RUTGERS
THE STATE UNIVERSITY
OF NEW JERSEY

Scalable Crash Consistency for Staging-based In-situ Scientific Workflows

Shaohua Duan, Manish Parashar

Rutgers Discovery Informatics Institute

Rutgers, The State University of New Jersey, USA

Outline

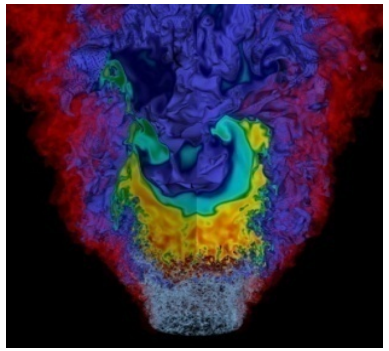
- ❑ Background
  - In-situ Scientific Workflows, Data Staging
  - Fault Tolerance Challenge
- ❑ Data Logging in Staging Framework
- ❑ System Design and Implementation
- ❑ Experimental Overview
- ❑ Conclusion and Future Work

Coupled Scientific Workflows at Extreme Scales

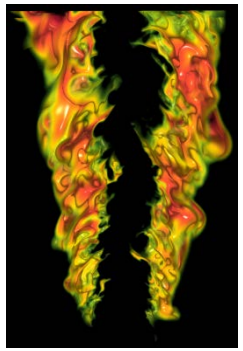
- Advanced scientific simulations running at extreme scale on high end systems generate large amounts of data.
  - ✓ Data must be processed to realize insights.
  - ✓ Dominating costs (performance, energy)
- In-situ scientific workflows are composed of multiple applications that interact and exchange data at runtime.
  - ✓ Multi-physics multi-model code coupling (*Combustion DNS-LES*)
  - ✓ Online data analysis/visualization (*Combustion simulation-visualization*)

Coupled Scientific Workflows at Extreme Scales

- Advanced scientific simulations running at extreme scale on high end systems generate large amounts of data.
  - ✓ Data must be processed to realize insights.
  - ✓ Dominating costs (performance, energy)
- In-situ scientific workflows are composed of multiple applications that interact and exchange data at runtime.
  - ✓ Multi-physics multi-model code coupling (*Combustion DNS-LES*)
  - ✓ Online data analysis/visualization (*Combustion simulation-visualization*)



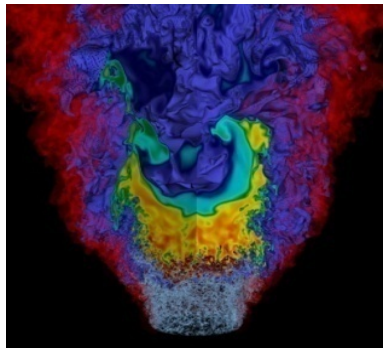
Simulation



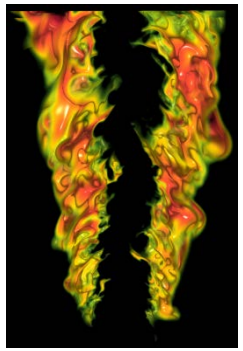
Coupled Simulation

Coupled Scientific Workflows at Extreme Scales

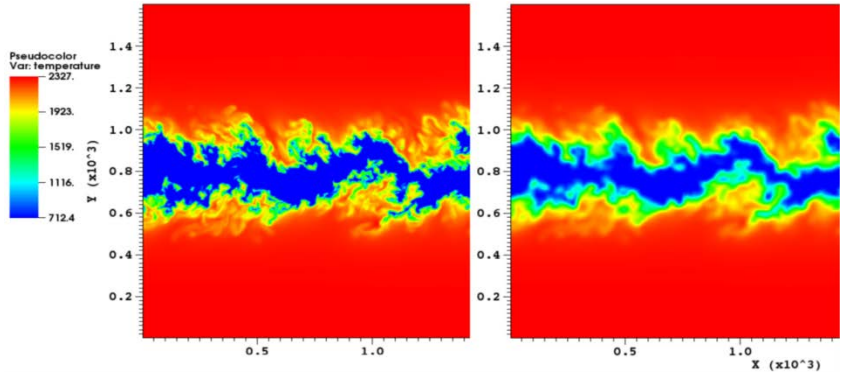
- Advanced scientific simulations running at extreme scale on high end systems generate large amounts of data.
  - ✓ Data must be processed to realize insights.
  - ✓ Dominating costs (performance, energy)
- In-situ scientific workflows are composed of multiple applications that interact and exchange data at runtime.
  - ✓ Multi-physics multi-model code coupling (*Combustion DNS-LES*)
  - ✓ Online data analysis/visualization (*Combustion simulation-visualization*)



Simulation



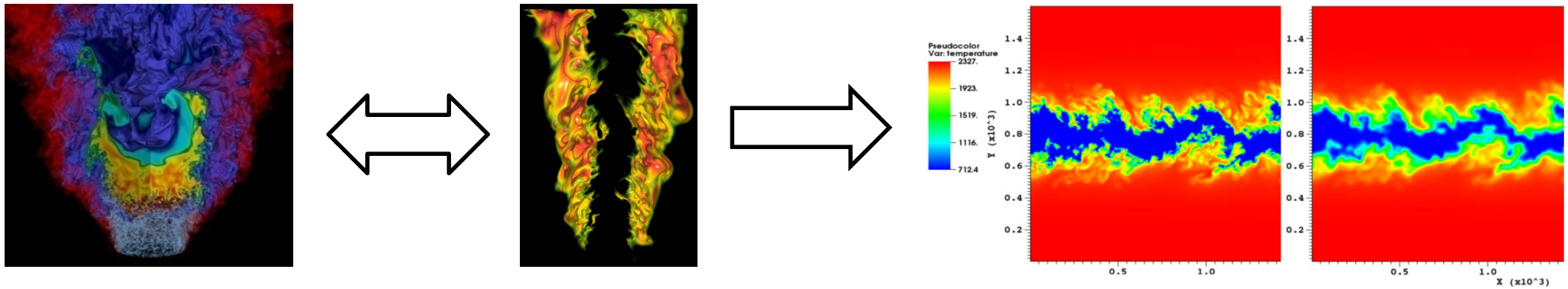
Coupled Simulation



Visualization

Coupled Scientific Workflows at Extreme Scales

- Advanced scientific simulations running at extreme scale on high end systems generate large amounts of data.
  - ✓ Data must be processed to realize insights.
  - ✓ Dominating costs (performance, energy)
- In-situ scientific workflows are composed of multiple applications that interact and exchange data at runtime.
  - ✓ Multi-physics multi-model code coupling (*Combustion DNS-LES*)
  - ✓ Online data analysis/visualization (*Combustion simulation-visualization*)



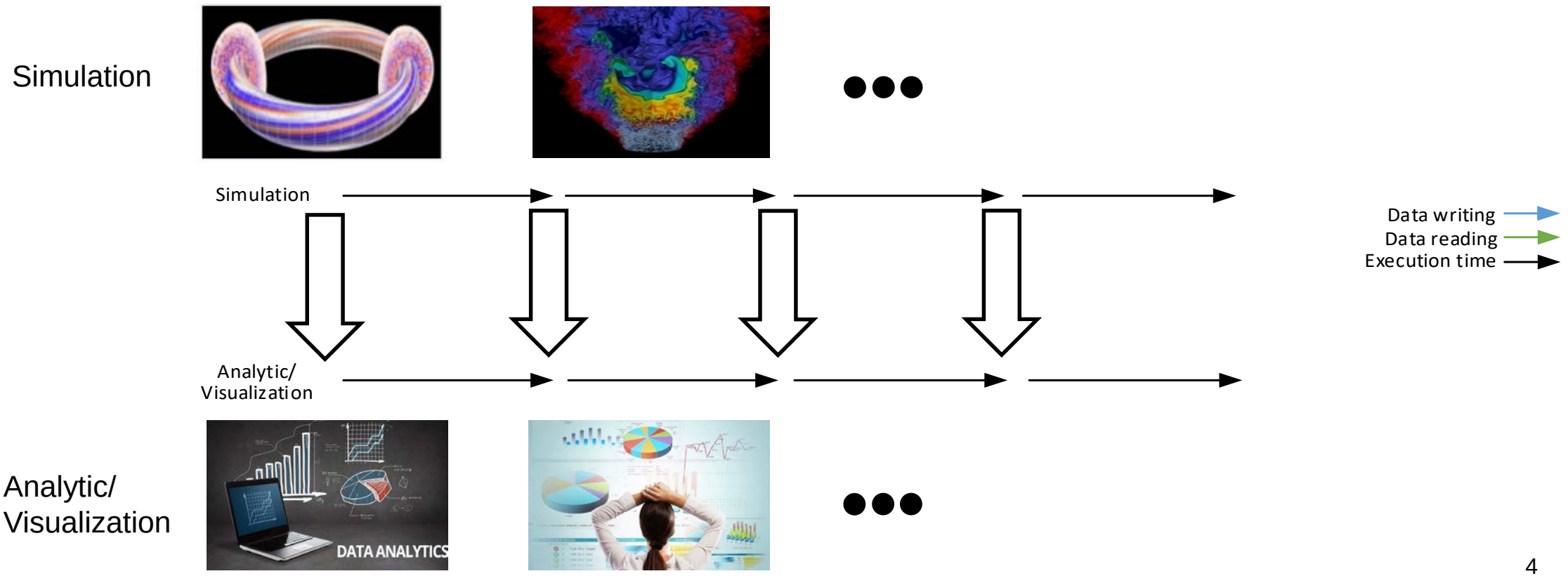
Simulation *Coupled Simulation* *Visualization*

Combustion DNS-LES Coupling Workflow *Combustion simulation-visualization workflow*

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 in scientific workflows.

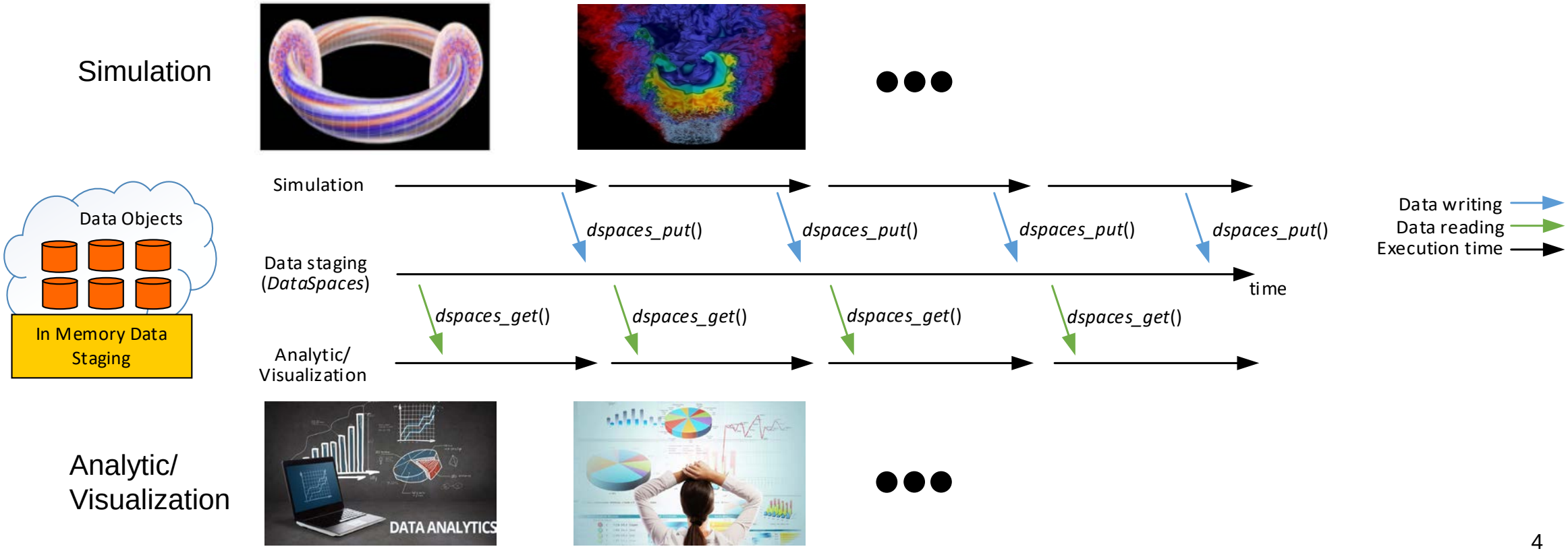
❑ **Characteristics**



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 in scientific workflows.

