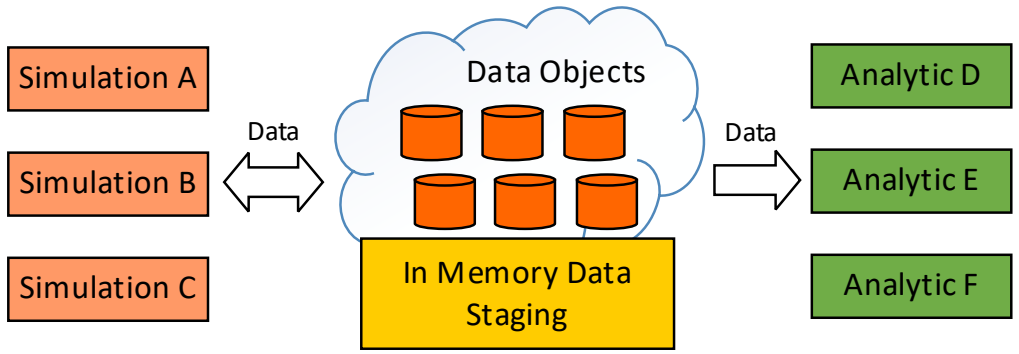
- **Silent errors in workflows**
  - ✓Addressing error detection for the application workflow including the analyses components remains an open challenge. The final results of the overall computation for workflows is the outputs of the workflow, and silent errors in any component of the workflow can invalidate these outputs.

  - ✓**Goal**: Providing a general and transparent silent error detection framework for in-situ scientific workflows.

  - ✓**Fault model**: Targeting any silent errors that can invalidate scientific dataset, regardless of their origin (software/hardware), during the execution of the workflow

Motivation for Error Detection in Staging

❑ Propagation of Silent Errors in Workflows



A typical data staging workflow

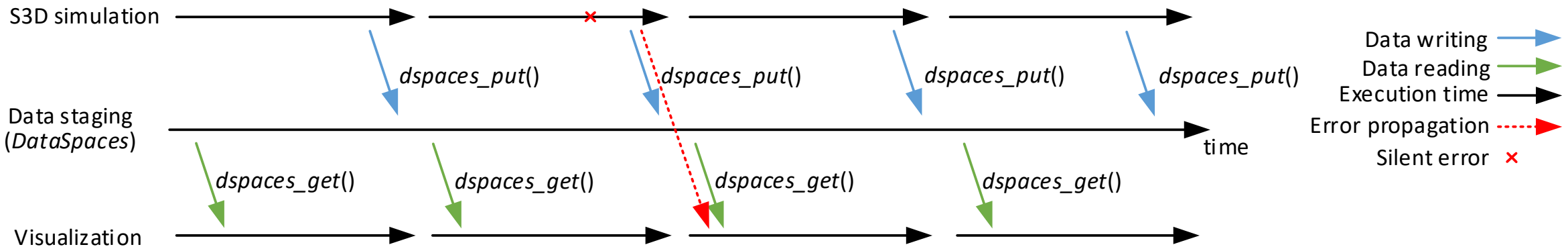
❑ S3D workflow

- S3D simulation generates data.
- Analytics processed data.

Observation:

