

# On the Role of Robust Staging Services for Extreme-scale In-Situ Workflows

Manish Parashar

Director, Scientific Computing & Imaging (SCI) Institute

Chair in Computational Science and Engineering

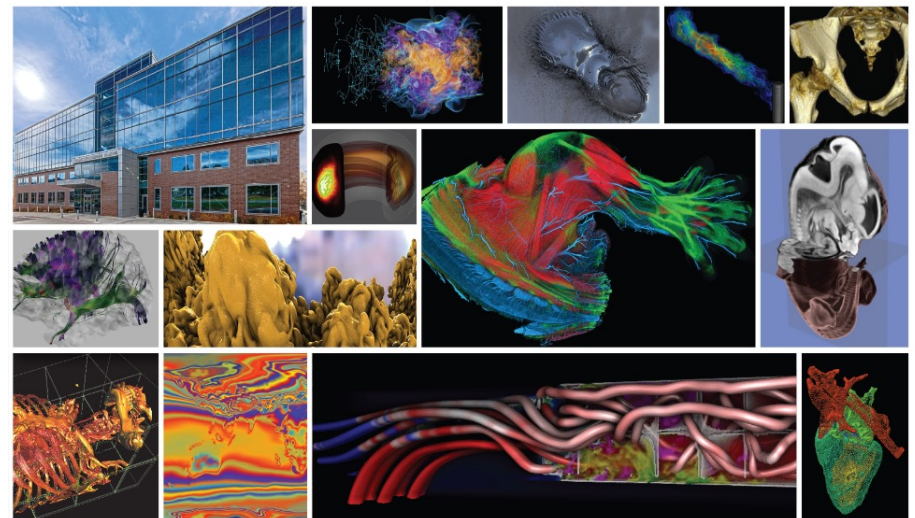
Presidential Professor, Kahlert School of Computing

University of Utah



PASC 2023, Davos, Switzerland  
June 26, 2023

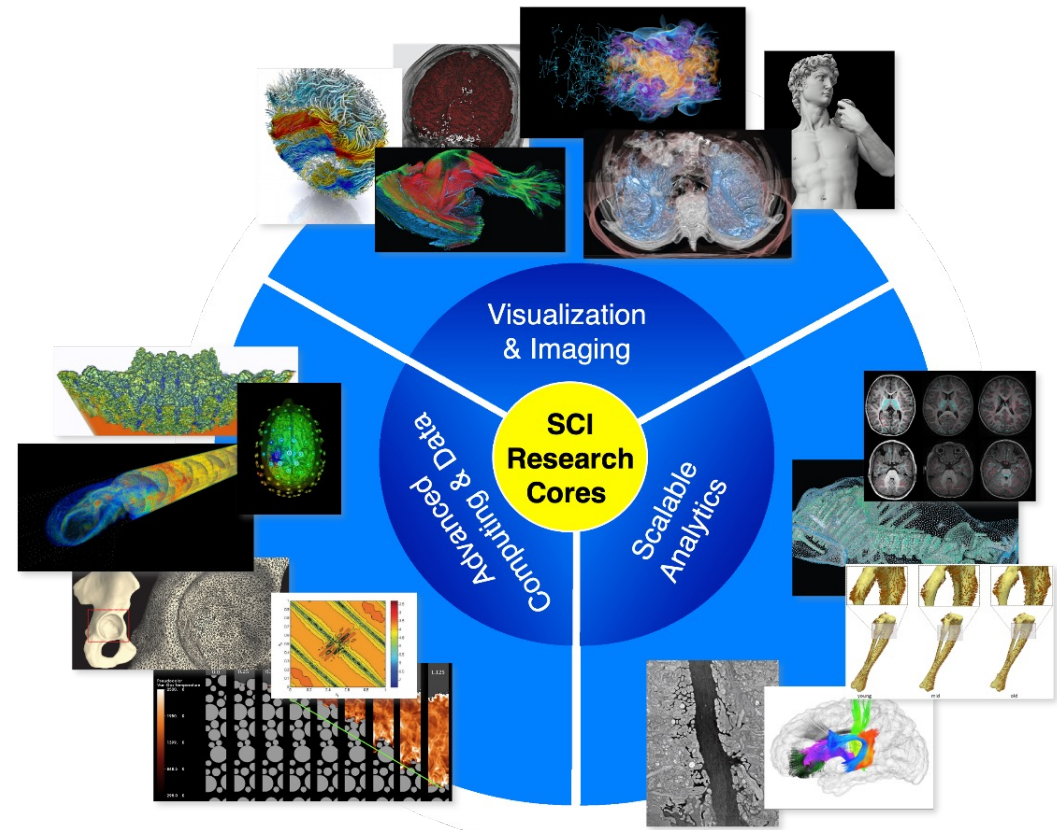
**Shaohua Duan, Pradeep Subedi, Philip E. Davis,  
& Keita Teranishi**



# Scientific Computing & Imaging (SCI) Institute

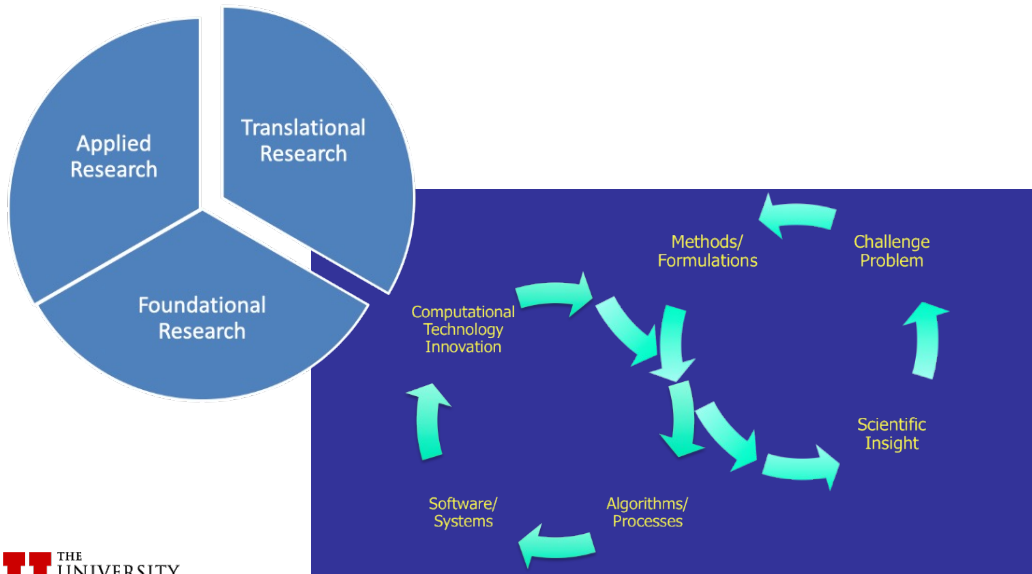
**Goal:** Transformation of science and society through translational research and innovation in computer, computational and data science

- Multidisciplinary, convergent, collaborative
- Simulation, imaging, visualization, data management/analytics, advanced computing
- Software/system development and distribution