❑ **Characteristics**



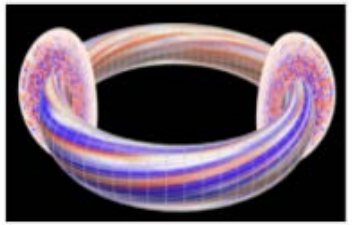
Staging Based Workflows

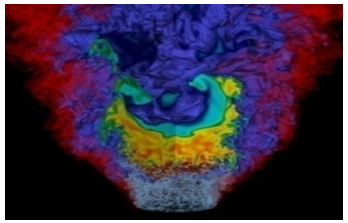
Data staging techniques have emerged as effective solutions for addressing data related challenges such as I/O storage challenge and data movement challenge in scientific workflows.

❑ Characteristics

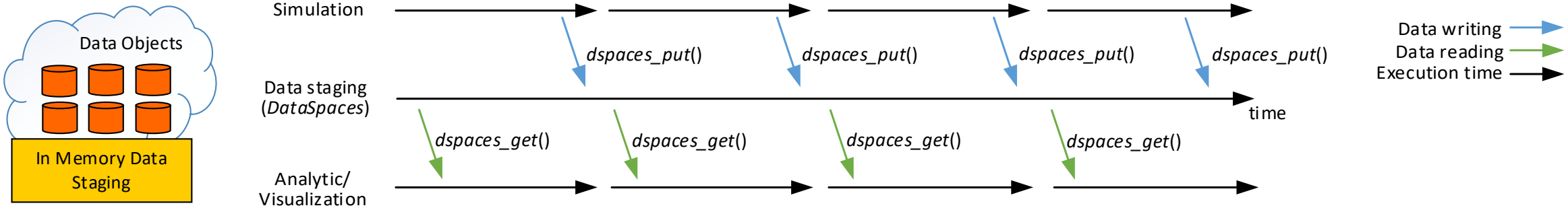
- ✓ In-memory storage distributed across set of cores/nodes
- ✓ Support runtime data processing, sharing and exchange

Simulation





●●●



Analytic/ Visualization

●●●

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 10^5 nodes system	5.3 min	53 min	9 h
MTBF for 10^6 nodes system	32 sec	5.3 min	53 min

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.

Fault Tolerance in In-situ Scientific Workflows

- **Challenge One: Crash Consistency**

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

Fault Tolerance in In-situ Scientific Workflows

❑ Challenge One: Crash Consistency

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

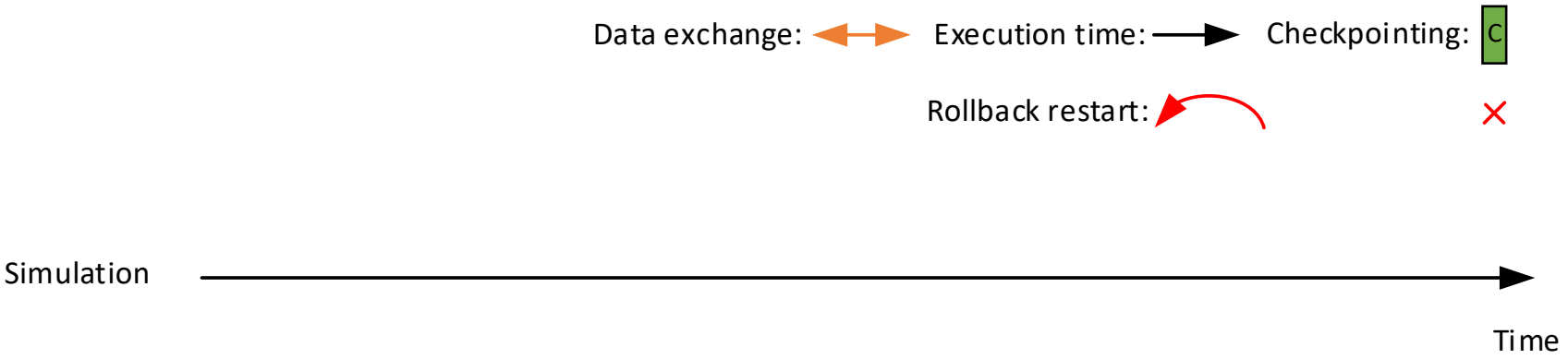


Figure 1: Individual checkpoint / restart for applications in workflows

Fault Tolerance in In-situ Scientific Workflows

❑ Challenge One: Crash Consistency

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

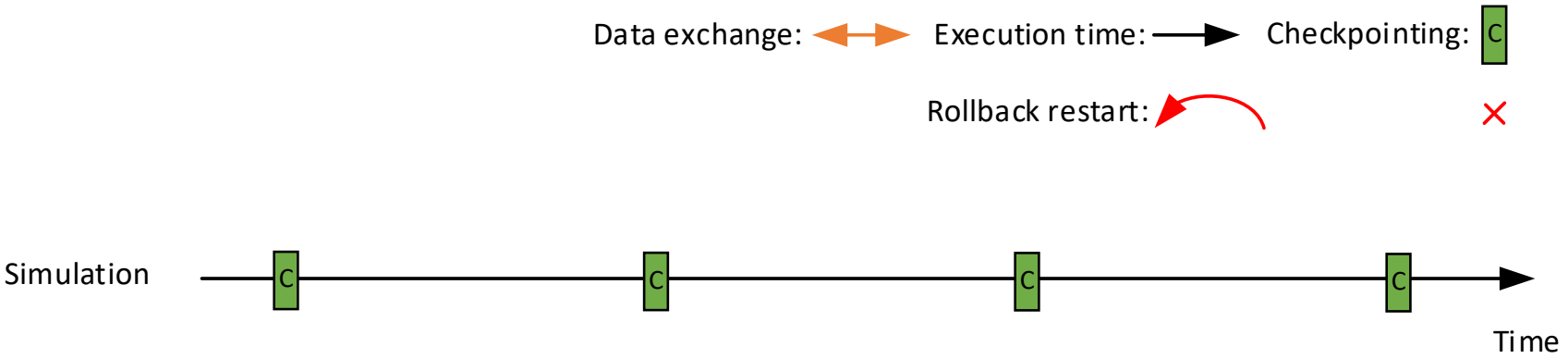


Figure 1: Individual checkpoint / restart for applications in workflows

Fault Tolerance in In-situ Scientific Workflows

❑ Challenge One: Crash Consistency

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

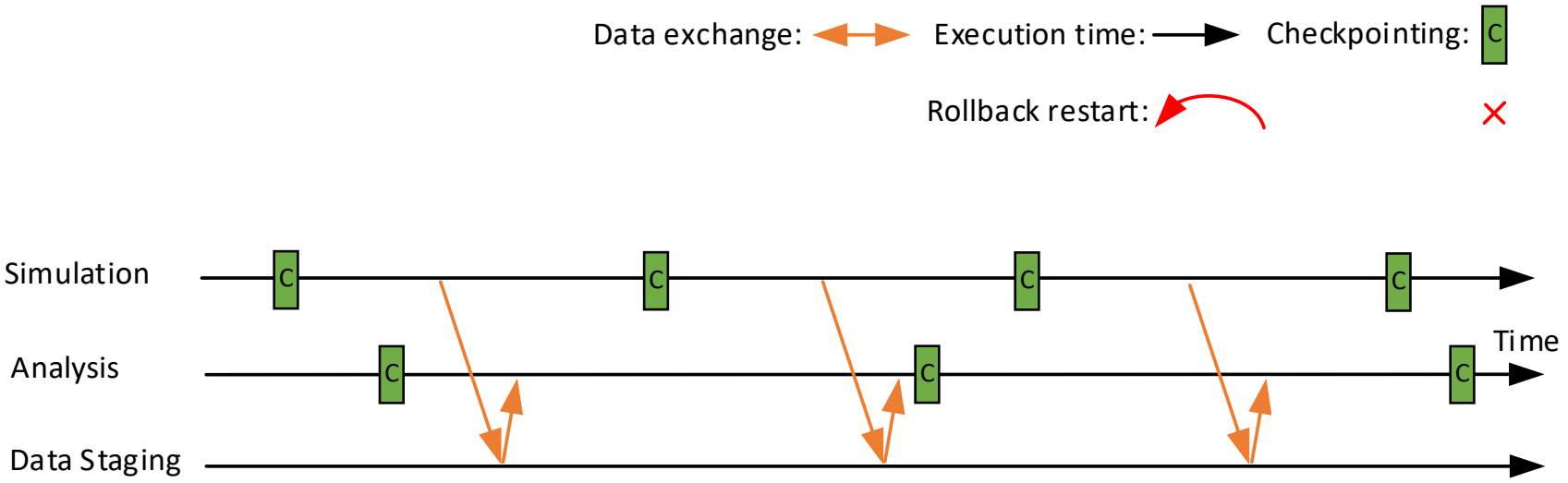
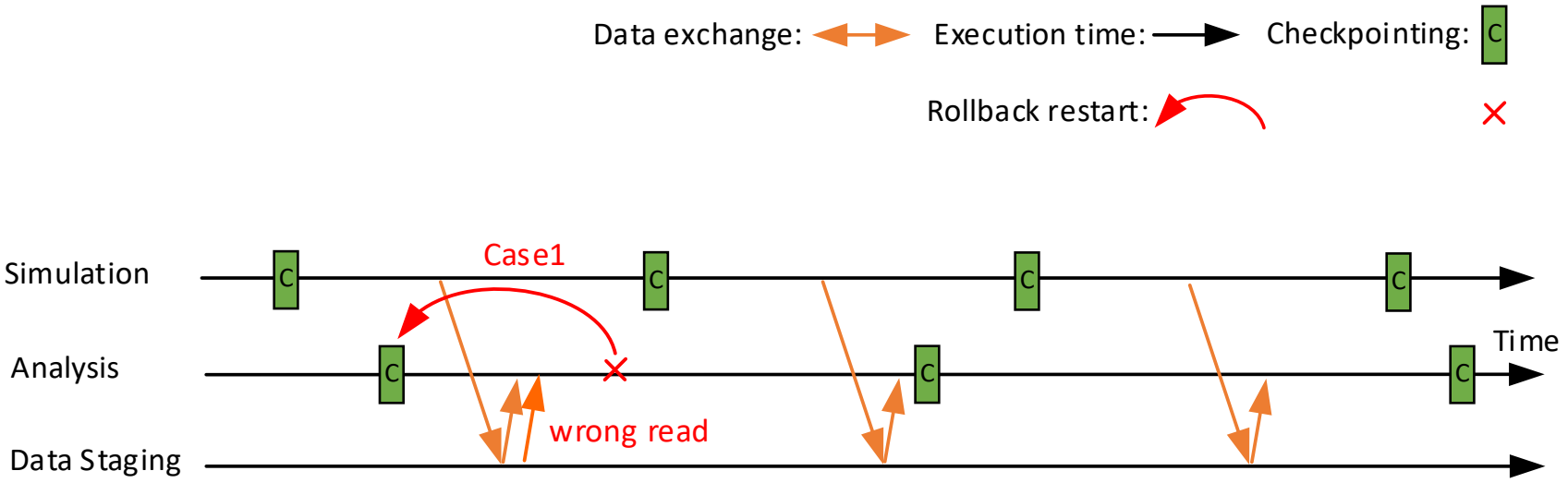


Figure 1: Individual checkpoint / restart for applications in workflows

Fault Tolerance in In-situ Scientific Workflows

❑ Challenge One: Crash Consistency

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



✓ Read the wrong version of data (Case 1).