1. Silent errors can be propagated.
2. Make the final result invalid.

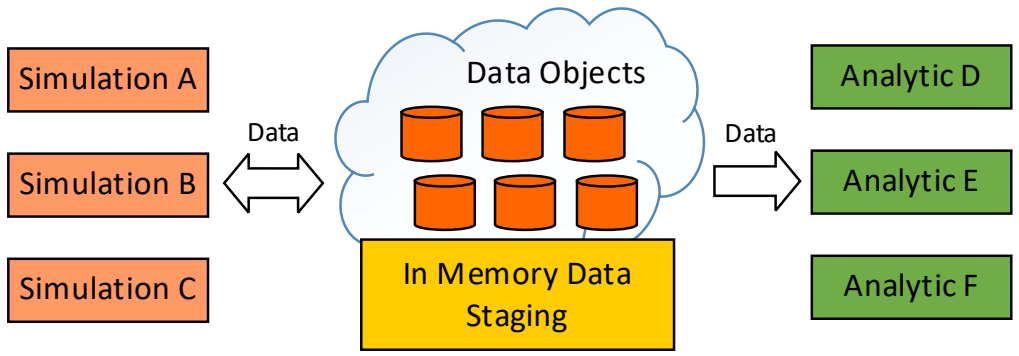
❑ Case Study 1: S3D workflow



Coupling and data-exchange pattern for the S3D coupled simulation workflow

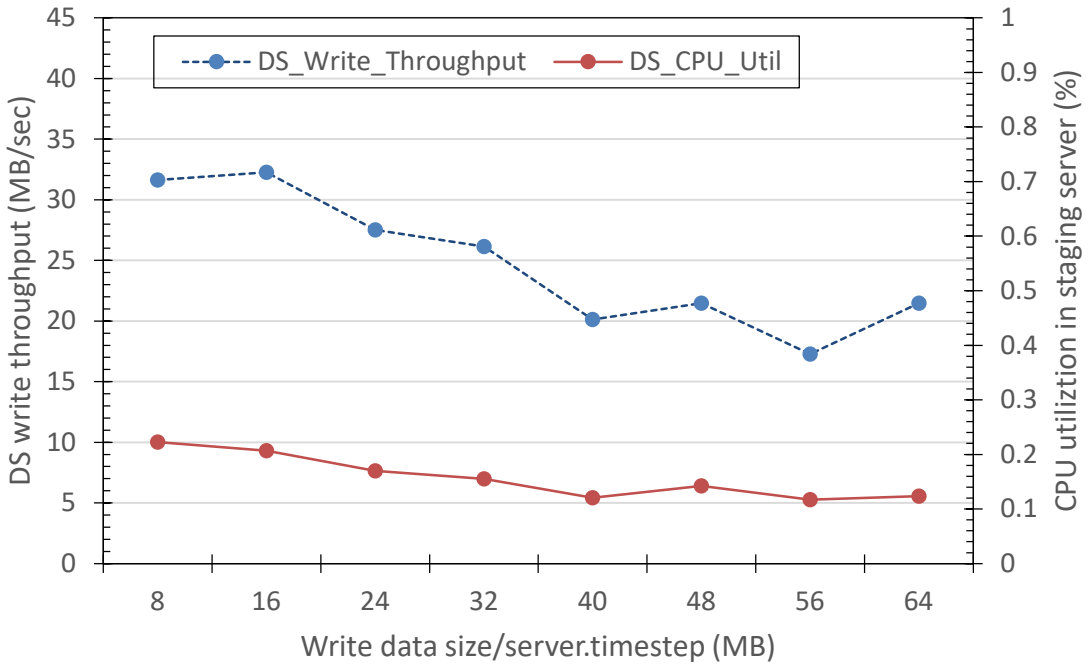
Motivation for Error Detection in Staging

❑ Utilizing Idle Compute Resource in Staging



A typical data staging workflow

CPU Utilization = compute time/write response time



❑ Case Study 2: Synthetic workflow

- A synthetic workflow on Titan Cray XK7 system.
- Write 8M~64M data for each server per time step (total 320M~2560M).

Observation:

1. The CPU utilization remained consistently low, and the maximum CPU utilization is 22%.
2. Write throughput reaches the peak and decrease.

Silent Error Detection Techniques

- **Process Replication**

  Detect errors by comparing computation results between replicated processes.

  ✓ High accuracy for error detection.

  ✓ Imposes large overheads.

- **Algorithm Based Fault Tolerance** (**ABFT**)

  Introduce information redundancy in the data, maintain this redundancy during computation, and finally verify output through redundancy.

  ✓ High accuracy and low overheads.

  ✓ Not general, some linear algebra applications.