# Scientific Computing & Imaging (SCI) Institute

**Goal:** Transformation of science and society through *translational research* and *innovation* in computer, computational and data science

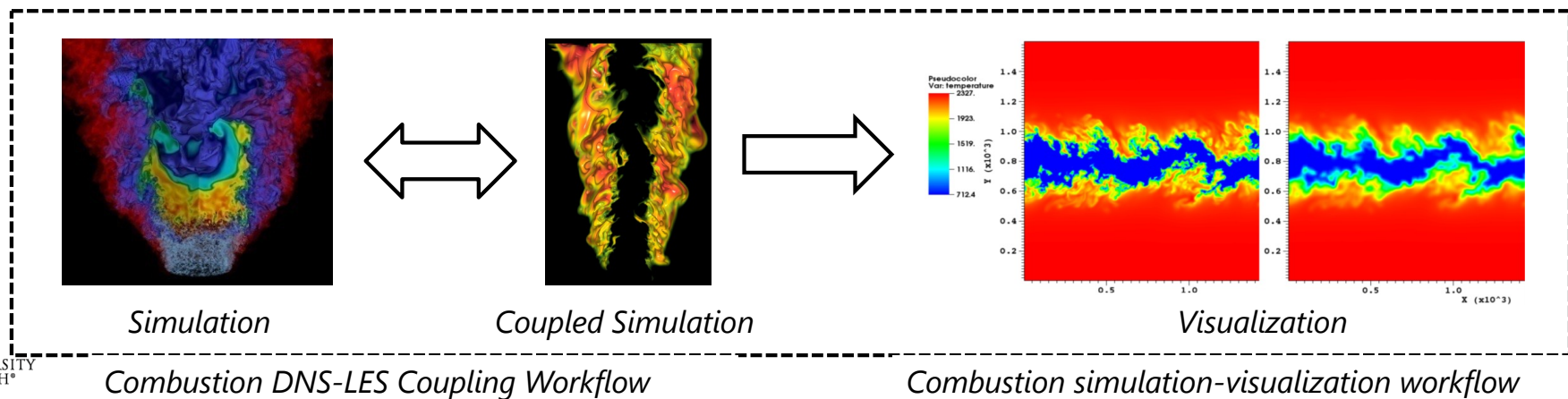


# Outline

- Introduction: In-situ Workflows, Data Staging, and Resilience
- Towards Resilient Staging-based In-Situ Workflows
- CoREC: A Scalable and Resilient In-memory Data Staging
- Conclusion

# Coupled Scientific Workflows at Extreme Scales

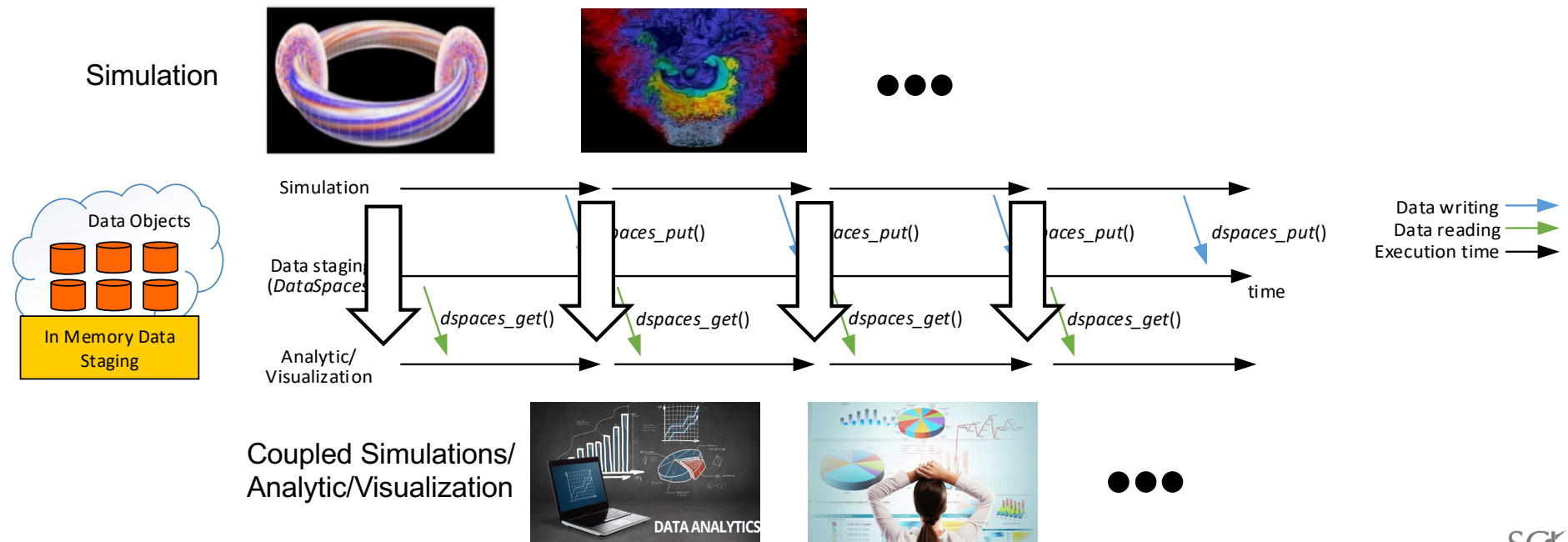
- Advanced scientific simulations running at extreme scale on high end systems generate large amounts of data
  - Transporting and processing data to realize insights is expensive (performance, energy)
- In-situ workflows compose of multiple applications running on the same system that efficiently interact and exchange data at runtime
  - Multi-physics multi-model code coupling (*Combustion DNS-LES*)
  - Online data analysis/visualization (*Combustion simulation-visualization*)