Figure 1: Individual checkpoint / restart for applications in workflows

Fault Tolerance in In-situ Scientific Workflows

❑ Challenge One: Crash Consistency

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

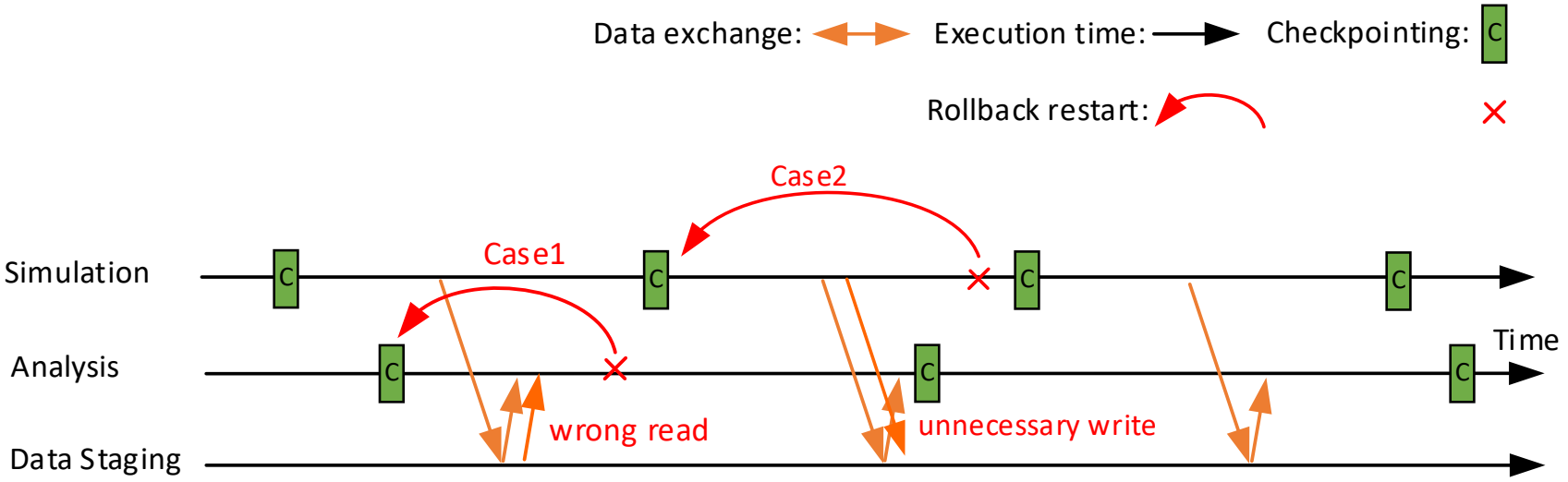


Figure 1: Individual checkpoint / restart for applications in workflows

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

Fault Tolerance in In-situ Scientific Workflows

- **Challenge Two: Diversification of Fault Tolerance Strategies**

Application components in workflows exhibit different resiliency requirements, program properties and failure characteristics, which result in diversification of fault tolerance strategies:

Fault Tolerance in In-situ Scientific Workflows

❑ Challenge Two: Diversification of Fault Tolerance Strategies

Application components in workflows exhibit different resiliency requirements, program properties and failure characteristics, which result in diversification of fault tolerance strategies:

- ✓Process Replication: Duplication, Triplication, Partial replication …

Fault Tolerance in In-situ Scientific Workflows

❑ Challenge Two: Diversification of Fault Tolerance Strategies

Application components in workflows exhibit different resiliency requirements, program properties and failure characteristics, which result in diversification of fault tolerance strategies:

- ✓Process Replication: Duplication, Triplication, Partial replication …
- ✓Checkpoint/Restart: Coordinated, Uncoordinated, Local recovery, Multi-level, Proactive …

Fault Tolerance in In-situ Scientific Workflows

❑ Challenge Two: Diversification of Fault Tolerance Strategies

Application components in workflows exhibit different resiliency requirements, program properties and failure characteristics, which result in diversification of fault tolerance strategies:

- ✓Process Replication: Duplication, Triplication, Partial replication …
- ✓Checkpoint/Restart: Coordinated, Uncoordinated, Local recovery, Multi-level, Proactive …
- ✓ABFT: Online, Offline …
- ✓…

Fault Tolerance in In-situ Scientific Workflows

❑ Challenge Two: Diversification of Fault Tolerance Strategies

Application components in workflows exhibit different resiliency requirements, program properties and failure characteristics, which result in diversification of fault tolerance strategies:

- ✓Process Replication: Duplication, Triplication, Partial replication …
- ✓Checkpoint/Restart: Coordinated, Uncoordinated, Local recovery, Multi-level, Proactive …
- ✓ABFT: Online, Offline …
- ✓…

Ideally, workflow fault tolerance framework should enable individual components to employ the wide area of fault tolerance approaches, while maintaining crash consistency.

Fault Tolerance in In-situ Scientific Workflows

❑ Global Coordinated Checkpoint/Restart

This approach perform checkpointing for application components of workflows coordinately, and in case of failure, it requires that all applications in workflow rollback to the last valid checkpoint place.

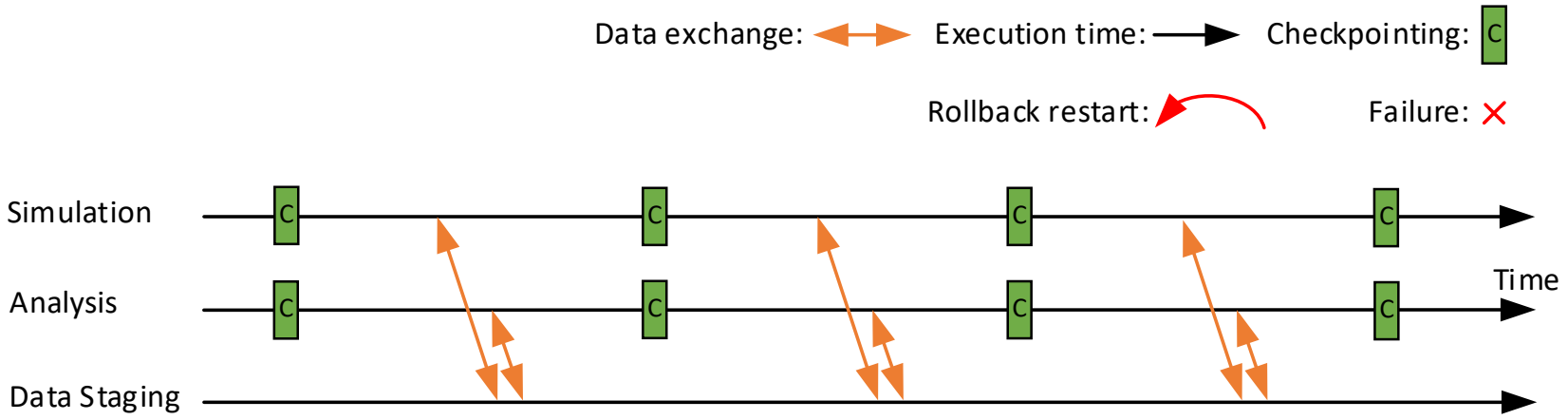


Figure 2: Coordinated checkpointing + Global recovery for entire workflows

Fault Tolerance in In-situ Scientific Workflows

❑ Global Coordinated Checkpoint/Restart

This approach perform checkpointing for application components of workflows coordinately, and in case of failure, it requires that all applications in workflow rollback to the last valid checkpoint place.

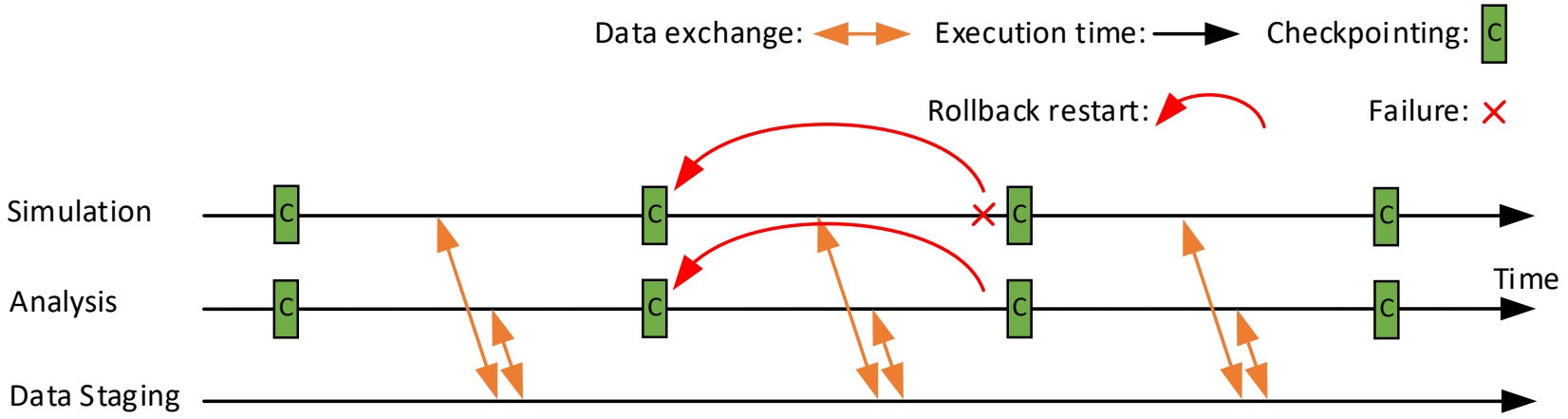


Figure 2: Coordinated checkpointing + Global recovery for entire workflows

✓ High cost: not be scalable.

Fault Tolerance in In-situ Scientific Workflows

❑ Global Coordinated Checkpoint/Restart

This approach perform checkpointing for application components of workflows coordinately, and in case of failure, it requires that all applications in workflow rollback to the last valid checkpoint place.

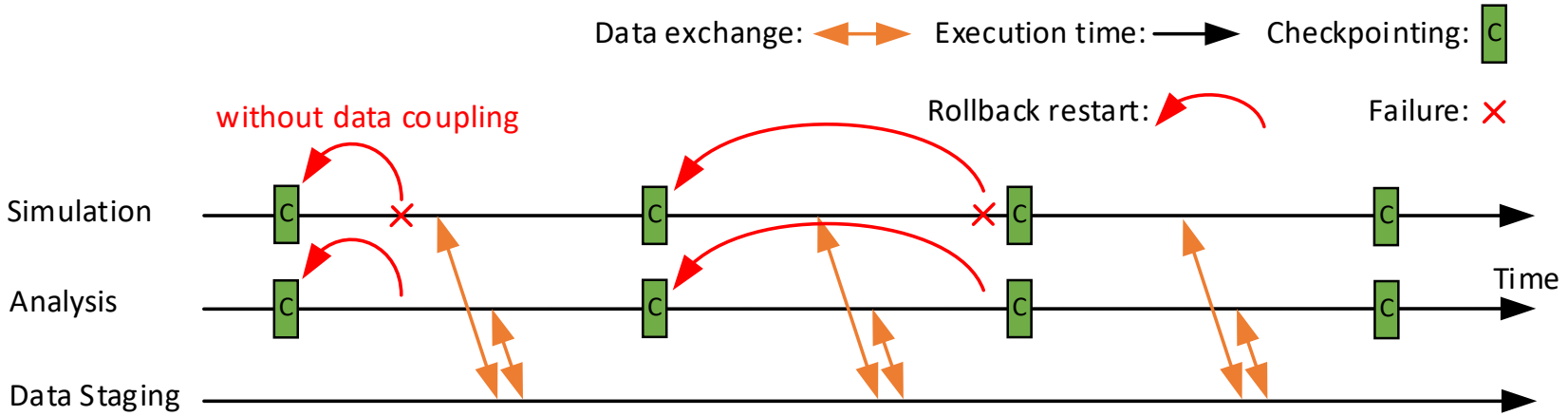


Figure 2: Coordinated checkpointing + Global recovery for entire workflows

- ✓ High cost: not be scalable.
- ✓ Wasteful: unnecessary to rollback healthy applications.

Fault Tolerance in In-situ Scientific Workflows

❑ Global Coordinated Checkpoint/Restart

This approach perform checkpointing for application components of workflows coordinately, and in case of failure, it requires that all applications in workflow rollback to the last valid checkpoint place.