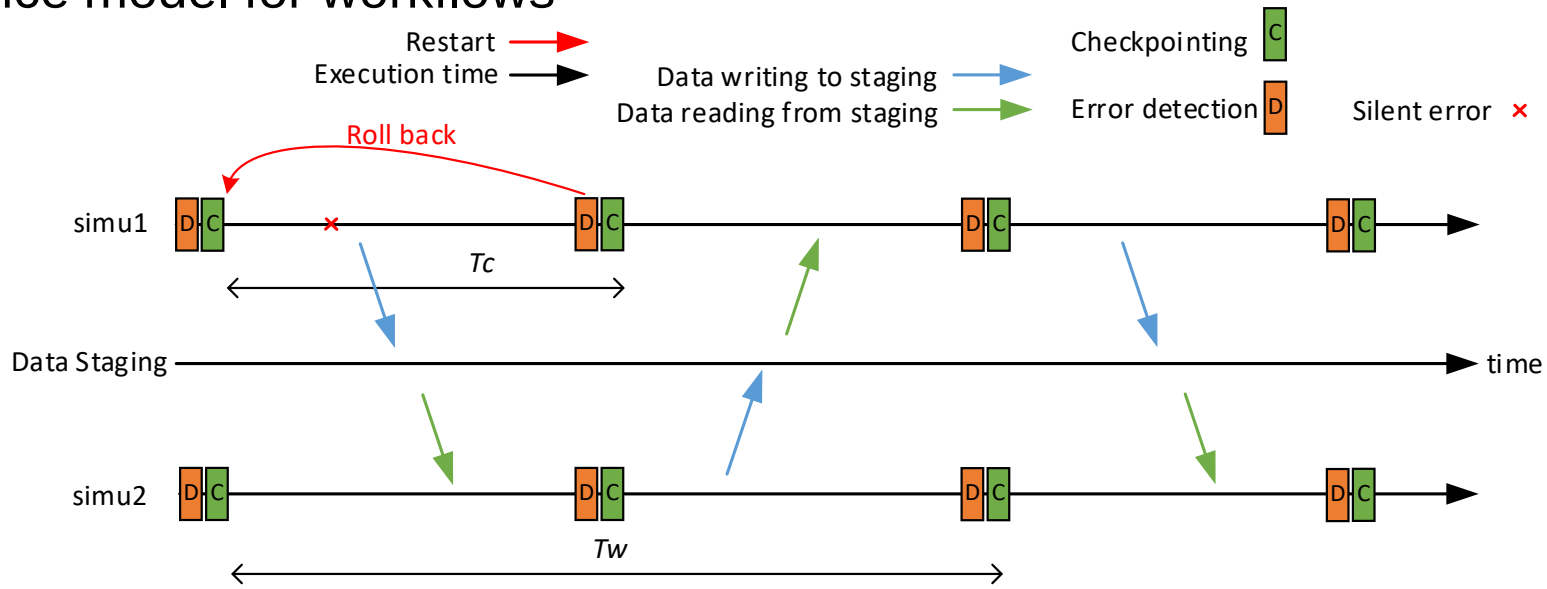
- **Real-time Data Analysis**

  Use interpolation techniques, such as time series prediction and spatial multivariate interpolation to recognize errors in scientific dataset.

  ✓ Lightweight, cover most of silent errors.

Modelling Error Detection in Staging

- Fault tolerance model for workflows



- Execution time of workflows:

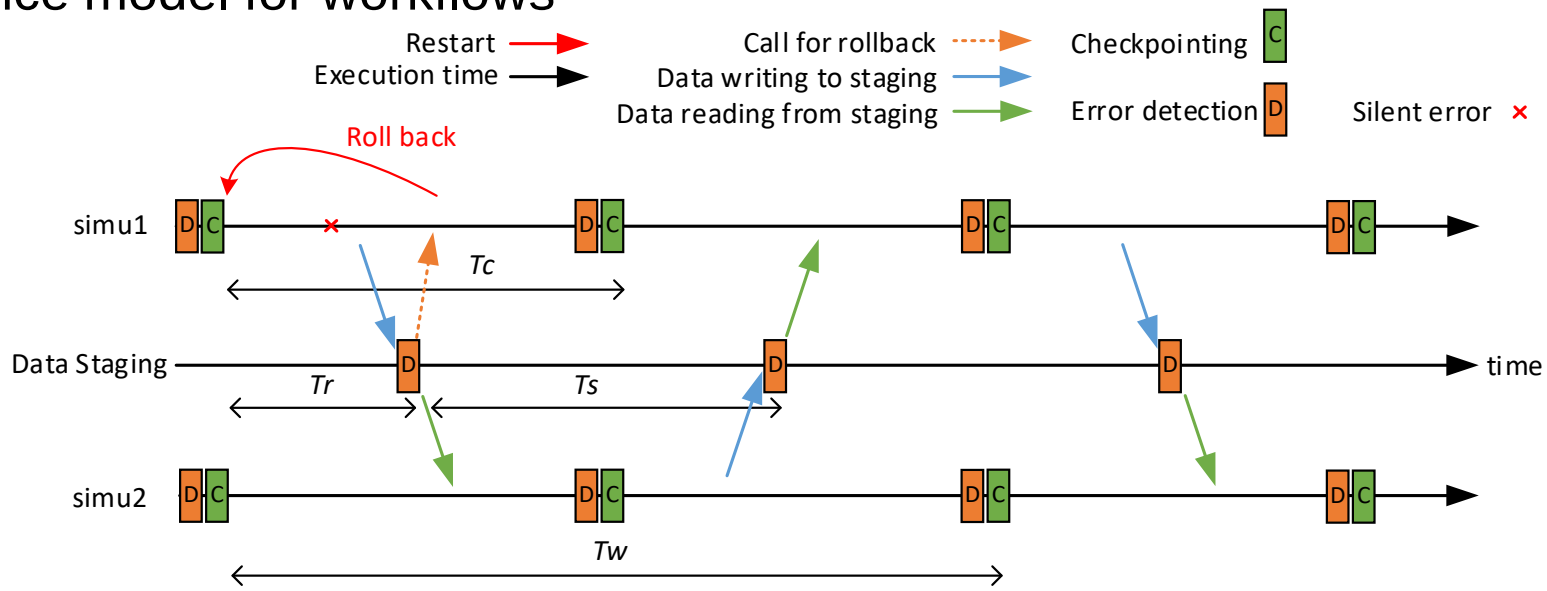
$$E(W_c) = \frac{T_w}{T_c}\left(\boxed{T_{DC}} + P_f * \boxed{T_{cC}}\right) + \boxed{W_{base}}$$

1 2 3

1. Fault tolerance cost: $T_{DC} = D + C$
2. Data Recovery cost: $T_{cC} = T_c - C$
3. Baseline: W_{base}

Modelling Error Detection in Staging

- Fault tolerance model for workflows



- Execution time of workflows:

$$E(W_s) = \frac{T_w}{T_c}\left(\boxed{T_{DC}} + P_f\left(\boxed{(1-R_s)T_{cC} + R_s T_r} + \boxed{T_p}\right)\right) + \frac{T_w}{T_s}\boxed{D_s} + \boxed{W_{base}}$$