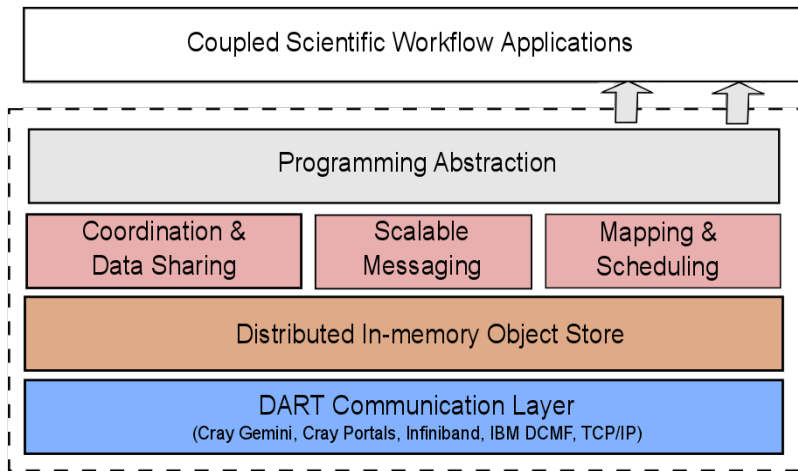


# Staging Based In-Situ Workflows

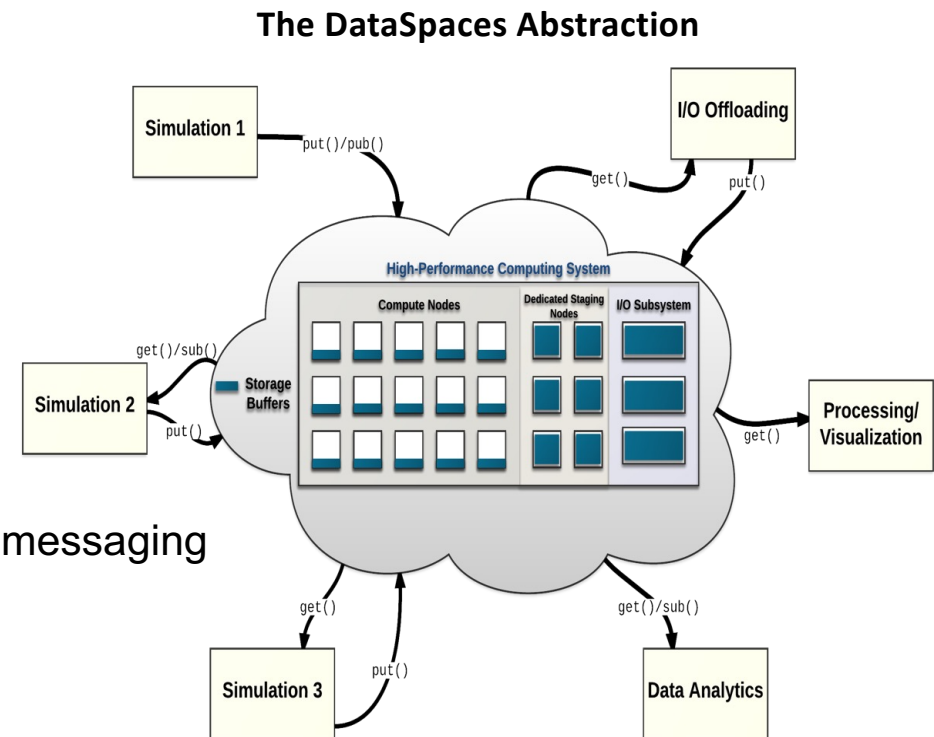
- Data staging techniques provide effective solutions to enable in-situ workflows to efficiently interact and exchange data at runtime
  - In-memory storage distributed across set of cores/nodes
  - Support runtime data processing, sharing and exchange



# DataSpaces: Data Staging Service for In-Situ Workflows



- Virtual shared-space programming abstraction
  - Simple API for coordination, interaction and messaging
- Distributed, associative, in-memory object store
  - Online data indexing, flexible querying
- Autonomic (cross-layer) runtime management
  - Hybrid in-situ/in-transit execution
- High-throughput/low-latency asynchronous data transport



## Failures in Extreme Scale Systems

### ❑ Fail-stop Failure, Silent Errors in Current Systems

- ✓ **Titan**: MTBF = 8 h, the longest period without any failures 24h (2014).
- ✓ **Jaguar** (18688 nodes): silent errors have been observed once per day (2010).
- ✓ **Hopper** (6000 nodes): encounters ~32 FITs per DRAM device (2015).

### ❑ For Extreme Scale Systems

- ✓ The estimated MTBF would be in minutes.

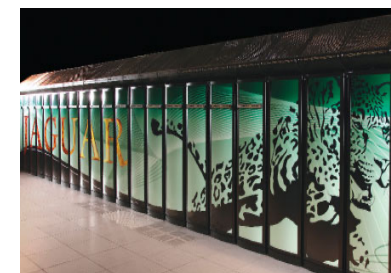
Failure Frequency for Extreme Scale Systems			
MTBF per node	1 year	10 years	100 years
MTBF for 10 <sup>5</sup> nodes system	5.3 min	53 min	9 h
MTBF for 10 <sup>6</sup> nodes system	32 sec	5.3 min	53 min

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 applications.

Data based on available public records in:

D. Tiwari, S. Gupta, S. S. Vazhkudai. “*Lazy checkpointing: Exploiting temporal locality in failures to mitigate checkpointing overheads on extreme-scale systems.*” DSN 2014

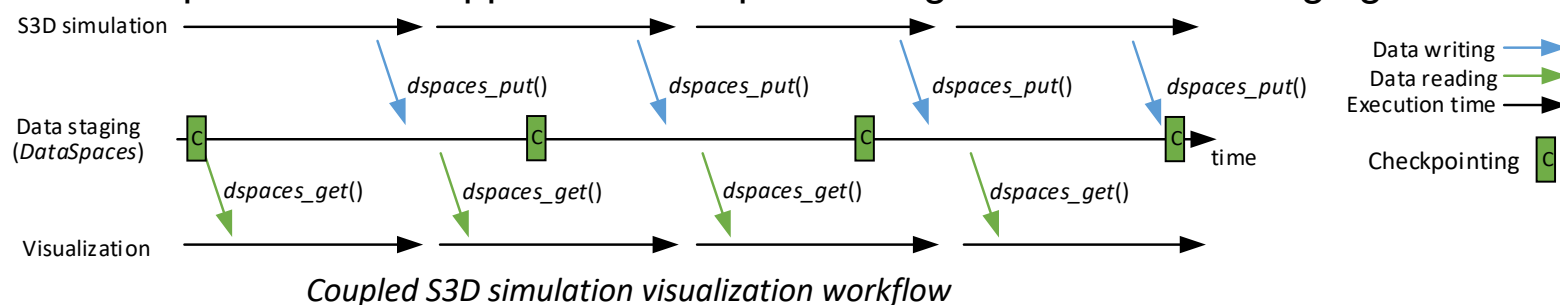
V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi. “Memory errors in modern systems: The good, the bad, and the ugly”. In Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems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





## Data Resilience for Data Staging

### □ Checkpoint/Restart Approach for Implementing Resilient Data Staging



### □ Checkpointing the Data in Data Staging to PFS



### □ Case Study 1:

- Workflow run on the Titan Cray XK7 system.
- Checkpoint 4Gb~32Gb data in data staging to PFS in every 5 mins (total 17~20 times).