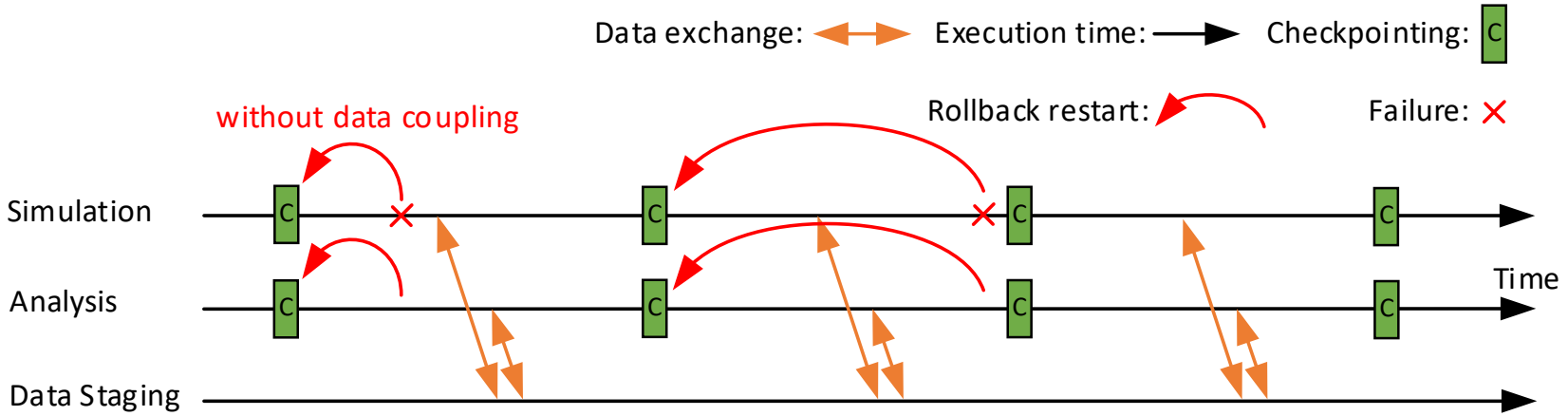


Figure 2: Coordinated checkpointing + Global recovery for entire workflows

- ✓ High cost: not be scalable.
- ✓ Wasteful: unnecessary to rollback healthy applications.
- ✓ Not support diversity of resilience approaches.

Checkpoint/Restart with Data Logging

❑ Data Logging

Logging the data transportation events and payloads between application components as it proceeds along the initial execution; In case of failure, replaying the log history, enforcing all data transportation events of the failed component to produce the same effect they had during the initial execution.

Checkpoint/Restart with Data Logging

❑ Data Logging

Logging the data transportation events and payloads between application components as it proceeds along the initial execution; In case of failure, replaying the log history, enforcing all data transportation events of the failed component to produce the same effect they had during the initial execution.

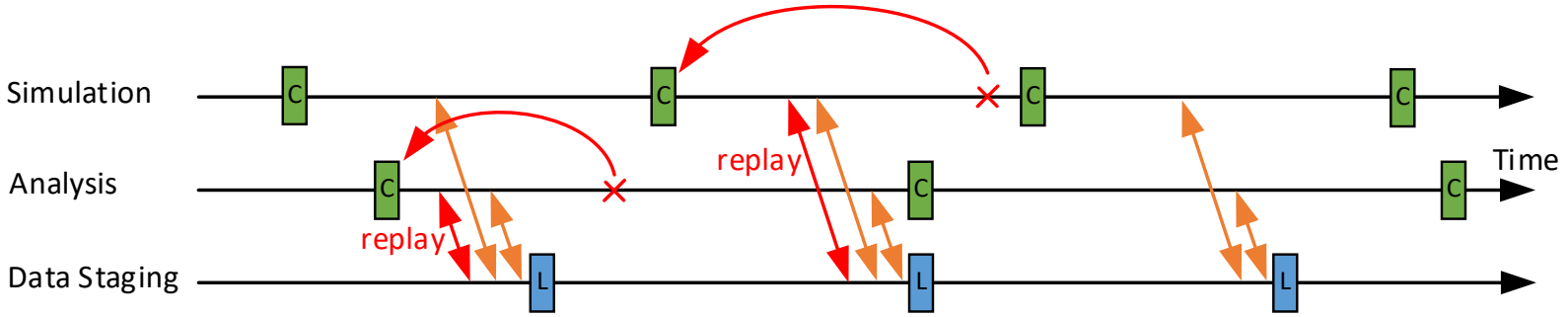


Figure 3: Uncoordinated checkpointing with data logging in staging

- ✓ Perform data logging in the staging area.
- ✓ Maintain data consistency among coupled application components.
- ✓ Compatible with diverse fault tolerance approaches.

Checkpoint/Restart with Data Logging

❑ Data Logging

Logging the data transportation events and payloads between application components as it proceeds along the initial execution; In case of failure, replaying the log history, enforcing all data transportation events of the failed component to produce the same effect they had during the initial execution.

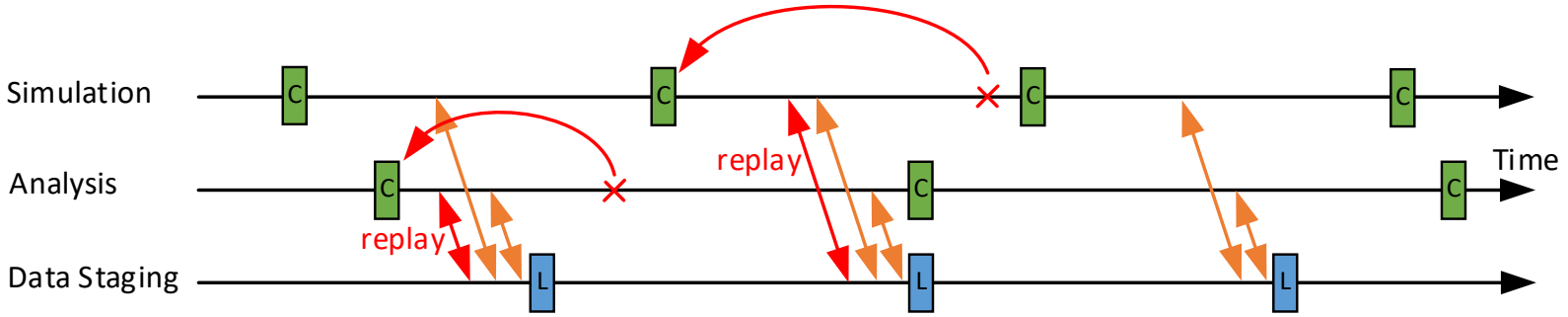


Figure 3: Uncoordinated checkpointing with data logging in staging

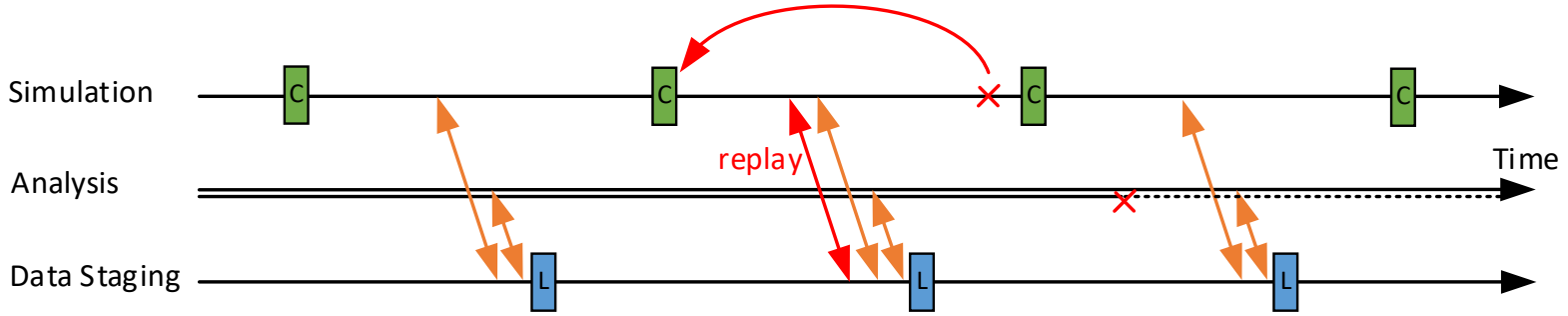


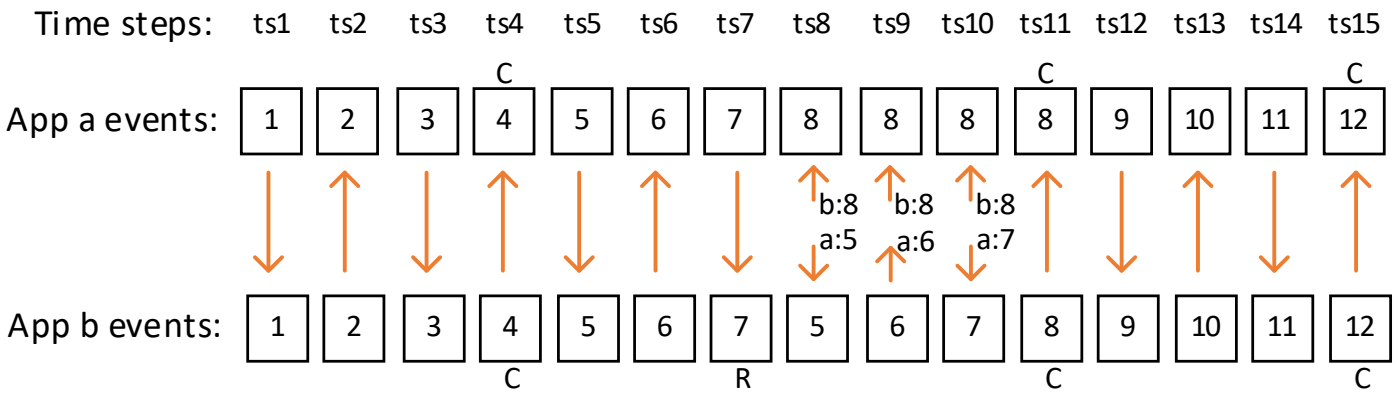
Figure 4: Hybrid Checkpointing (checkpoint/restart + process replication)

- ✓ Perform data logging in the staging area.
- ✓ Maintain data consistency among coupled application components.
- ✓ Compatible with diverse fault tolerance approaches.

Checkpoint/Restart with Data Logging

❑ Data Logging in Staging Area

A queue based algorithm in staging area to keep data/event consistency.



G: Data Get
P: Data Put
C: Checkpointing
R: Rollback

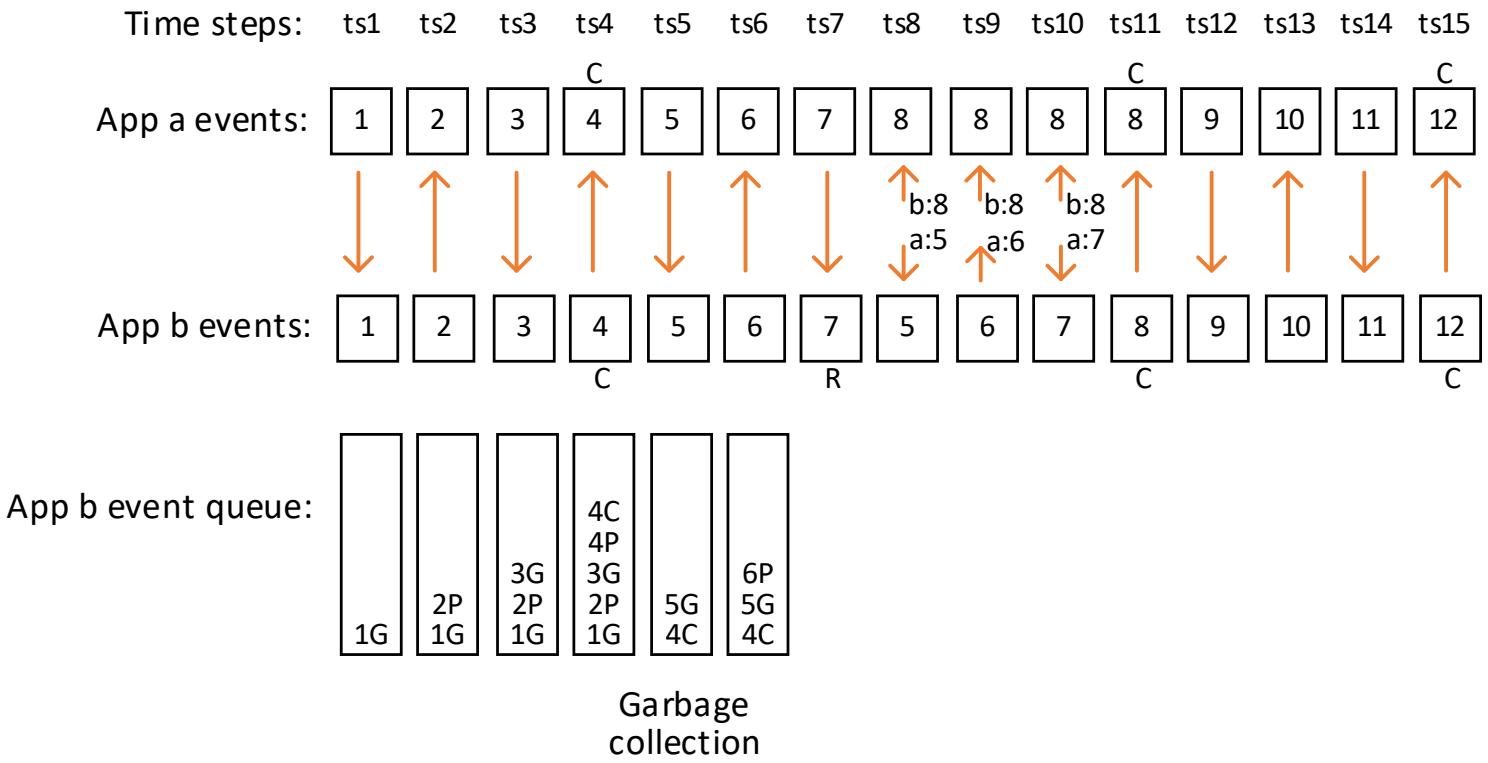
App b event queue:

Figure 5: An illustration of queue based data consistency algorithm for workflows.