1 2 4 1 3

1. Fault tolerance cost: $T_{DC} = D + C$, D_s
2. Data recovery cost: $(1-R_s)T_{cC} + R_s T_r$
3. Baseline: W_{base}
4. Unnecessary data recovery cost: $T_p = (1-P_s)T_r$

Modelling Error Detection in Staging

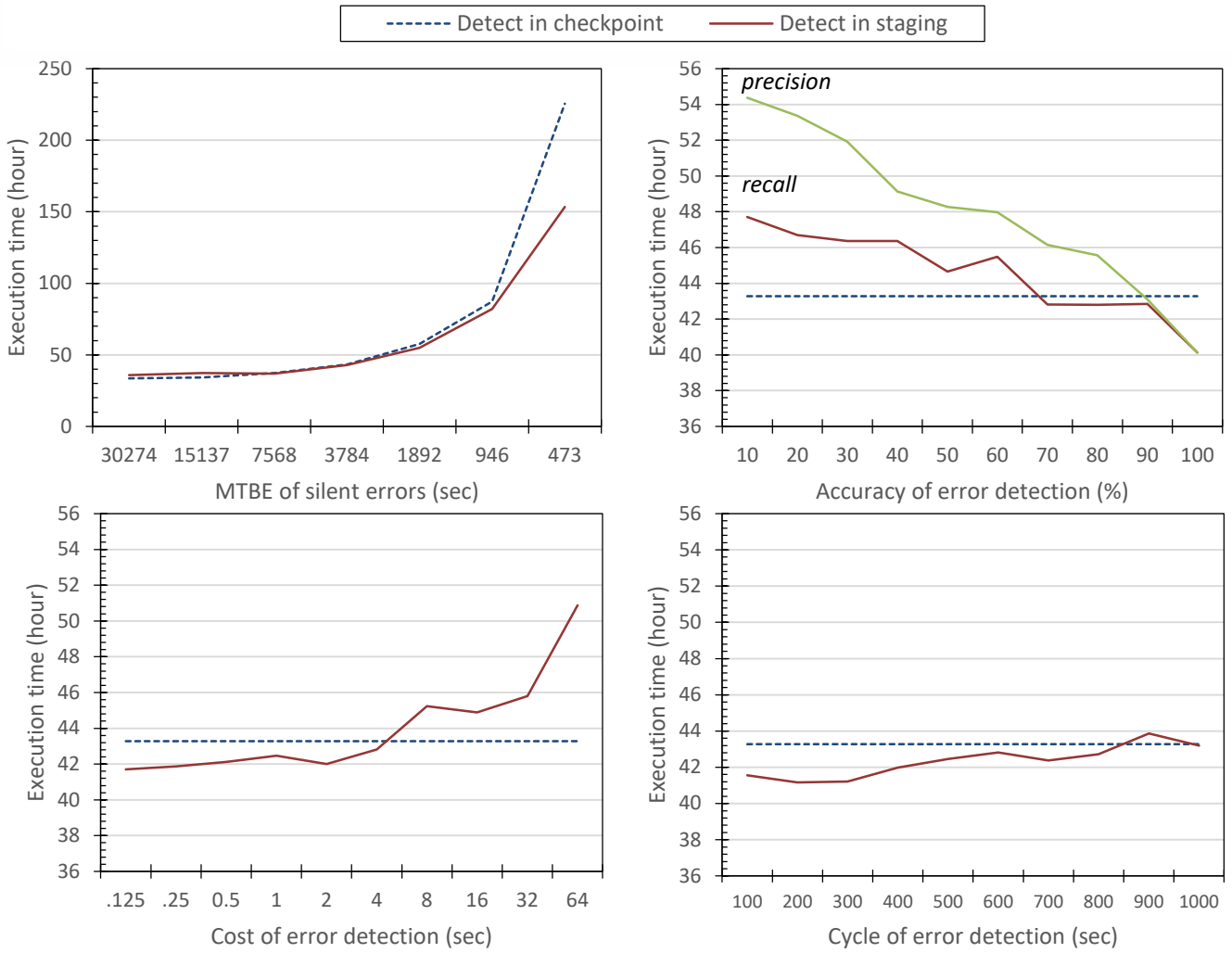
- Reduce workflow execution time:

$$E(W_c) - E(W_s) = \frac{T_w}{T_c} P_f \left(R_s (T_{cC} - T_r) - T_p \right) - \frac{T_w}{T_s} D_s$$

- the advantage of error detection in staging depends on following factors:
  - (1) The frequency of silent errors *MTBE* (positive relation with P_f);
  - (2) The *recall* and *precision* of error detection in staging R_s, P_s (positive relation with T_p, higher is better);
  - (3) The cost of error detection in staging D_s (lower is better);
  - (4) The frequency of error detection in staging T_s (positive relation with T_r).

Modelling Error Detection in Staging

- Reduce workflow execution time:



✓ Simulate the execution time of workflows using python program, and vary the MTBE, and accuracy, frequency, and cost of error detection.

