Characterizing Impacts of Storage Faults on HPC Applications: A Methodology and Insights

*Bo Fang *(Pacific Northwest National Laboratory)*

***Daoce Wang** *(Washington State University)*

Sian Jin *(Washington State University)*

Quincey Koziol *(Lawrence Berkeley National Laboratory)*

Zhao Zhang *(Texas Advanced Computing Center)*

Qiang Guan *(Kent State University)*

Suren Byna *(Lawrence Berkeley National Laboratory)*

Sriram Krishnamoorthy *(Pacific Northwest National Laboratory)*

Dingwen Tao *(Washington State University)*

*These authors contributed equally

Introduction

- Goal
  - Provide methodology for characterizing how different SSD-related faults would affect the behaviors of the high-performance computing (HPC) apps and mitigate data corruption in HPC apps
- What we have done
  - Build a fault injection framework, (FUSE-based Fault Injection for Storage) FFIS, to model SSD-related failures and to inject faults systematically into HPC apps
  - Apply FFIS on three real-world HPC apps through large-scale experiments
  - Show that apps exhibit distinct error resilience characteristics for different fault models
  - Offer a detailed explanation for each app's unique resilience characteristics
  - Unveil app-specific behaviors on HDF5 and show the fault-tolerance behaviors of the HDF5 against errors affecting the HDF5 metadata
  - Identify certain fields in the metadata that may cause SDC outcomes and provide auto-correction solutions

Motivation & Background

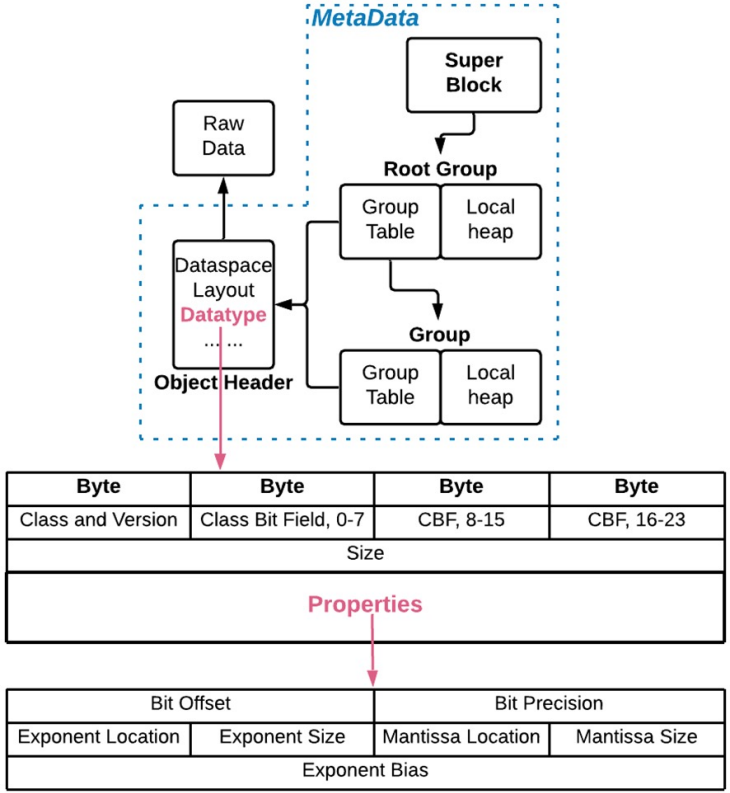
- Why solid-state disks (SSDs)?
  - The increasing complexity in scientific simulations urges (HPC) platforms to equip with more advanced storage infrastructures (SSD)
- SSD Failure & Why this work
  - The reliability challenges faced by the HPC applications under the SSD-related failures remain unclear
  - Some SSD data corruption can bypass the file system and affect application behavior*
  - It is urgent to understand the impact of SSD-related faults on HPC apps