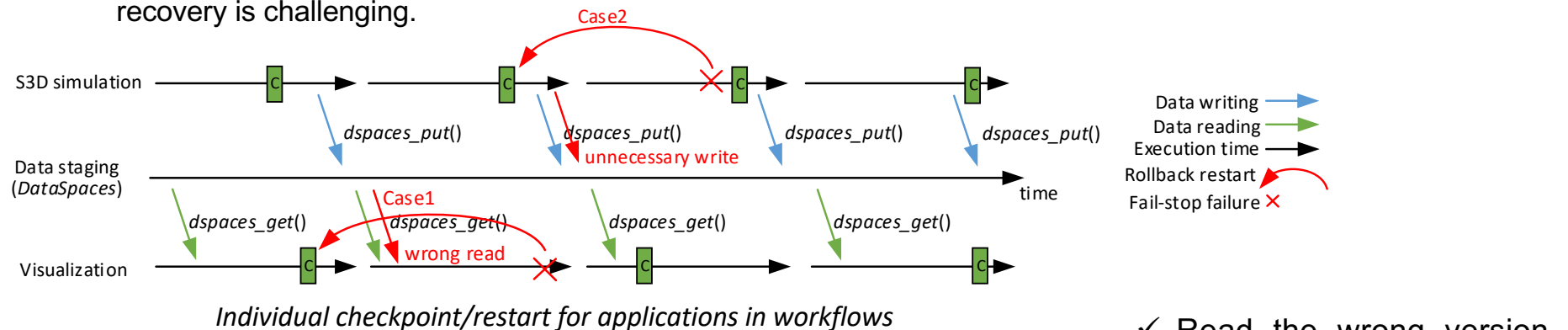
### Observation:

- It took ~ 15.6% of the workflow run-time to achieve fault tolerance for just the staging in the maximum case.

# Failure Recovery for In-situ Workflows

- Crash Consistency

- Coupled applications exchanging large amount of data in extreme scale. To keep data consistency during failure recovery is challenging.



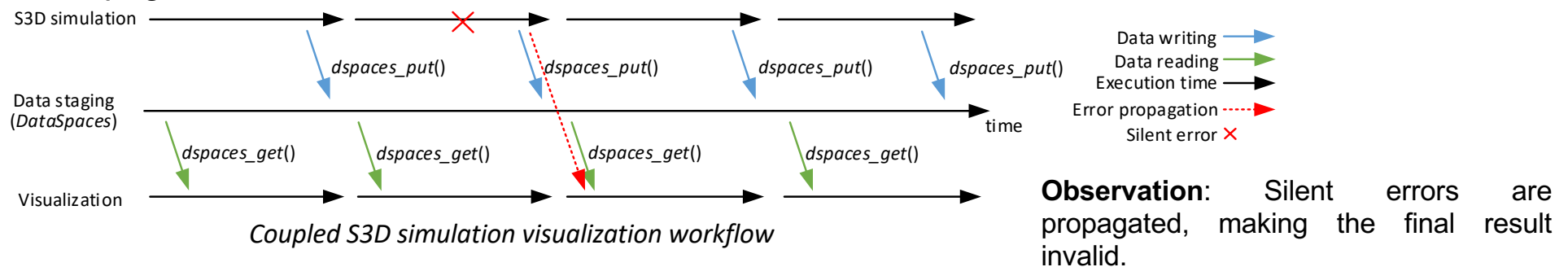
- ✓ Read the wrong version of data (Case 1).
- ✓ Unnecessarily write data twice (Case 2).

- Diversification of Fault Tolerance Strategies

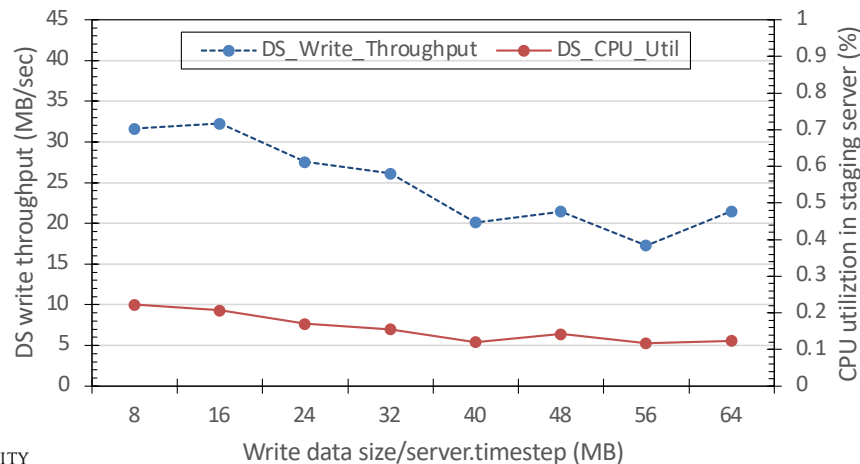
- Allow diversification of fault tolerance strategy among different components (E.g., Process replication, Checkpoint/restart, ABFT).

# Error Detection for In-situ Workflows

## ❑ Propagation of Silent Errors in Workflows



## ❑ Utilizing Idle Compute Resource in Data Staging



## ❑ Case Study 2:

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

## **Observation:**

- CPU utilization remained consistently low
- Maximum CPU utilization 22%

# Resilient In-Situ Workflows: Requirements/Challenges

The final results of the overall computation for workflows is the outputs of the workflow, and failures (fail-stop failures, silent errors) or data inconsistency in any component of the workflow can invalidate these outputs

Requirements/challenges include:

- Managing data resilience in staging with high-performance, low overhead, and minimize the inference for regular data operation of staging
- Providing a general transparent error detection framework for workflows to prevent the propagation of these errors between components
- A loosely coupled fault tolerance mechanism to minimize the inference between components, while still maintaining the consistent states of workflows



Shaohua Duan

# Towards Resilient Staging-based In-Situ Workflows

- Design and implementation of CoREC, a hybrid erasure coding scheme that provides scalable data resilience and failures recovery for data staging.
  - [IPDPS18] “Scalable data resilience for in- memory data staging”, in Proceedings of the 32th IEEE International Parallel and Distributed Processing Symposium (IPDPS’18), pages 105–115, May 2018.
  - [TOPC20] “CoREC: Scalable and Resilient In-Memory Data Staging for In-Situ Workflows”, in International Journal of ACM Transactions on Parallel Computing, May 2020.
- Design and implementation of an in-staging error detection framework that provides data verification for staging based in-situ workflows.
  - [SC’19] “Addressing Data Resiliency for Staging Based Scientific Workflows”, in Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC), 2019 International Conference, November 2019.
- Design and implementation of checkpoint/restart with data logging framework for in-situ scientific workflows that effectively maintain crash consistency during recovery.
  - [HIPS20] “Scalable Crash Consistency for Staging-based In-situ Scientific Workflows”, in 25th Proceedings of the International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS), May 2020.