Result:

(1) High frequent silent errors *MTBE*, high accuracy P_s, R_s, and light weight cost D_s will see more benefit.

(2) Low *precision* P_s can significantly deteriorate the performance.

Implementing Error Detection in Staging

- Error detection in CPU-GPU hybrid staging

- CPU cores in serve as communication and staging operations (Ex: indexing, storing) while offloading the error detection task to local GPU's.

- Partition dataset into small tiles, to adapt to relatively limited bandwidth and memory size on GPU devices.

- Fit for dedicated staging nodes or CPU-only workflows (Ex: S3D workflow).

Algorithm 1 Error Detection Processing

Input: spatial data P, meta data Q
Output: outlier value O_{max}, error decision $flag$

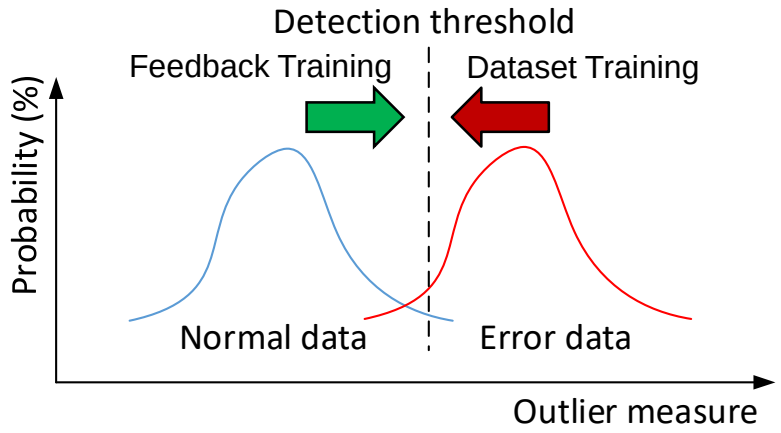
```
1:  procedure DSPACES_PUT(P)
2:      flag = 0
3:      Detector_Kernel(Q, P)
4:      if O_max > threshold then
5:          flag = 1
6:      end if
7:      insert P into staging
8:      update meta data Q
9:  end procedure
10: procedure DETECTOR_KERNEL(Q, P)
11:     for each point p in P do
12:         D = d(p)
13:         B = beta(p)
14:         O = D * B
15:         O_max = Max(O, O_max)
16:     end for
17: end procedure
```

pseudo code of spatial outlier error detection (SLOM) in hybrid staging

Implementing Error Detection in Staging

❑ Tuning *precision* and *recall* in Staging

- A low *precision* (high False Positive) can significantly deteriorate the performance of error detection in staging.
- **Dataset Training**: Insert bit flip errors into dataset, set a threshold to cover all of synthetic errors, and get high *recall.*
- **Feedback Training**: Verify the detected errors, adjust the threshold, and achieve high *precision*.



Dataset/Feedback training

$$\uparrow recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

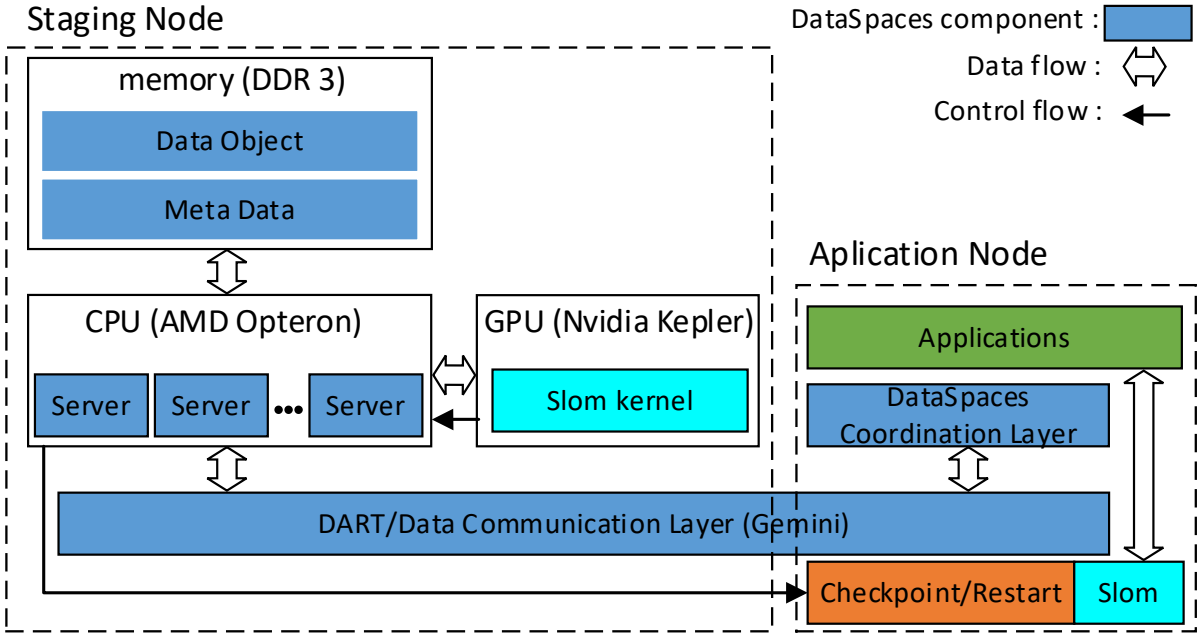
$$\uparrow precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Experimental Evaluation