S. Jaffer, S. Maneas, A. Hwang, and B. Schroeder, "Evaluating file system reliability on solid state drives,"

https://laur.ie/blog/2015/06/ssds-a-gift-and-a-curse/

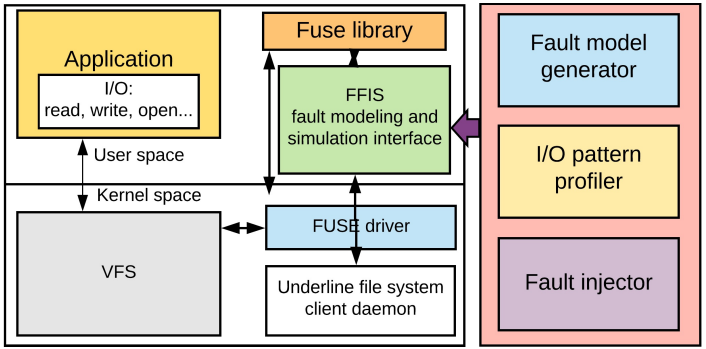
Motivation & Background

- HDF5 & Why HDF5
  - Provides an API for performing I/O, data management tools, and a portable file format
  - The most used I/O library on HPC systems at the NERSC and at several US DOE supercomputing facilities
  - Has a rich ecosystem and various third-party bindings are available to manage data
  - Cascading style metadata: holds a superblock that points to multiple groups, and each group represents an object header that may store other groups or datasets within it



FFIS Framework

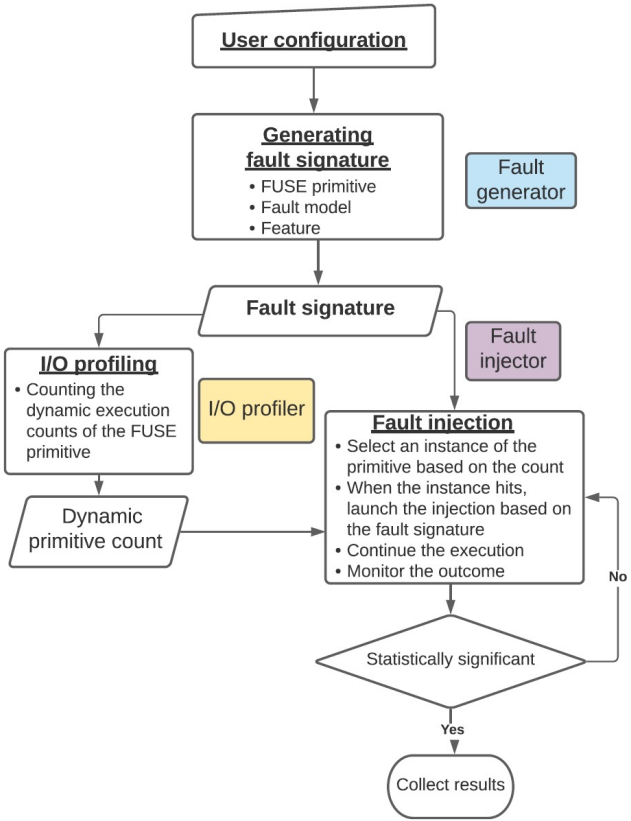
- File System in User-space (FUSE)
  - Allows the users to implement their own file operations
  - Allows the apps to call the user's I/O primitives without modification
- Characteristic
  - ***Transparency***: Transparently plant a fault into an app at runtime without modification
  - ***Convenience***: Deploy without modifying the environment of the app
  - ***Comprehensiveness***: Support multiple fault models for different types of SSD-related failures.
  - ***Repressiveness***: Introduce faults uniformly over all corresponding file operations



- Fault models
  - **Bitflip:** flip 2 consecutive bits randomly chosen in the buffer
  - **Shorn Write:** lose the last 1/8th of the data in the buffer
  - **Dropped Write:** Ignores the *pwrite* call for that instance inside the FFIS write