# CoREC (Combining Replication and Erasure Coding)

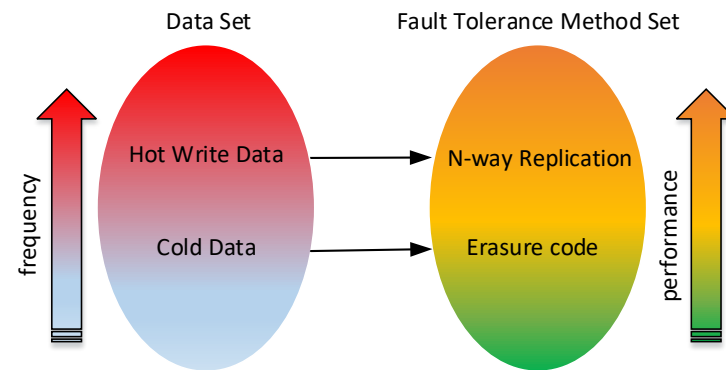
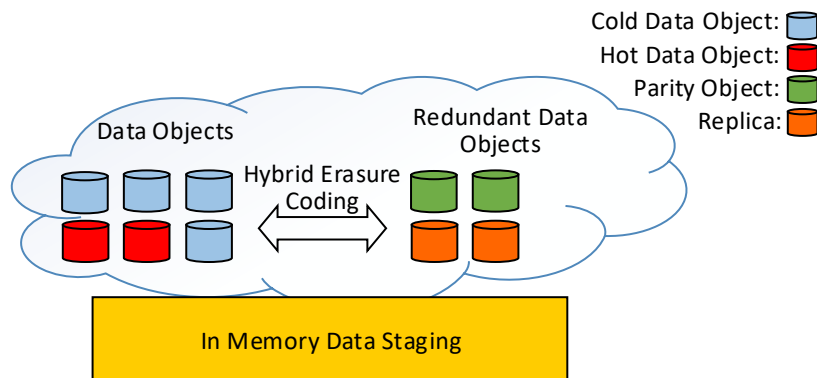
❑ A hybrid approach to data resilience for staging-based workflows

❑ Leverages data classification for intelligent decision making

✓ Spatial/Temporal Data Locality

✓ Hot data → Replication

✓ Cold Data → Erasure Coding



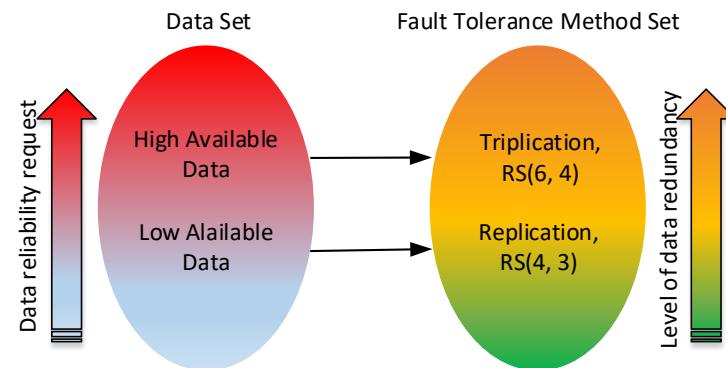
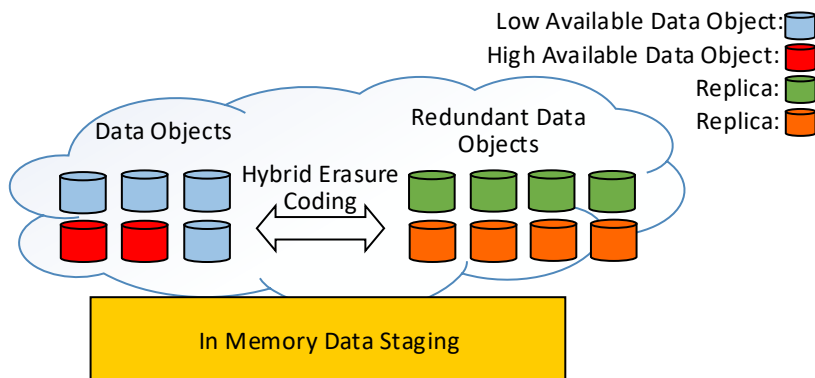
## ❑ Hot/Cold data:

If a data object has been recently accessed more than a number of times within a certain time interval it is considered as **hot data**, otherwise it is considered as **cold data**.

## CoREC-multilevel (CoREC with multilevel data redundancy)

- ❑ Provide different levels of data reliability with an acceptable overall costs and the associated trade-off of achieved resilience, overheads, performance, storage etc.
- ❑ Vary data redundancy scheme (n-way replications and erasure coding schemes) based on the requirements of data resilience level.

- ✓ High reliability data -> Triplication, RS(6, 4)
- ✓ Low reliability data -> Duplication, RS(4, 3)



## Modeling the CoREC / CoREC-multilevel Approach

- A time complexity of CoREC:

$$\begin{aligned}
 C_{CoREC} &= C_r f_h n P_h + C_e f_c n P_c \\
 &= (C_r f_h - C_e f_c) n P_h + C_e f_c n \quad \text{P}_c = 1 - P_h \\
 &= (C_r f_h - C_e f_c + (C_e - C_r) f_h r_m) n P_h + C_e f_c n \quad (C_e - C_r) f_h r_m n P_h
 \end{aligned}$$

$C_r$   $C_e$ : Time Complexity of replication / erasure coding

$f_h$   $f_c$ : Frequency of updates for hot / cold data

$P_h$   $P_c$ : Percentage of hot / cold data

$n$ : The scale of workload

$r_m$ : Miss ratio

- A time complexity of CoREC-multilevel:

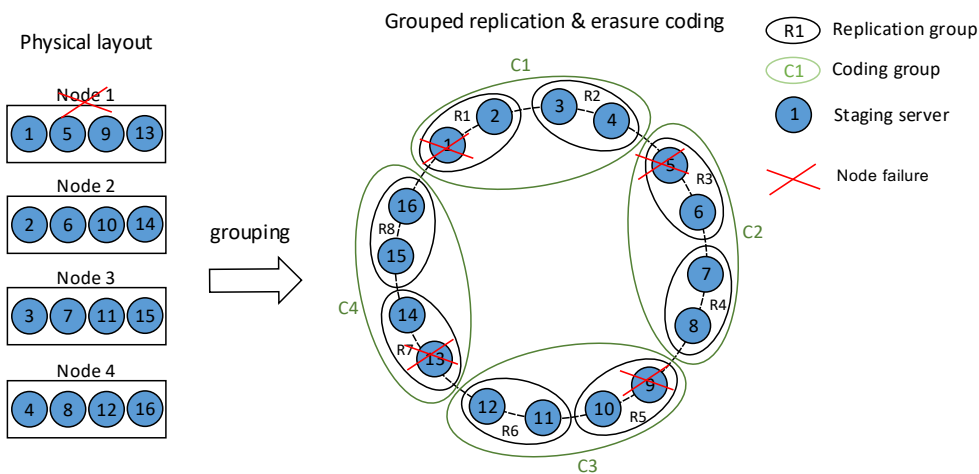
$$C_{CoRECM} = (\widetilde{C}_r f_h - \widetilde{C}_e f_c + (\widetilde{C}_e - \widetilde{C}_r) f_h r_m) n P_h + \widetilde{C}_e f_c n$$