- Hybrid Staging Architecture

Titan, hybrid-architecture Cray XK7

- 18,688 nodes
- Gemini interconnect
- 600 TB system memory
- 16-core AMD Opteron processor, with 32GB DDR3 RAM
- NVIDIA Kepler accelerator, with 6 GB GDDR RAM



Experiments buildup of error detection in staging on Titan Cray XK7

- **Dedicated staging node:**
  - DataSpaces, Error detection SLOM (GPU)
- **Applicartion node:**
  - S3D simulation/visualization, synthetic applications, checkpoint/ restart, Error detection SLOM

Experimental Evaluation

- **Performance Experiments**: write performance of staging servers with error detection.

Titan, hybrid-architecture Cray XK7

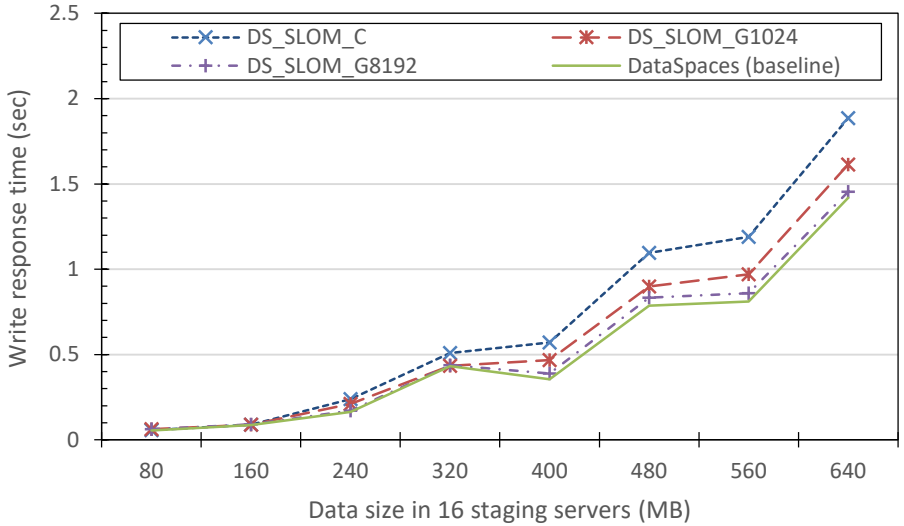
- 18,688 nodes
- Gemini interconnect
- 600 TB system memory
- 16-core AMD Opteron processor, with 32GB DDR3 RAM
- NVIDIA Kepler accelerator, with 6 GB GDDR RAM

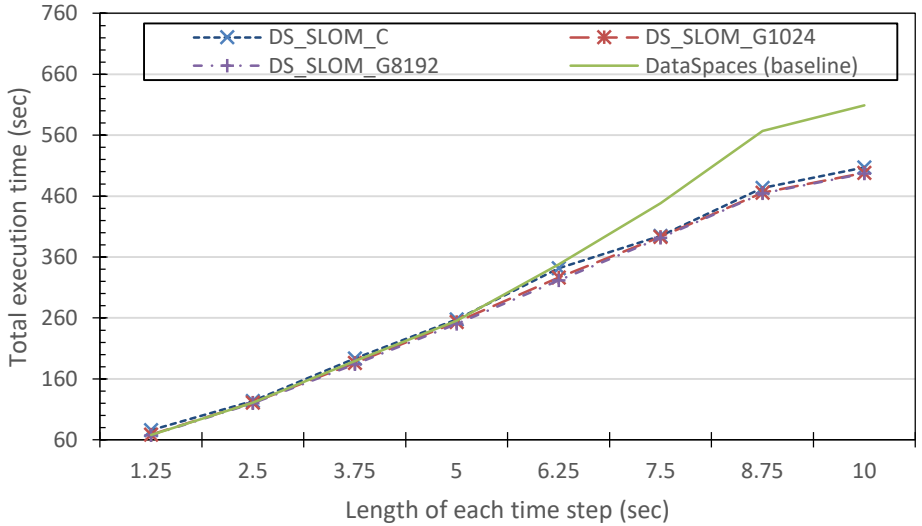
Performance Experiments	
Total number of cores	(16~128) + 16 + 16 = 48~160
No. of parallel writer cores	2 x 4 x 2 ~ 8 x 4 x 4= 16~128
No. of staging cores (node)	16 (1)
No. of parallel reader cores	16
Volume size	64 x 128 x 64 ~ 256 x 128 x 128
In staging data size	80 ~ 640MB
Workflow pattern	write immediately followed by read
Cycle for error detection	1 ts
Synthetic Experiments	
Real Experiments: S3D Workflow	

Baselines: Error detection only on the application node.