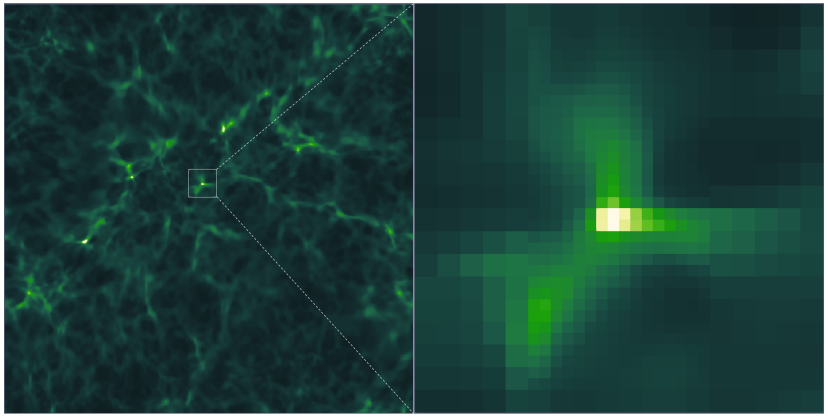
FFIS Framework

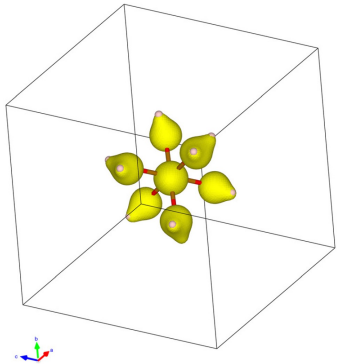
- Workflow
  - **Fault Generator:** reads the configuration to produce a fault signature
  - **I/O Profiler:** count the number of times that the primitive (i.e., configured in the fault signature) gets executed during the execution.
  - **Fault Injector:** performs the actual fault injection operations with the fault signature, including the fault model, the feature and the primitive, and the dynamic count obtained from the profiler

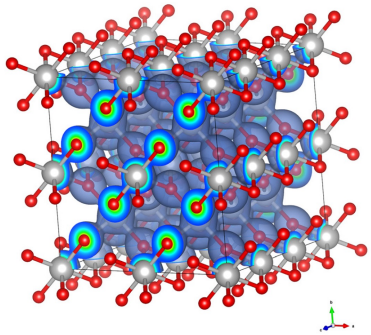


Evaluation Applications & Fault Injection Approaches

- Nyx
  - An adaptive mesh, hydrodynamics algorithm that model astrophysical reacting flows.
  - We select the most popular post-analysis: **HALO FINDER**, which aims to find the halos using the “baryon density” field
- QMCPACK
  - Implements numerous Quantum Monte Carlo (QMC) algorithms for electronic structure calculations
  - Use the QMCA tool in QMCPACK to obtain the total energies (2.90372 Hartree)







Evaluation Applications & Fault Injection Approaches

- Montage
  - A toolkit to assemble Flexible Image Transport System (FITS) images into custom mosaics.
  - Takes ten stages to generate the final image, four of which involve a large amount of I/O
  - Inject the faults in different stages to study the fault propagation

- Outcome Classification

	Benign	Detected	Silent Data Corruption
Nyx	Outputs of Halo-finder unchanged	Outputs differ, no halo found	Outputs differ, find some halos
QMCPACK	E = -2.90372 Hartree	otherwise	E in [-2.91, -2.90]
Montage	Min unchanged	otherwise	Min in [82.82, 82.83]

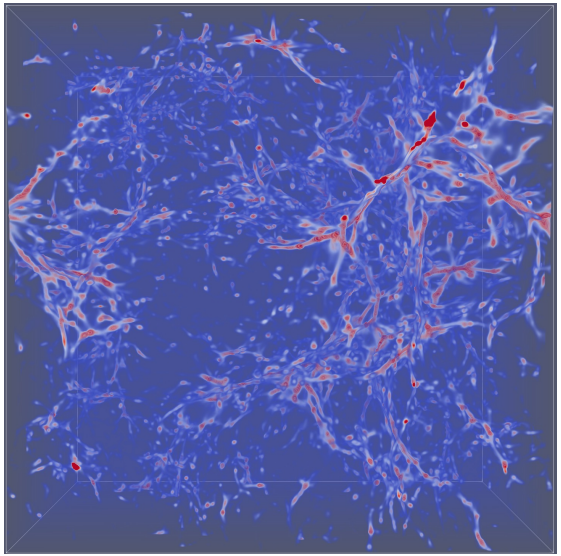
Results for Faults Affecting HDF5 Metadata

- How HDF5 file is created & how we inject fault in Metadata
  - Locks the file to prevent the concurrent writes
  - Performs multiple writes to store the raw data
  - Packs all metadata and write them to the file
  - Unlocks the file for later access
  - Identifies the specific write operation for metadata then perform a byte-wise injection

Fault Types	Case Number	Example Metadata Fields and Bytes
SDC	4 (0.2%)	Bit-5 of Mantissa Normalization, Exponent Location, Mantissa Location, Mantissa Size, Exponent Bias, Address of Raw Data (ARD)
Benign	2085 (85.7%)	Bit Offset, Bit Precision, Size Reserved and unused bytes, other trifling fields, etc.
Crash	343 (14.1%)	Version # of Data Object Header, Version # of Data Object Header Message, Symbol Table Node signature, B-tree signature, etc.