Checkpoint/Restart with Data Logging

❑ Data Logging in Staging Area

A queue based algorithm in staging area to keep data/event consistency.



G: Data Get
P: Data Put
C: Checkpointing
R: Rollback

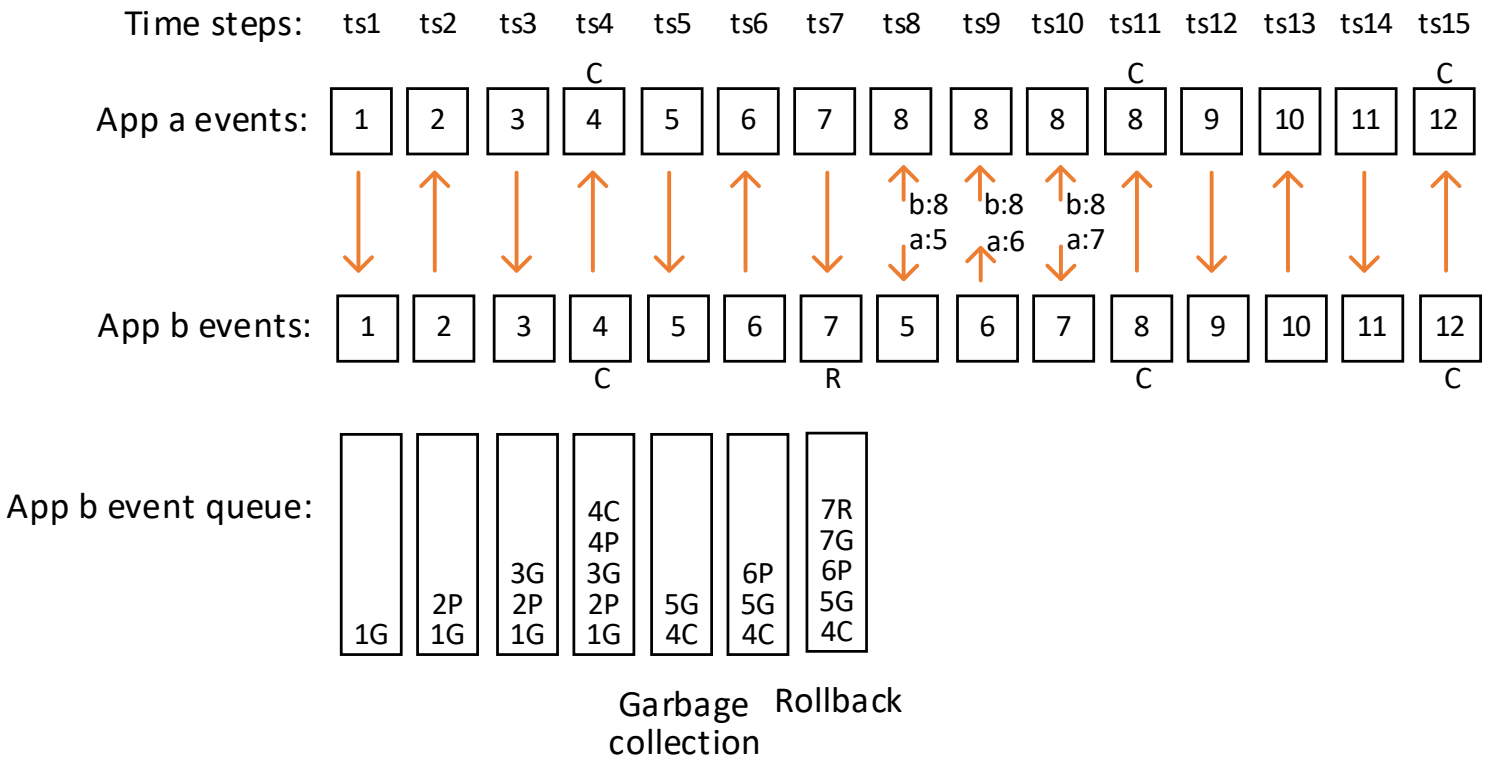
- *TS 1 ~ TS 7: Simulation b performs checkpoint and garbage collection in staging area;*

Figure 5: An illustration of queue based data consistency algorithm for workflows.

Checkpoint/Restart with Data Logging

❑ Data Logging in Staging Area

A queue based algorithm in staging area to keep data/event consistency.



G: Data Get
P: Data Put
C: Checkpointing
R: Rollback

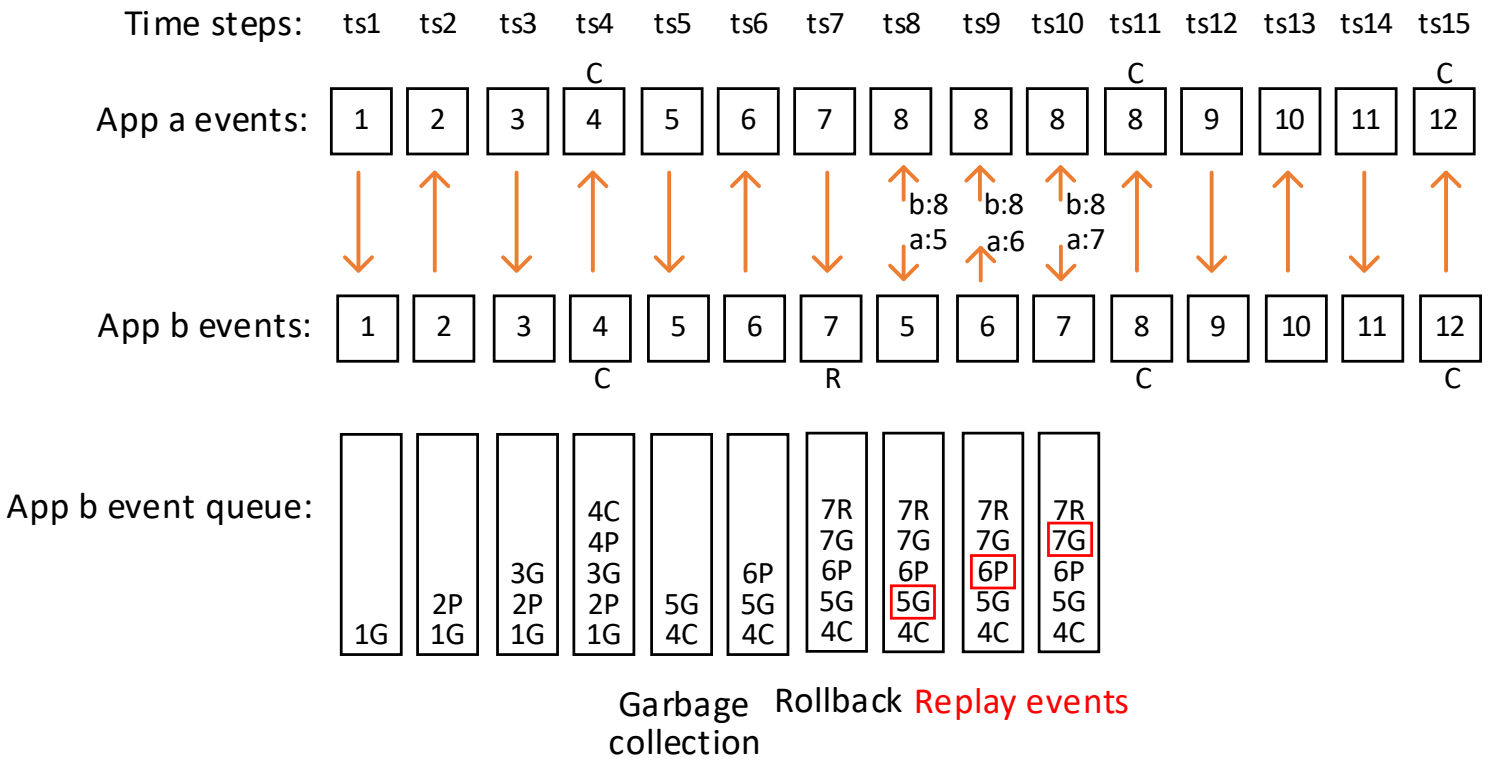
- *TS 1 ~ TS 7: Simulation b performs checkpoint and garbage collection in staging area;*
- *TS 7: Simulation b failed and performs rollback recovery;*

Figure 5: An illustration of queue based data consistency algorithm for workflows.

Checkpoint/Restart with Data Logging

❑ Data Logging in Staging Area

A queue based algorithm in staging area to keep data/event consistency.



G: Data Get
P: Data Put
C: Checkpointing
R: Rollback

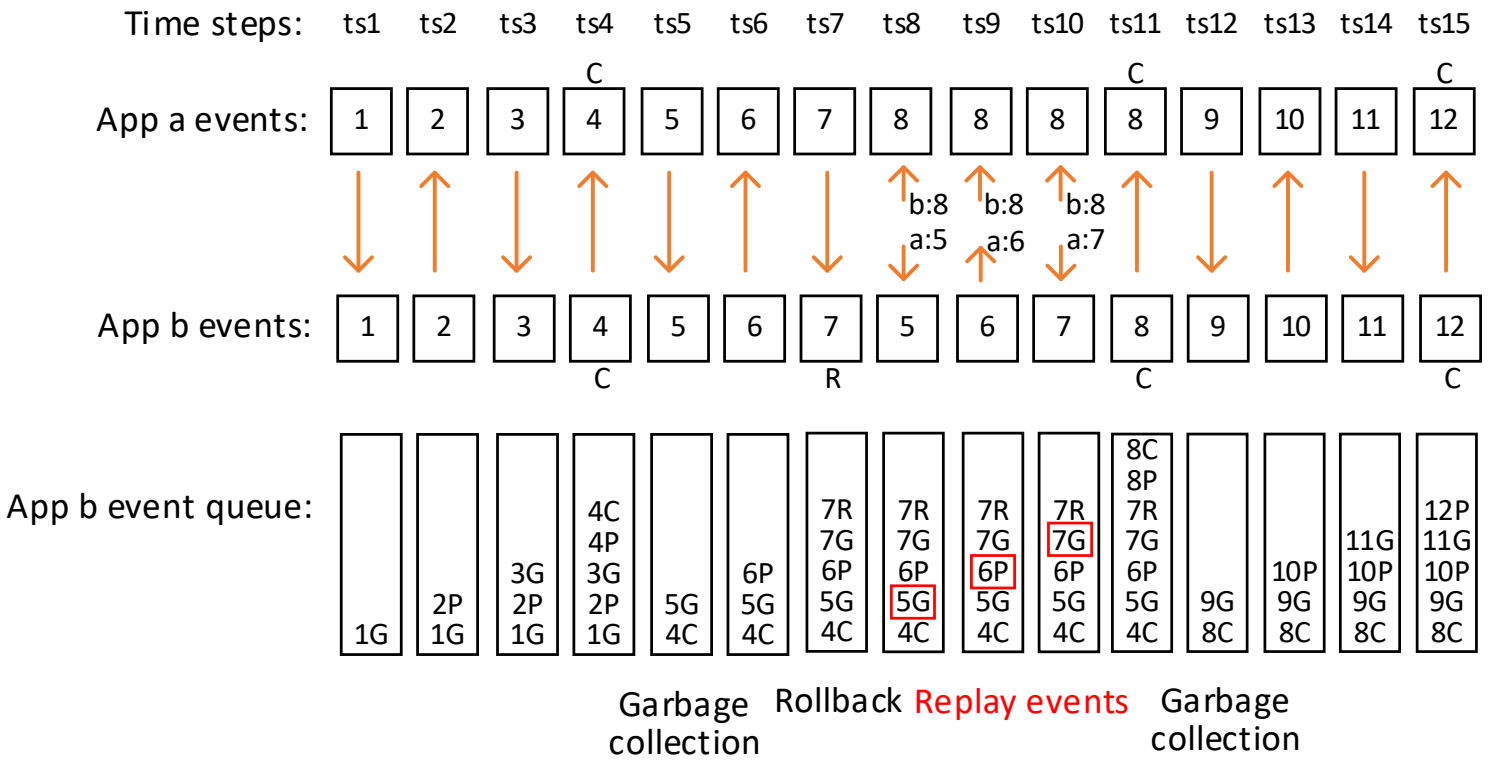
- *TS 1 ~ TS 7: Simulation b performs checkpoint and garbage collection in staging area;*
- *TS 7: Simulation b failed and performs rollback recovery;*
- *TS 8 ~ TS10: staging area relays the events (ts5~ts7) in the event queue.*

Figure 5: An illustration of queue based data consistency algorithm for workflows.