$$\widetilde{C}_r = P_{r1} C_{r1} + P_{r2} C_{r2} + \dots + P_{rn} C_{rn}$$

$$\widetilde{C}_e = P_{e1} C_{e1} + P_{e2} C_{e2} + \dots + P_{en} C_{en}$$

# The System Design of CoREC

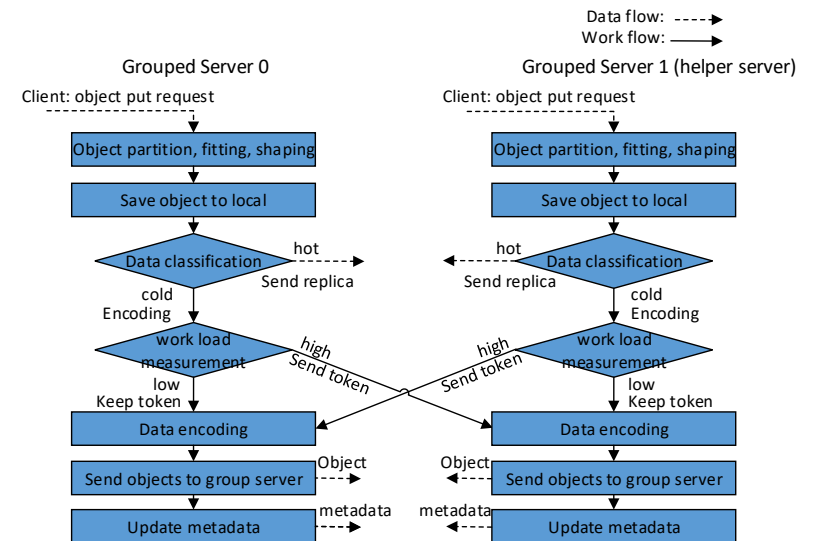
## Grouped Replication & Erasure Coding



Data Objects, Replicas and Parity layout in data staging.  
(replication group size  $k=2$ , Erasure coding group size  $n=4$ ).

**Advantage:** tolerate concurrent correlated staging server failures  
(e.g., Node 1 failure).

## Load balancing and conflict avoid encoding

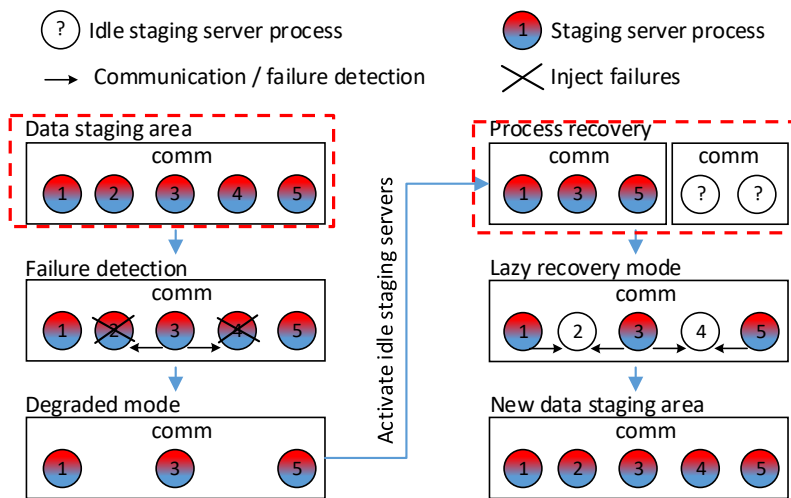


An encoding workflow with 1 server and 1 paired server  
(replication group size = 2).

**Advantage:** keep parity object consistency;  
Balance staging server workload within group.

# The System Design of CoREC

## Recovering Data Staging Server from Failures



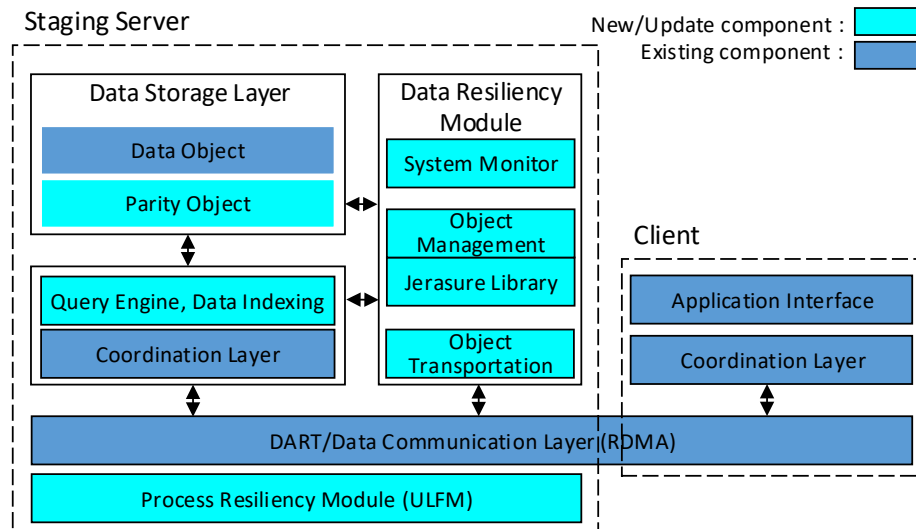
*Data and process recovery in data staging area.*

- ❑ **Failure detection:** Detecting failures by RDMA connection error codes, and handling failures through ULFM-enabled MPI.
- ❑ **Degraded mode:** Only the requested data is re-constructed, sent to the application and discarded.
- ❑ **Process recovery:** The same number of backup staging processes are activated and merged with the existing data staging process group.
- ❑ **Lazy recovery mode:** Each object on the failed server will be recovered immediately after it is queried or updated. The recovery of all other remaining objects are triggered based on the time-limit set for delayed data recovery.

**Advantage:** Alleviate data-recovery overheads and interference with data-reads requests.



# The System Implementation of CoREC



*System Architecture of CoREC.*

## ❑ Local Object Management

- ✓ Local data objects classification and data objects, replicas, parities, metadata's storage.
- ✓ Jerasure open-source library for encoding and decoding.

## ❑ Object Transportation


- ✓ Data objects, replicas, parities, metadata's synchronization and transportation.

## ❑ System Status Monitor

- ✓ Staging server's workload monitoring, failure detection and recovery initiation.

## ❑ Process Resiliency


- ✓ Manages a spare process pool and implements the detection and handling of staging server failures using ULFM.



**Titan, Cray XK7**

- 18,688 nodes
- Gemini interconnect
- 16-core AMD 6200 series Opteron processor
- 32GB memory per node, 600 TB system memory