- Why so many benign cases
  - Reserved fields, alignment space between fields, and space for future
  - Certain fields that exhibit resilient behaviors for the HDF5 files.

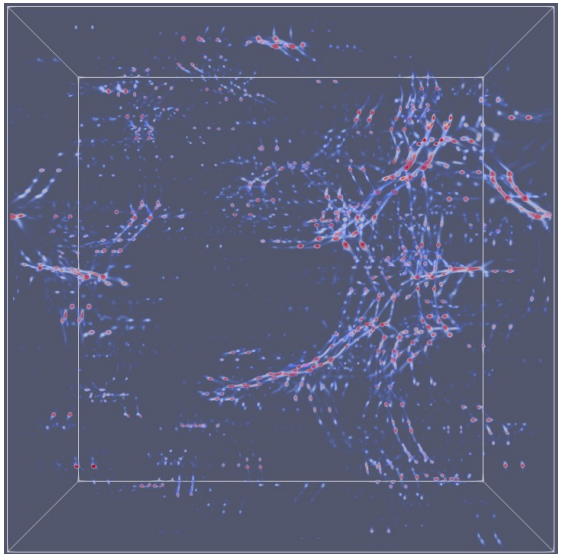
Metadata SDC cases Vis



Original Data

Faulty ExpandBias

Scales up the input data



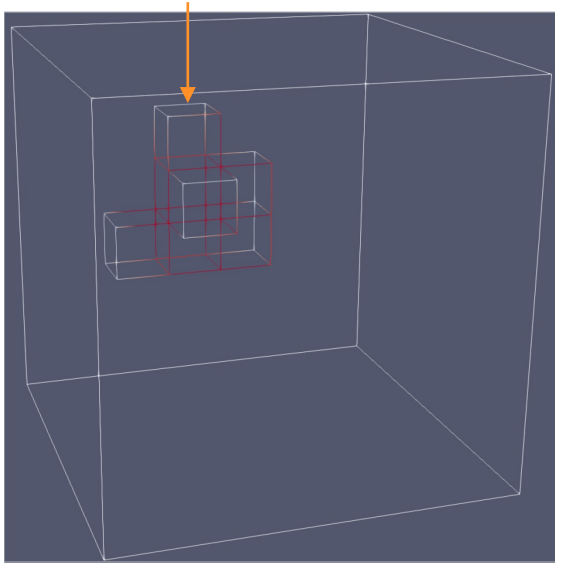
Original Data

Faulty ARB

Shifts the input data

The number of halo cell candidates is reduced compared to the original case, thus there are not enough halo candidates to form a halo.

A box indicates a halo cell candidate that meets the threshold



Original Data

Faulty HaloSize

1. The mass of an object(s) must be greater than a threshold (e.g., 81.66 times the average mass of the whole dataset) to become a halo cell candidate
2. There must be enough halo cell candidates in a certain area to form a halo

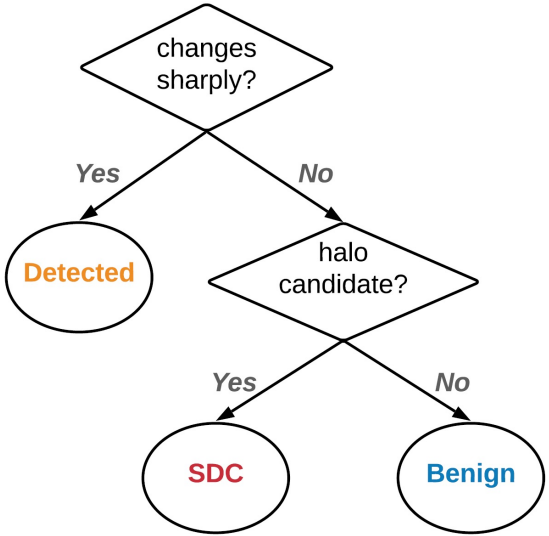
Detection & Correction & Protection

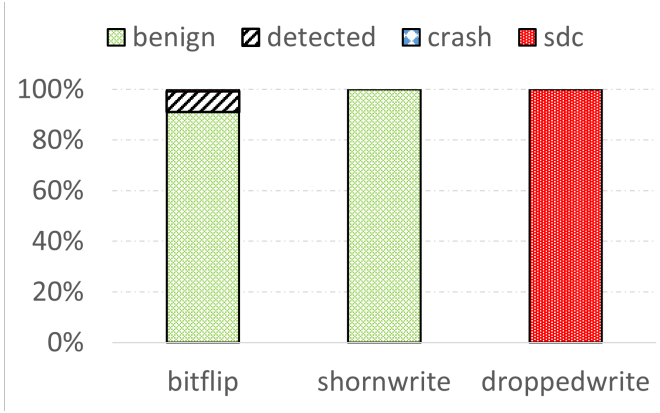
Avg-based Detection: Avg (input_data) == 1 due to the law of mass conservation