Checkpoint/Restart with Data Logging

❑ Data Logging in Staging Area

A queue based algorithm in staging area to keep data/event consistency.



G: Data Get
P: Data Put
C: Checkpointing
R: Rollback

- *TS 1 ~ TS 7: Simulation b performs checkpoint and garbage collection in staging area;*
- *TS 7: Simulation b failed and performs rollback recovery;*
- *TS 8 ~ TS10: staging area relays the events (ts5~ts7) in the event queue.*

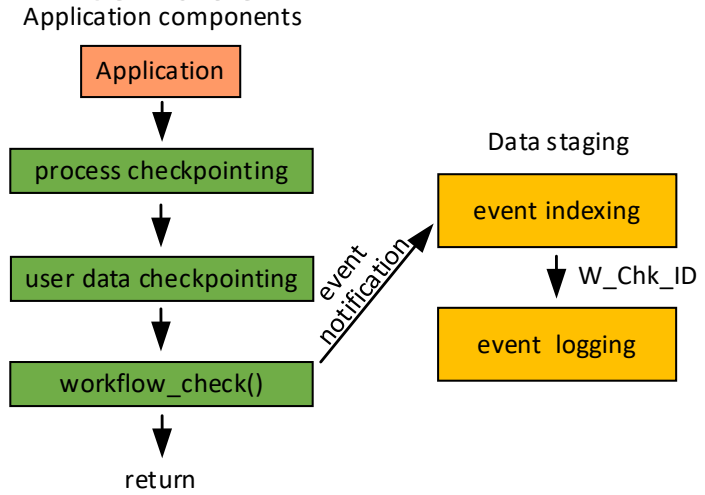
Figure 5: An illustration of queue based data consistency algorithm for workflows.

Checkpoint/Restart with Data Logging

- **User Interface**

Checkpoint/Restart with Data Logging

❑ User Interface

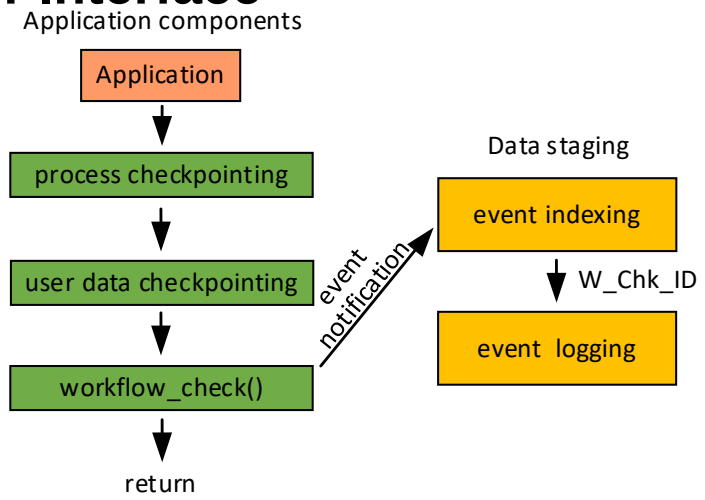


***workflow_check*()**

✓ Send a checkpoint event to data staging.

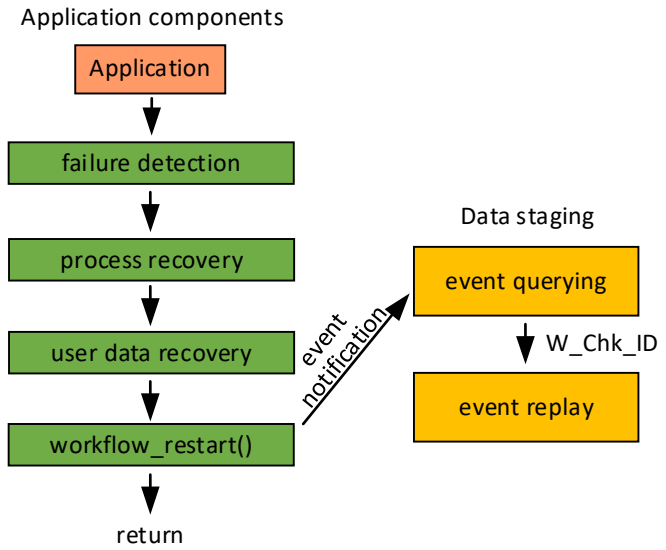
Checkpoint/Restart with Data Logging

❑ User Interface



***workflow_check*()**

✓ Send a checkpoint event to data staging.

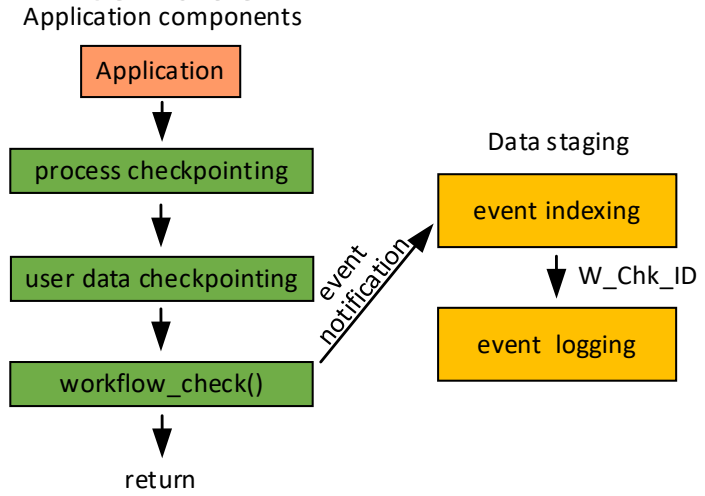


***workflow_restart*()**

✓ Recover data staging client, and notify the recovery event to data staging.

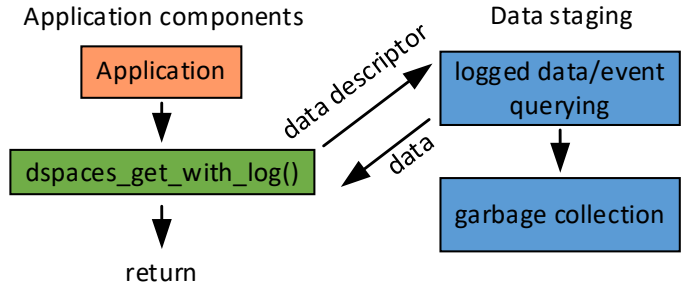
Checkpoint/Restart with Data Logging

❑ User Interface



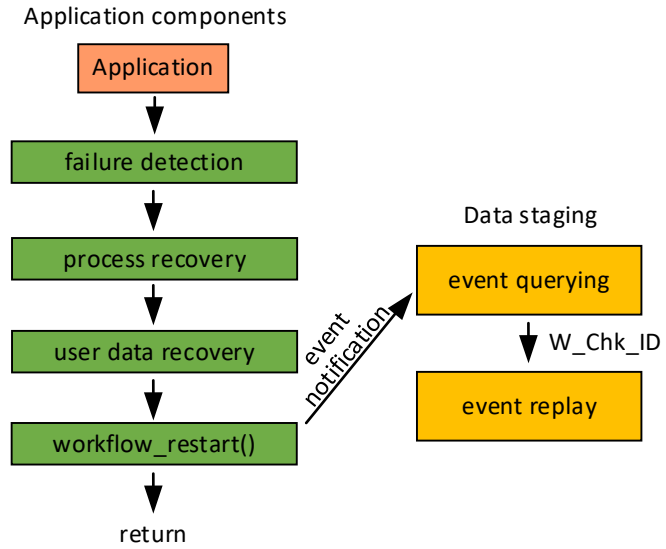
workflow_check()

✓ Send a checkpoint event to data staging.



dspaces_get_with_log()

✓ Retrieve the logged data specified by geometric descriptor from data staging.

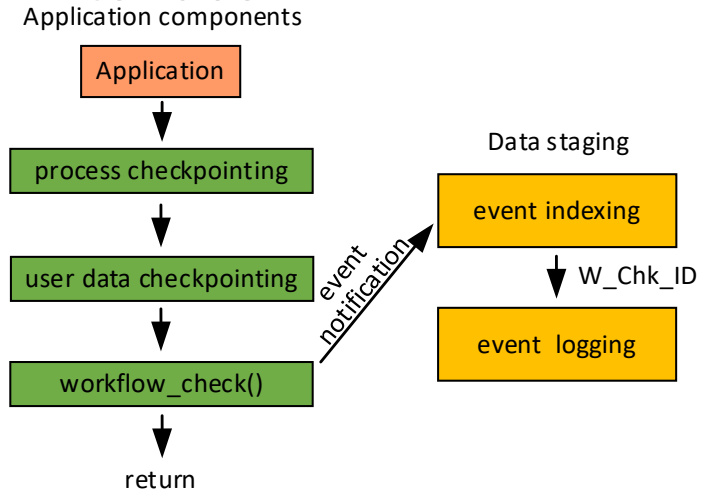


workflow_restart()

✓ Recover data staging client, and notify the recovery event to data staging.

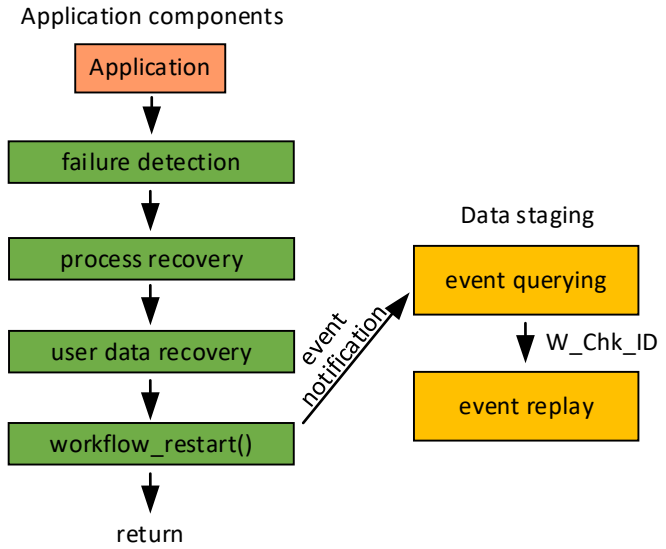
Checkpoint/Restart with Data Logging

❑ User Interface



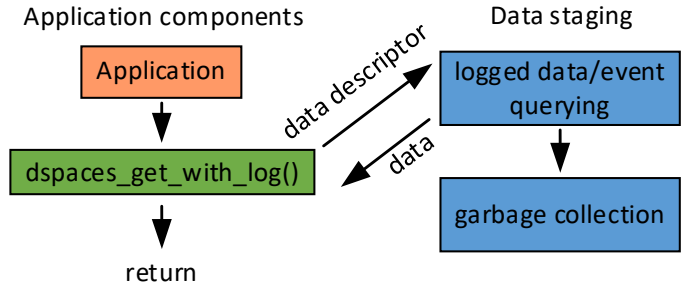
workflow_check()

✓ Send a checkpoint event to data staging.