OF UTAH\*



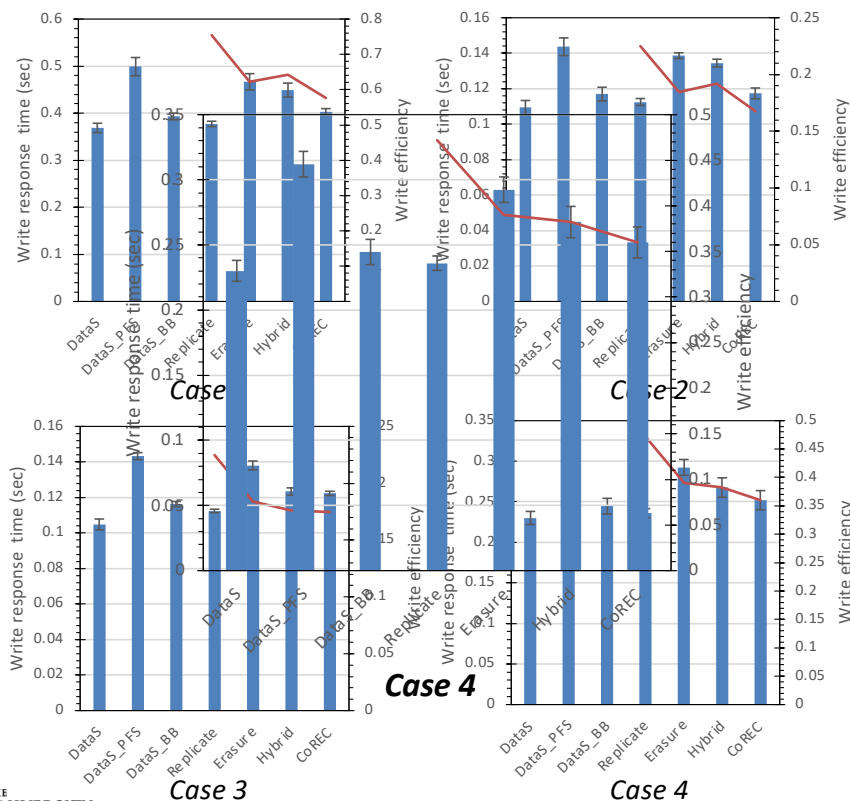
**Cori, Cray XC40**

- 622,336 Cores
- Aries interconnect
- Intel Xeon Phi 7250 68Cores 1.4GHz
- 878,592 GB system memory

SCI  
www.sci.utah.edu

# Experimental Evaluation

❑ **Synthetic Experiments:** 5 cases with data read/write patterns from real scientific workflows.



Case #	Description
1	Write the entire data domain in each time step.
2	Write the entire data domain in multiple time steps.
3	Write a subset of the data domain at a higher frequency than others.
4	Write subsets of the data domain with random access pattern.

*Write efficiency* = *Write response time*/*Storage Efficiency* (lower is better)

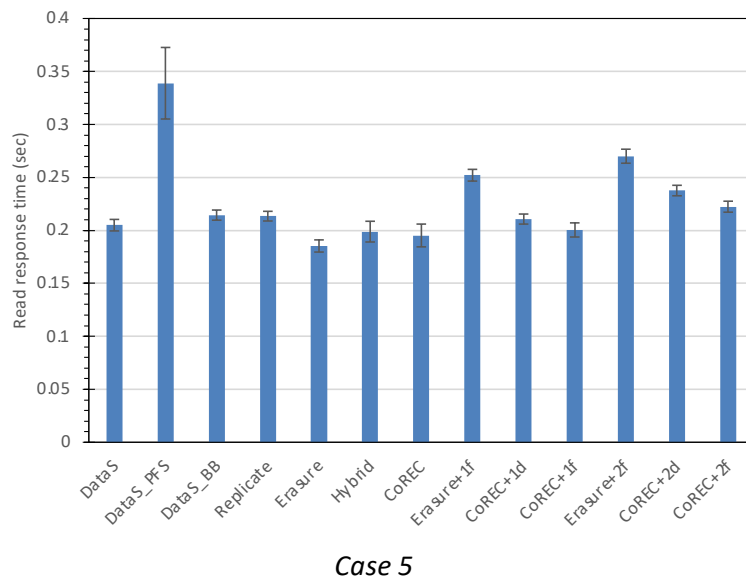
**Baselines:** **DataS:** Data Staging without fault tolerance; **Replicate:** In-memory Replication; **Erasure:** In-memory Erasure coding; **Hybrid:** Simple Hybrid erasure coding with LRU; **\_PFS:** Parallel File System; **\_BB:** Burst Buffer;

**Result:**

- ✓ **CoREC** improves **13.8%**, **5.8%** relative to **Erasure** and **Hybrid** (in Case 4).
- ✓ **CoREC** gets better performance than **Erasure** and **Hybrid** in 4 Cases.

# Experimental Evaluation

❑ **Synthetic Experiments:** 5 cases with data read/write patterns from real scientific workflows.



Case #	Description
5	Read the entire data domain in each time step.

**CoREC+1d or 2d:** in degraded mode with 1 or 2 failures

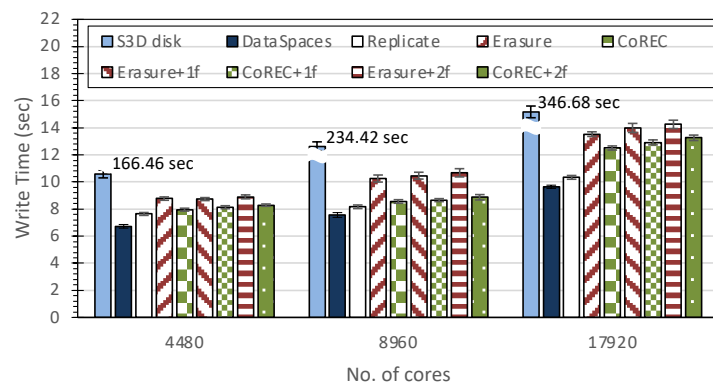
**CoREC+1f or 2f:** in lazy recovery mode with 1 or 2 failures

## Result:

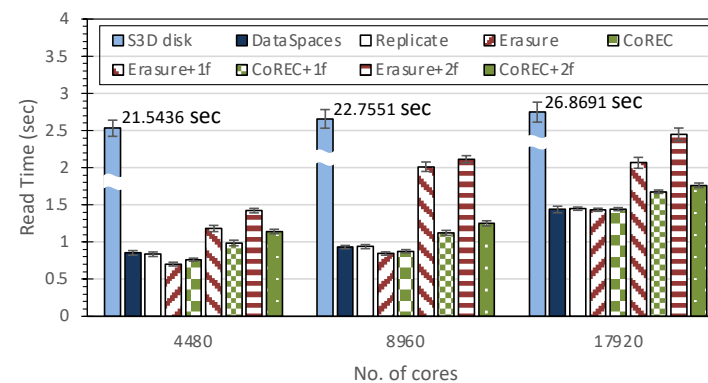
- ✓ Degraded mode: read response time increases by 4.11% (CoREC+1f), 23.4% (CoREC+2f) as compared to failure-free.
- ✓ Lazy recovery: read response time increases only by 2.41% (CoREC+1d), 8.43% (CoREC+2d) as compared to failure-free.
- ✓ **Lazy recovery mode significantly reduces the failure recovery overhead.**

# Experimental Evaluation

## ❑ Real Experiments: S3D workflows



Cumulative write response time



Cumulative read response time

### S3D combustion simulation analysis workflow on Titan Cray XK7

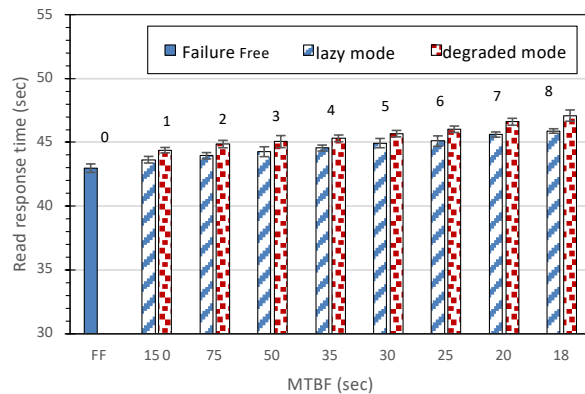
No. of cores	4480	8960	17920
Volume size	1024x1024x1024	2048x1024x1024	2048x2048x1024
Data size (GB)	160	320	640