- Exponent Bias
  - **If avg (input_data) = 2^n**
  - Re-scale Exponent Bias field based on the average value observed
    - Avg becomes 4096 after fault injection, the Exponent Bias changes from 0x0000007f to 0x00000073, can be corrected via added by 12 (i.e., $2^{12} = 4096$)
- Exponent Location, Mantissa Location, or Mantissa Size
  - **If avg (input_data) in (1, 2)**
  - Fix the fault using the constraint of these three fields
    - Exponent Location == Mantissa Size; Mantissa Size + Exponent Size == Precision - 1
- ARD
  - cannot determine whether there is a fault based on avg (input_data)
  - Introduce a protection mechanism
    - ARD == size of metadata

Results for Faults Affecting Application Data: NYX

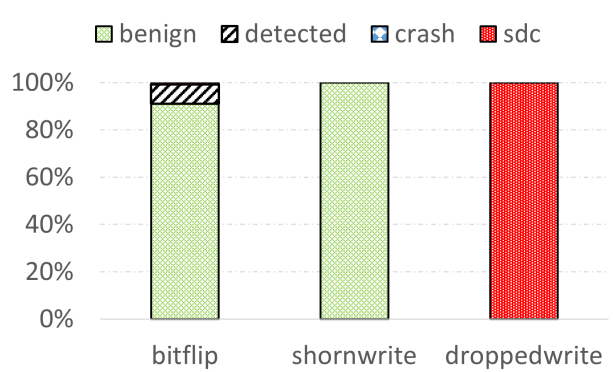
- Bit Flip
  - When the mass of a point changes sharply, avg (input_data) changes accordingly -> the mass of all points in the dataset is less than the threshold (Detected)
  - Change is not significant -> original **halo candidates** still satisfy the threshold, but a **halo candidate** affected by the fault may cause (SDC)
  - Most of the points are not **halo candidate** -> Benign

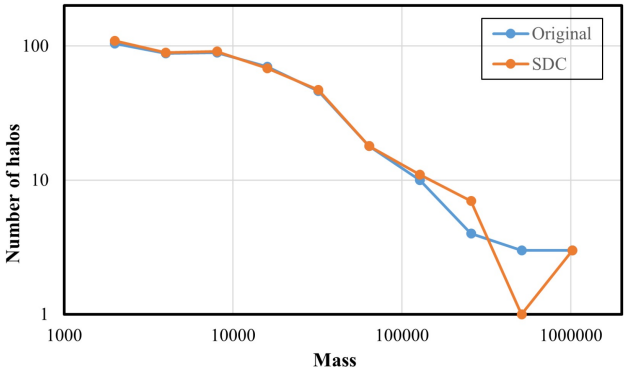




Results for Faults Affecting Application Data: NYX

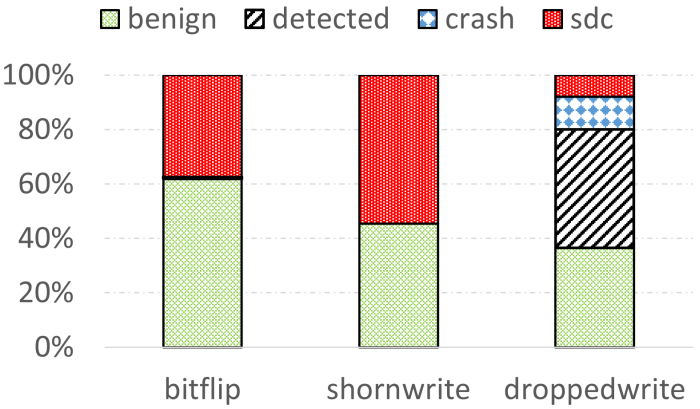
- Shorn Write
  - Replaces the unsuccessfully written data with the data that is within an order of magnitude
  - These “moderate” faults are all mitigated
- Dropped Write
  - A Dropped Write fault drops a large piece of data -> avg change -> diff output
  - halos with larger mass are more susceptible to Dropped Write
- Observation and Insight
  - All the SDC cases in Dropped Write can be detected using avg-based method (avg < 0.9983)
  - **Sensitive** to large deviations, **mitigate** small deviations -> highly resilient to faults using the avg-based method