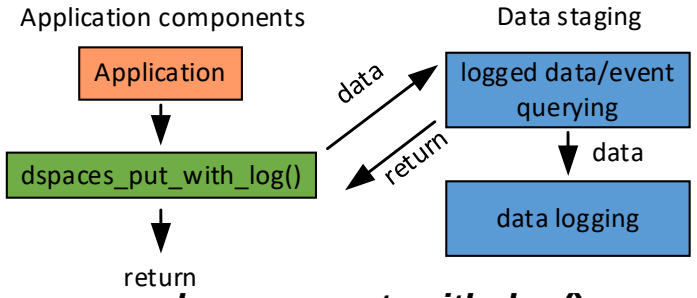
workflow_restart()

✓ Recover data staging client, and notify the recovery event to data staging.



dspaces_get_with_log()

✓ Retrieve the logged data specified by geometric descriptor from data staging.



dspaces_put_with_log()

✓ Log data to data staging, and omit data write requests in recovery phase.

Checkpoint/Restart with Data Logging

❑ Workflow Checkpoint/Restart Architecture

NERSC Cori, Cray XC40 system

- 622,336 Cores
- Aries interconnect
- 878,592 GB system memory
- Intel Xeon Phi 7250 68Cores 1.4GHz
- 8 NVIDIA V100 ('Volta') GPUs, each with 16 GB HBM2 memory

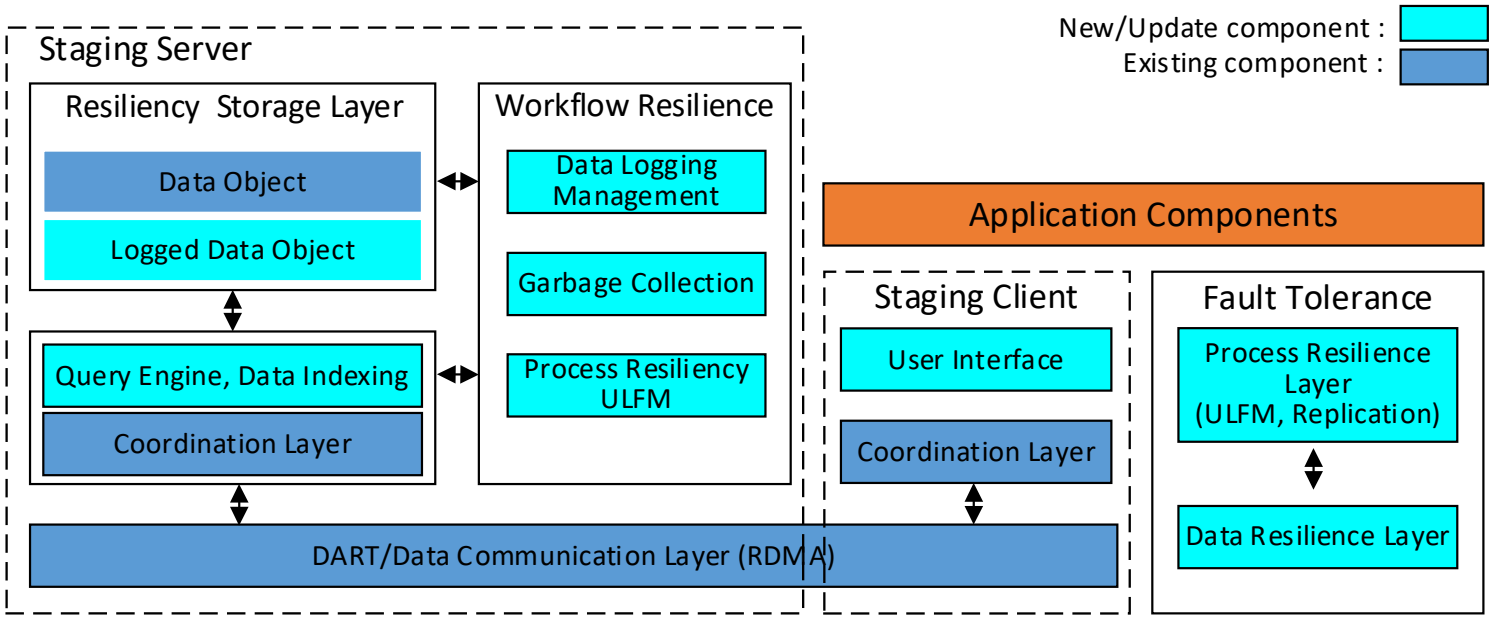


Figure 7: Experiments buildup of Checkpoint/Restart with data logging

- ❑ **Dedicated staging node:**
  - DataSpaces (CoREC) with data logging
- ❑ **Application node:**
  - Synthetic application workflows, Checkpoint/restart (ULFM), Process replication.

Experimental Evaluation

- **Synthetic Experiments**: 2 test cases with typical data read/write pattern from real workflow.

NERSC Cori, Cray XC40 system

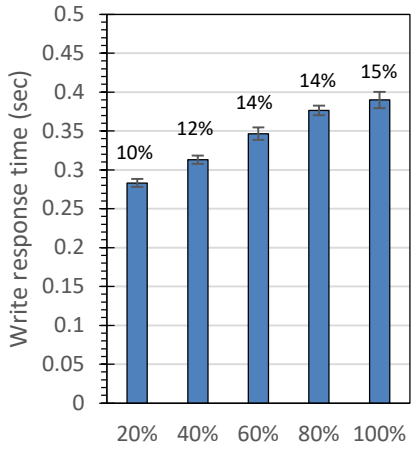
- 622,336 Cores
- Aries interconnect
- 878,592 GB system memory
- Intel Xeon Phi 7250 68Cores 1.4GHz
- 8 NVIDIA V100 ('Volta') GPUs, each with 16 GB HBM2 memory

Synthetic Experiments	
Total No. of cores	256 + 64 + 32 = 352
No. of parallel writer cores	8 x 8 x 4 = 256
No. of staging cores	32
No. of parallel reader cores	64
Volume size	512 x 512 x 256
In staging data size (40 ts)	20GB
Data access pattern	write immediately followed by read
Coordinated checkpoint period (ts)	4
Simulation checkpoint period (ts)	4
Analytic checkpoint period (ts)	5
Scalability Experiments	

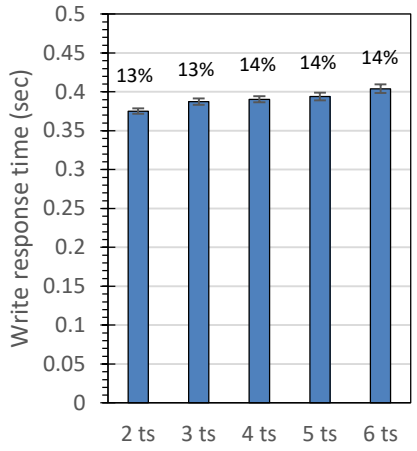
Baselines: Global Coordinated Checkpoint / Restart.

Experimental Evaluation

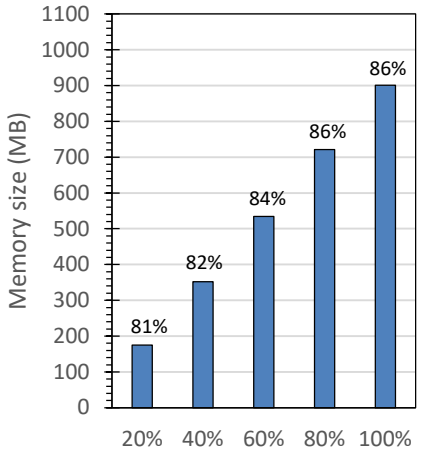
❑ Synthetic Experiments



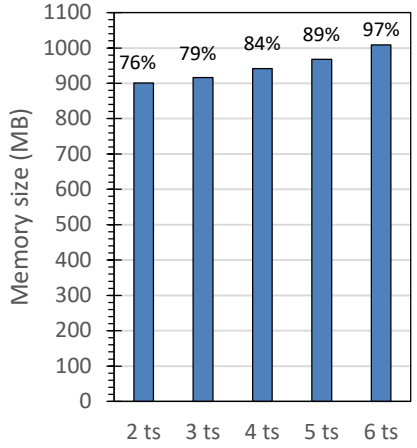
(a) *Case 1 write latency*



(b) *Case 2 write latency*



(c) *Case 1 storage cost*



(d) *Case 2 storage cost*

Case #	Description
1	Write different percentage subsets of the entire data domain in each time step.
2	Write the entire data domain and perform checkpointing with different frequencies.

Measuring cumulative data write response time and memory usage.

✓ *Percentages on top of bars indicate extra write response time / memory usage.*

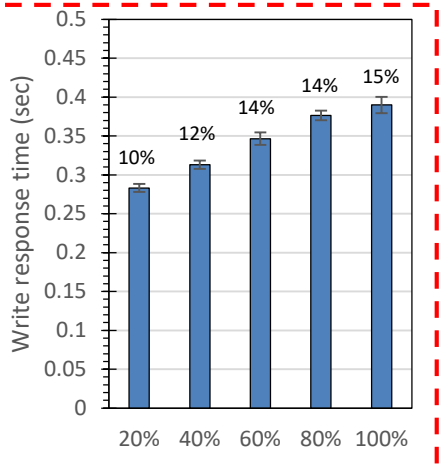
Result:

✓ Case 1: data logging increased the write response time by **10%**, **12%**, **14%**, **14%**, and **15%** respectively. increased the memory usage by **81%**, **82%**, **84%**, **86%**, and **86%** respectively.

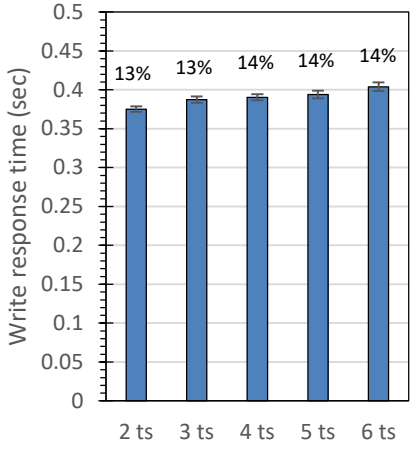
✓ Case 2: data logging increased the write response time by maximum **14%**. increases memory usage by **76%**, **79%**, **84%**, **89%**, and **97%** respectively.

Experimental Evaluation

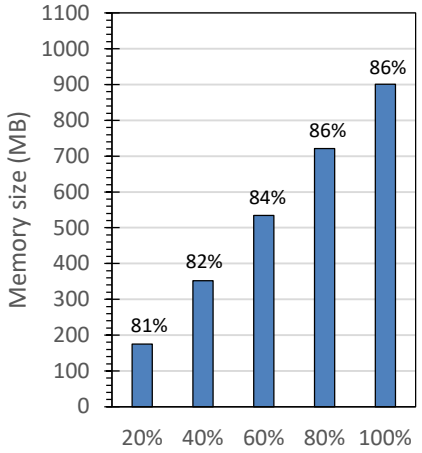
❑ Synthetic Experiments



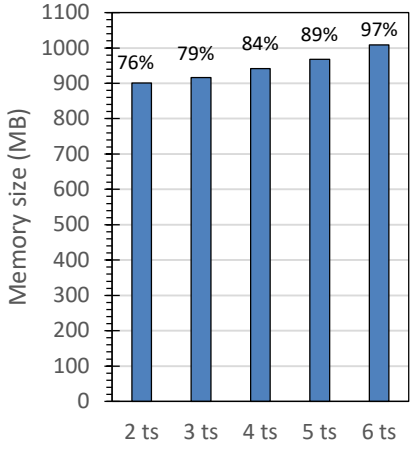
(a) *Case 1 write latency*



(b) *Case 2 write latency*



(c) *Case 1 storage cost*



(d) *Case 2 storage cost*

Case #	Description
1	Write different percentage subsets of the entire data domain in each time step.
2	Write the entire data domain and perform checkpointing with different frequencies.