Experimental Evaluation

- Performance Experiments





DataSpaces: Data staging only (baseline); DS_SLOM_C: error detection in CPU staging; DS_SLOM_G1024/8192: error detection in Hybrid staging (1024/8192 threads).

- Evaluate write-response time of data staging under different data size (80~640M).
- Evaluate total execution time of workflows under different error detection cycle (1.25~10sec).

Result:

(1) The write response time increases by maximum **24.11**% (CPU), **8.41**% (GPU1024), **2.43**% (GPU8192).

(2) High frequent error detection in staging (less than 5 sec) cannot get benefit.

Experimental Evaluation

- **Synthetic Experiments**: 4 test cases with typical data read/write pattern from real workflow.

Titan, hybrid-architecture Cray XK7

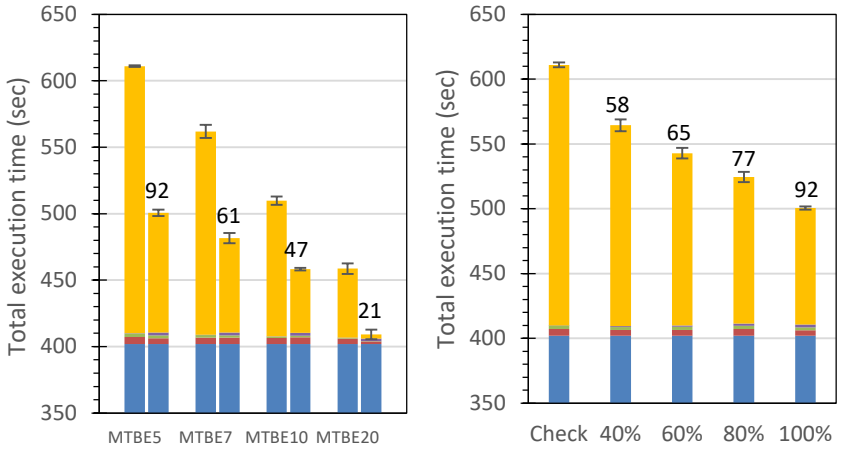
- 18,688 nodes
- Gemini interconnect
- 600 TB system memory
- 16-core AMD Opteron processor, with 32GB DDR3 RAM
- NVIDIA Kepler accelerator, with 6 GB GDDR RAM

Performance Experiments	
Synthetic Experiments	
Total number of cores	96 + 16 + 16 = 128
No. of parallel writer cores	6 x 4 x 4= 96
No. of staging cores (node)	16 (1)
No. of parallel reader cores	16
Volume size	192 x 128 x 128
In staging data size (40 ts)	960MB
Workflow pattern	write immediately followed by read
Checkpoint cycle	10 ts
Real Experiments: S3D Workflow	

Baselines: Error detection only on the application node.

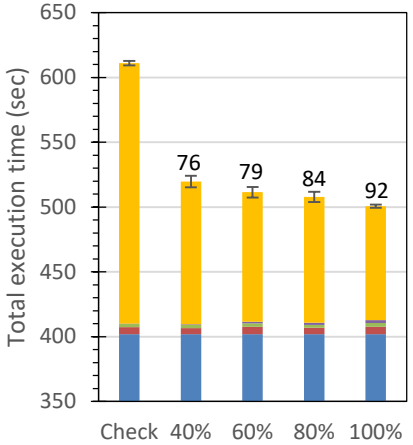
Experimental Evaluation

- Synthetic Experiments

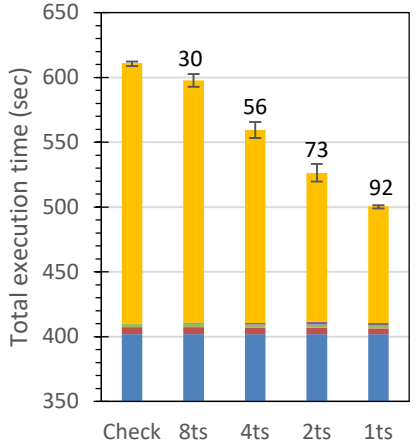


(a) *Case 1*

(b) *Case 2*



(c) *Case 3*



(d) *Case 4*

Total execution time for workflows

Case #	Description
1	Write the entire data domain in each time step under different *MTBE*.
2	Write a subset of the data domain and perform error detection in each time step.
3	Write the entire data domain in multiple time steps and perform error detection in each time step.
4	Write the entire data domain and perform error detection with different time step cycle.

Numbers on top of the bars indicate the detected errors by staging

Result: achieves a decrease of total workflow execution time more than 10.1% in case 1, 7.6% in case 2, 14.9% in case 3, 2.2% in case 4, and get maximum 18.1% decrease in all cases.