Results for Faults Affecting Application Data: QMCPACK

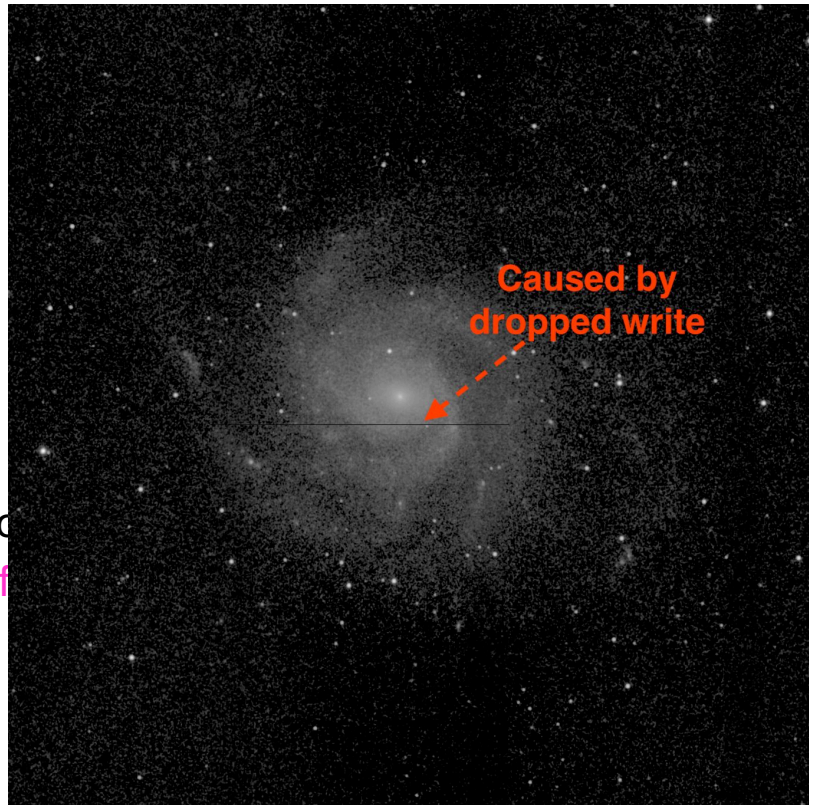
- Analyze of three fault models
  - Compared to **Bit Flip** (2-bit), **Shorn Write** (512-bytes) has a higher percentage of SDC
  - All the **Shorn Write** faults are SDC while some of **Bit Flip** faults are detected
  - Output of **Dropped Write** deviates more from the correct energy compared to Shorn Write -> more detected cases



- Observation and Insight
  - QMCPACK is not resilient to Bit Flip and Shorn Write, as they have a high likelihood to cause a minor deviation (SDC) in the outcomes
  - More advanced techniques guided by more domain knowledge need to be considered for QMCPACK

Results for Faults Affecting Application Data: Montage

- Analyze of three fault models
  - **Bit Flip:** SDC cases stay stable (e.g., 12.8%, 8%, 9%, 6.8%).
  - **Shorn Write:** SDC and benign cases vary slightly (e.g., 56.6%, 40%, 52.5%, 48.5%)
  - **Dropped Write:** SDC and benign cases vary more drastically (e.g., 83.5%, 37.3%, 98.3%, 50.4%) than **Shorn Write**
- Observation and Insight
  - Relatively small fluctuation in the SDC rates fo
  - Different Montage stages seem to bound the f stage decouples from each other



Future Work & Conclusion

- Conclusion
  - Focuses on the HPC applications affected by the SSD-related failures
  - Proposes the fault injection framework FFIS to study the impact of failures on the apps
  - Conduct comprehensive fault injection experiments on HPC apps from different domains, and show that different apps exhibit dramatically error resilience behaviors
  - Unveil application-specific behaviors operating on HDF5 and show the fault tolerance behaviors of the HDF5 metadata
  - Propose a detection approach to identify which metadata field is potentially incorrect and corresponding correction methodology
- What to do next?
  - Evaluate with wider variety of applications and try to generalize the insights