Measuring cumulative data write response time and memory usage.

✓ *Percentages on top of bars indicate extra write response time / memory usage.*

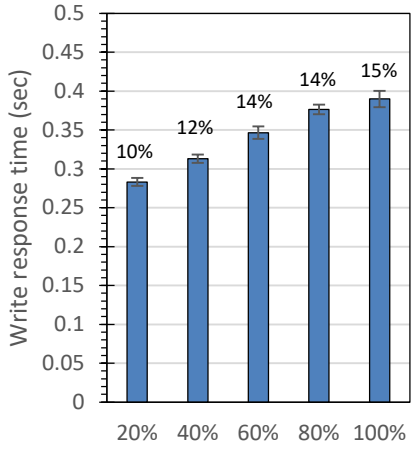
Result:

✓ Case 1: data logging increased the write response time by **10%**, **12%**, **14%**, **14%**, and **15%** respectively. increased the memory usage by **81%**, **82%**, **84%**, **86%**, and **86%** respectively.

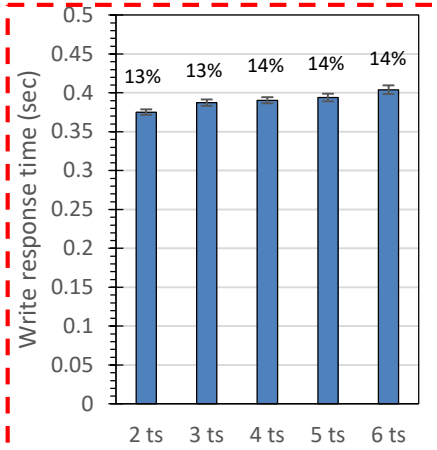
✓ Case 2: data logging increased the write response time by maximum **14%**. increases memory usage by **76%**, **79%**, **84%**, **89%**, and **97%** respectively.

Experimental Evaluation

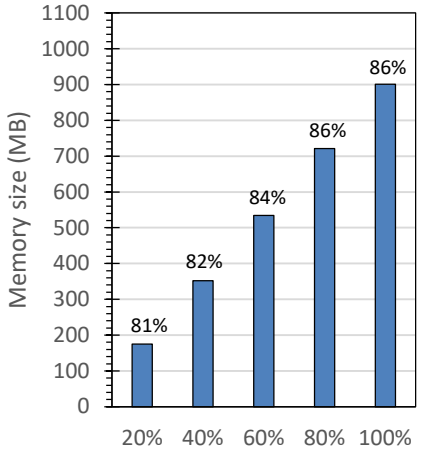
❑ Synthetic Experiments



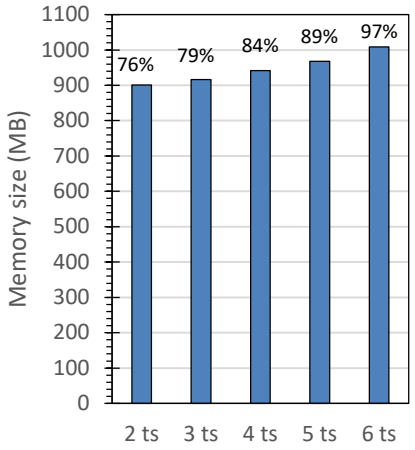
(a) *Case 1 write latency*



(b) *Case 2 write latency*



(c) *Case 1 storage cost*



(d) *Case 2 storage cost*

Case #	Description
1	Write different percentage subsets of the entire data domain in each time step.
2	Write the entire data domain and perform checkpointing with different frequencies.

Measuring cumulative data write response time and memory usage.

✓ *Percentages on top of bars indicate extra write response time / memory usage.*

Result:

✓ Case 1: data logging increased the write response time by **10%**, **12%**, **14%**, **14%**, and **15%** respectively. increased the memory usage by **81%**, **82%**, **84%**, **86%**, and **86%** respectively.

✓ Case 2: data logging increased the write response time by maximum **14%**. increases memory usage by **76%**, **79%**, **84%**, **89%**, and **97%** respectively.

Experimental Evaluation

❑ Synthetic Experiments

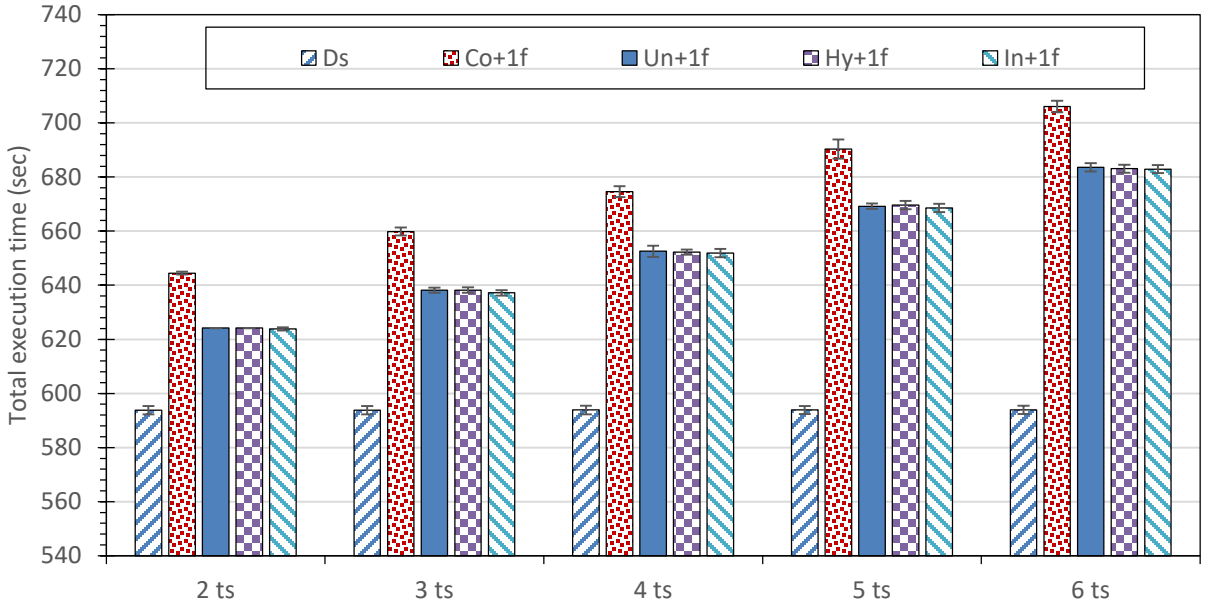


Figure 9: Case 2 Workflow execution time

***Ds**: The workflow without data logging in staging;*

***Co**: Global coordinated checkpoint/restart;*

***Un**: Uncoordinated checkpoint/restart;*

***Hy**: Hybrid Checkpoint/restart with process replication;*

***In**: Individual checkpoint/restart;*

***+1f**: with 1 synthetic process failure.*

Result:

✓ The uncoordinated and hybrid checkpoint/restart with data logging reduce the total execution time around **3.15%** for 2 *ts*, **3.28%** for 3 *ts*, **3.26%** for 4 *ts*, **3.05%** for 5 *ts* and **3.18%** for 6 *ts* relative to global coordinated checkpoint (baseline) respectively.

Experimental Evaluation

❑ Scalability Experiments

NERSC Cori, Cray XC40 system

- 622,336 Cores
- Aries interconnect
- 878,592 GB system memory
- Intel Xeon Phi 7250 68Cores 1.4GHz
- 8 NVIDIA V100 ('Volta') GPUs, each with 16 GB HBM2 memory

Synthetic Experiments					
Scalability Experiments					
Total No. of cores	704	1408	2816	5632	11264
No. of simulation cores	512	1024	2048	4096	8192
No. of staging cores	64	128	256	512	1024
No. of analytic cores	128	256	512	1024	2048
No. of analysis cores	40	80	160	320	640
Data size (40 ts)(GB)	40	80	160	320	640
Coordinated checkpoint period (ts)	8	8	8	8	8
Simulation checkpoint period (ts)	8	8	8	8	8
Analytic checkpoint period (ts)	10	10	10	10	10
MTBF (sec)/No. of failures		600 sec /1,	300 sec /2,	200 sec /3	

Baselines: Global Coordinated Checkpoint / Restart.

Experimental Evaluation

❑ Scalability Experiments

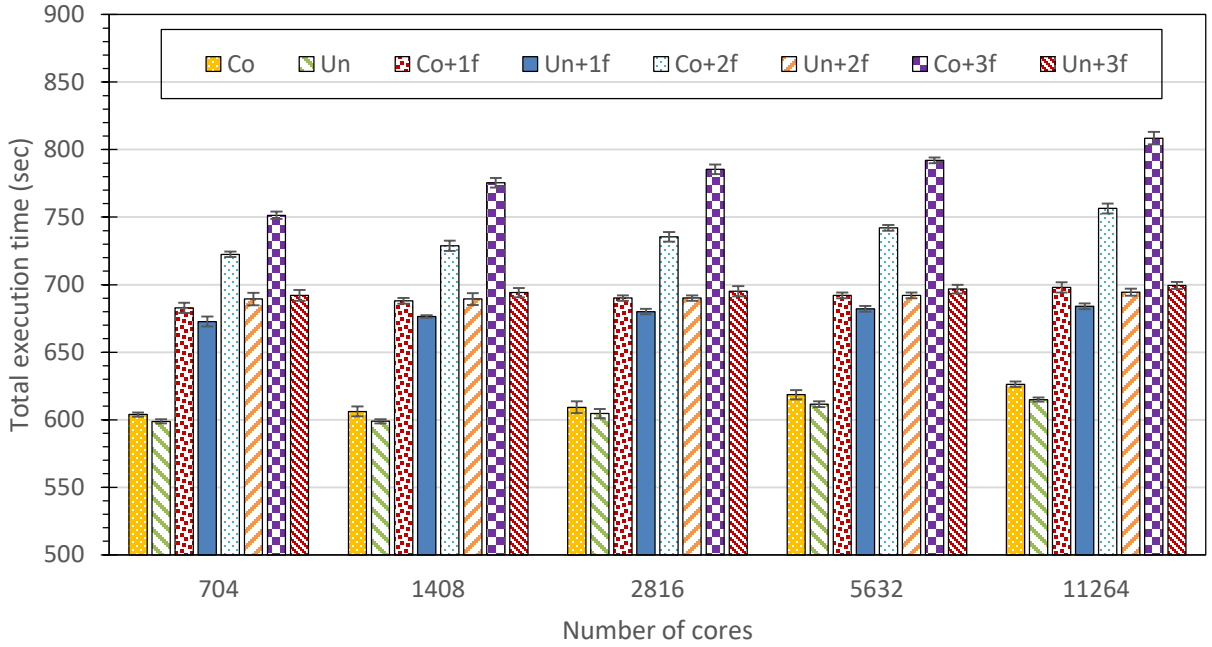


Figure 10: Workflow execution time at different scales

***Co**: Global coordinated checkpoint/restart;*

***Un**: Uncoordinated checkpoint/restart;*

***+1f, +2f, +3f,** : with 1, 2, 3 synthetic process failures.*

Result:

✓ Workflow-level uncoordinated checkpoint/restart with data logging reduced the total workflow execution time by up to **7.89%**, **10.48%**, **11.5%**, **12.03%**, and **13.48%** on 704, 1408, 2816, 5632, and 11264 cores scales in comparison to global coordinated checkpoint/restart (baseline).

Conclusion

- Proposed a checkpoint/restart with data logging framework for tight coupled in-situ workflows to enable diverse fault tolerance schemes to be used in workflows while maintaining crash consistency effectively and efficiently.
  - ✓ The use of queue based data logging algorithm in staging to record and replay data access events and maintaining crash consistency.
  - ✓ A user interface for integrating this fault tolerance framework with application components.
- Evaluate its effectiveness and performance through synthetic cases experiment and scalability experiment.

Future Work

- Integrate the approach described in this paper with other fault tolerance methods such as proactive checkpointing and hierarchical checkpointing, and investigate the impact of our approach using real application workflows such as S3D.

Acknowledgments

This worked was supported by

National Science Foundation (NSF) via grant number CCF-1725649,

Sandia National Laboratories under contract DE-NA-0003525,

National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.

Thank You!