#### Result:

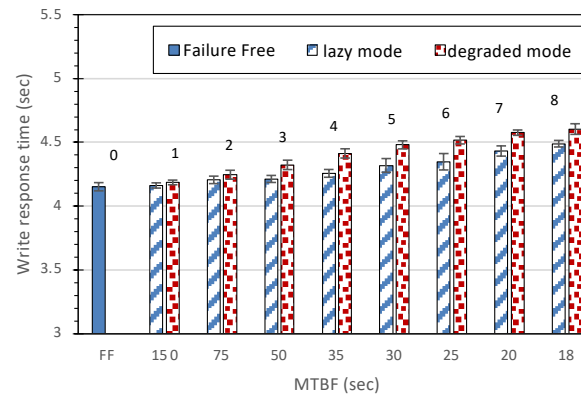
- ✓ Reduces write response time by **7.3%**, **14.8%**, and **5.4%** as compared to full erasure coding on three scales respectively.
- ✓ Reduces read response time by up to **40.8%** and **37.4%** for one and two failures respectively.
- ✓ **CoREC has better performance than full erasure coding in real large scale workflows.**

# Experimental Evaluation

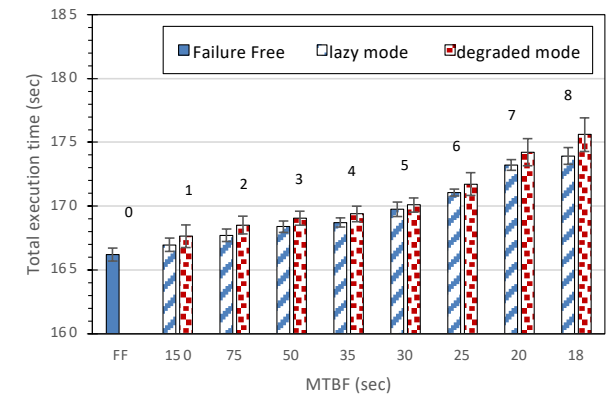
## Node Failures Experiments



Cumulative read response time



Cumulative write response time



Workflow execution time

### Synthetic workflow on Caliburn

Data Size	3.2GB
No. of staging cores (nodes)	256 (32 nodes)
Total number of failures	8 (1 node, MTBF150) ~ 64 (8 nodes, MTBF18)

**Baseline:** FF: CoREC failure free

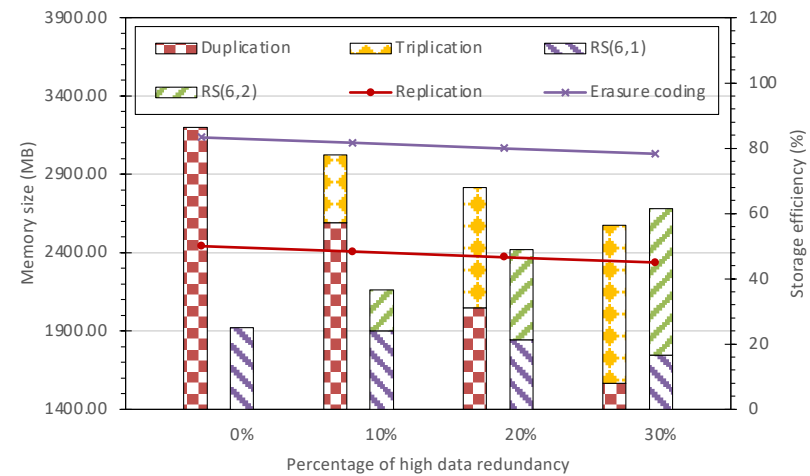
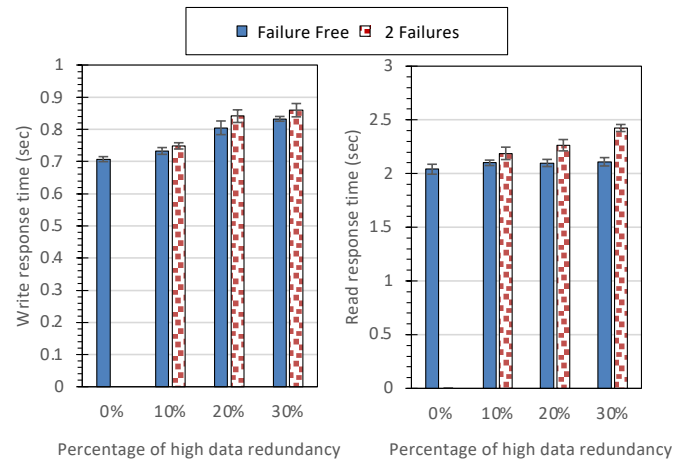
### Result:

- ✓ Increases read response time up to **9.58%** and **6.77%** in degraded mode and lazy recovery mode as compared to baseline.
- ✓ **CoREC can tolerate high frequent process/node failures under light overhead.**



# Experimental Evaluation

## ❑ Multilevel Data Redundancy Experiments



### Synthetic workflow on Titan Cray XK7

Data Size	3.2GB	
Low data redundancy	Duplication	RS(6, 5)
High data redundancy	Triplication	RS(6, 4)

**Baseline:** CoREC with failure free.

### Result:

- ✓ Increase write response time by **2.2%, 4.5%, 3.2%** and read response time by **4.1%, 7.9%, 15.5%** as compared to failure free.
- ✓ Replication cost increase from **3.2Gb** to **2.576Gb**, and the erasure coding cost from **1.92Gb** to **2.683Gb**.

**CoREC-Multilevel provides multiple data redundancy with acceptable storage and computation overhead.**

# Summary

- As HPC systems grow and scale and complexity, the impacts of failures (fail-stop failures, silent errors) or data inconsistencies can significantly impact in-situ workflows.
  - The resiliency of in-situ workflows remains a challenge.
- Addressing resilience for staging-based in-situ workflows:
  - CoREC/CoREC-multilevel, a scalable hybrid approach for data staging frameworks that used online data access classification to effectively combines replication and erasure codes, and to balance computation and storage overheads.
  - A staging-based framework for detecting data corruption that uses idle computation resource to effectively detect silent errors for in-situ workflows.
  - A checkpoint/restart with data logging framework for tight coupled in-situ scientific workflows to enable diverse fault tolerance schemes in workflows, while still maintaining crash consistency.
- Solutions integrated as part of the DataSpaces data-staging service.



# Thank you!

Manish Parashar

Email: [manish.parashar@utah.edu](mailto:manish.parashar@utah.edu)

WWW: [manishparashar.org](http://manishparashar.org)

[dataspaces.org](http://dataspaces.org)



Pradeep Subedi, Philip Davis, Daniel Balouek-Thomert, Zhe Wang,  
Bo Zhang, and **many** students and collaborators