Experimental Evaluation

- **Real Experiments**: S3D workflows

Titan, hybrid-architecture Cray XK7

- 18,688 nodes
- Gemini interconnect
- 600 TB system memory
- 16-core AMD Opteron processor, with 32GB DDR3 RAM
- NVIDIA Kepler accelerator, with 6 GB GDDR RAM

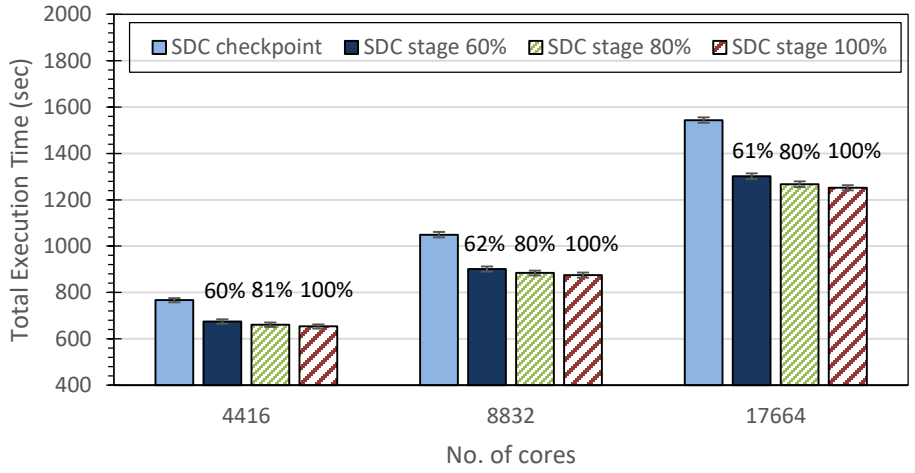
Performance Experiments			
Synthetic Experiments			
Real Experiments: S3D Workflow			
No. of cores	4416	8832	17664
No. of simulation cores	4096	8192	16384
No. of staging cores(nodes)	256(16)	512(32)	1024(64)
No. of analysis cores	64	128	256
Volume size	256x256x256	512x256x256	512x512x256
Data size (GB)	110	220	440
Checkpoint cycle	10 ts	10 ts	10 ts
Detection cycle in staging	1 ts	1 ts	1 ts
MTBE of silent errors	300 sec	300 sec	300 sec

Baselines: Error detection only on the application node.

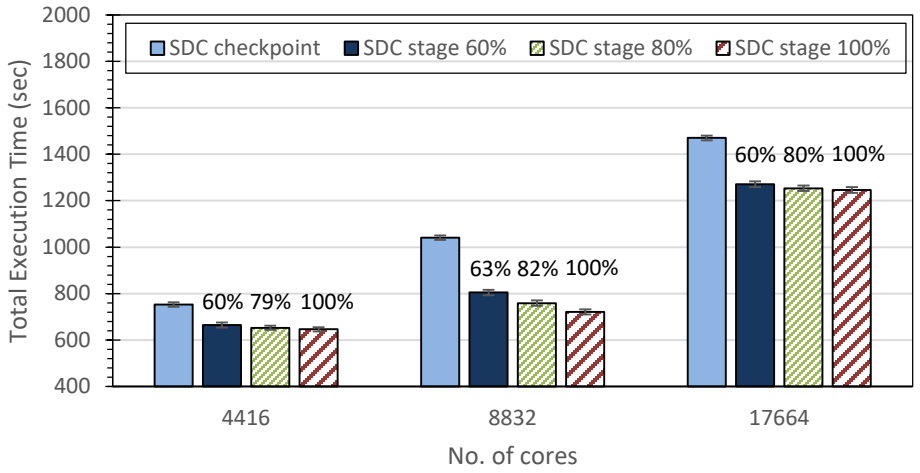
Experimental Evaluation

- Real Experiments: S3D workflows

recall of SDC staging shown on top of the bars



S3D workflow with CPU staging



S3D workflow with Hybrid staging

Result:

(1) Hybrid staging always has better performance than the corresponding CPU staging.

(2) CPU staging reduces the workflow execution time by up to **14.7%**, **16.5%**, and **18.9%** as compared to baseline.

(3) Hybrid staging reduces the workflow execution time by up to **14.1**%, **22.6**%, and **15.4**% as compared to baseline.

Conclusion

- Proposed a staging-based silent errors detection framework that leverages idle computation resource in staging to effectively detect silent errors for in-situ workflows.
  - ✓ The use of dataset/feedback training to achieve the high accuracy for the spatial outlier detection technique.
  - ✓ A CPU-GPU hybrid staging architecture to minimize the impact of error detection on regular I/O operations of data staging.
- Evaluate its effectiveness and performance through performance experiment, synthetic cases experiment and real world large scale S3D workflow experiment.

Future Work

- Investigate the impact of our approach on other classes of application workflows and machines.

- Explore other error detection approaches in staging and evaluate trade-offs for the cost (Ex: energy consumption) /benefit of these approaches on application workflows.

Acknowledgments

This worked was supported by
National Science Foundation (NSF) via grant number CCF-1725649,
Sandia National Laboratories under contract DE-NA-0003525,
Oak Ridge National Laboratory under contract DE-AC05-00OR22725.

Specially thanks to
Dr. Zheng Zhang from Rutgers University
Dr. Hemanth Kolla from Sandia National Laboratories

Thank You!