SOGANG UNIVERSITY

Fault-Tolerant Deep Learning Cache with Hash Ring for Load Balancing in HPC Systems

Seoyeong Lee[1], Awais Khan[2], Yoochan Kim[1], Junghwan Park[1], Soon Hwang[1], Jae-Kook Lee[3], Taeyoung Hong[3], Christopher Zimmer[2], Youngjae Kim[1]

[1] Dept. of Computer Science and Engineering, Sogang University
[2] Oak Ridge National Laboratory, [3] KISTI,

2024 9th International Workshop on Parallel Data Storage, held in conjunction with SC24, Nov 17–22, 2024, Atlanta, GA

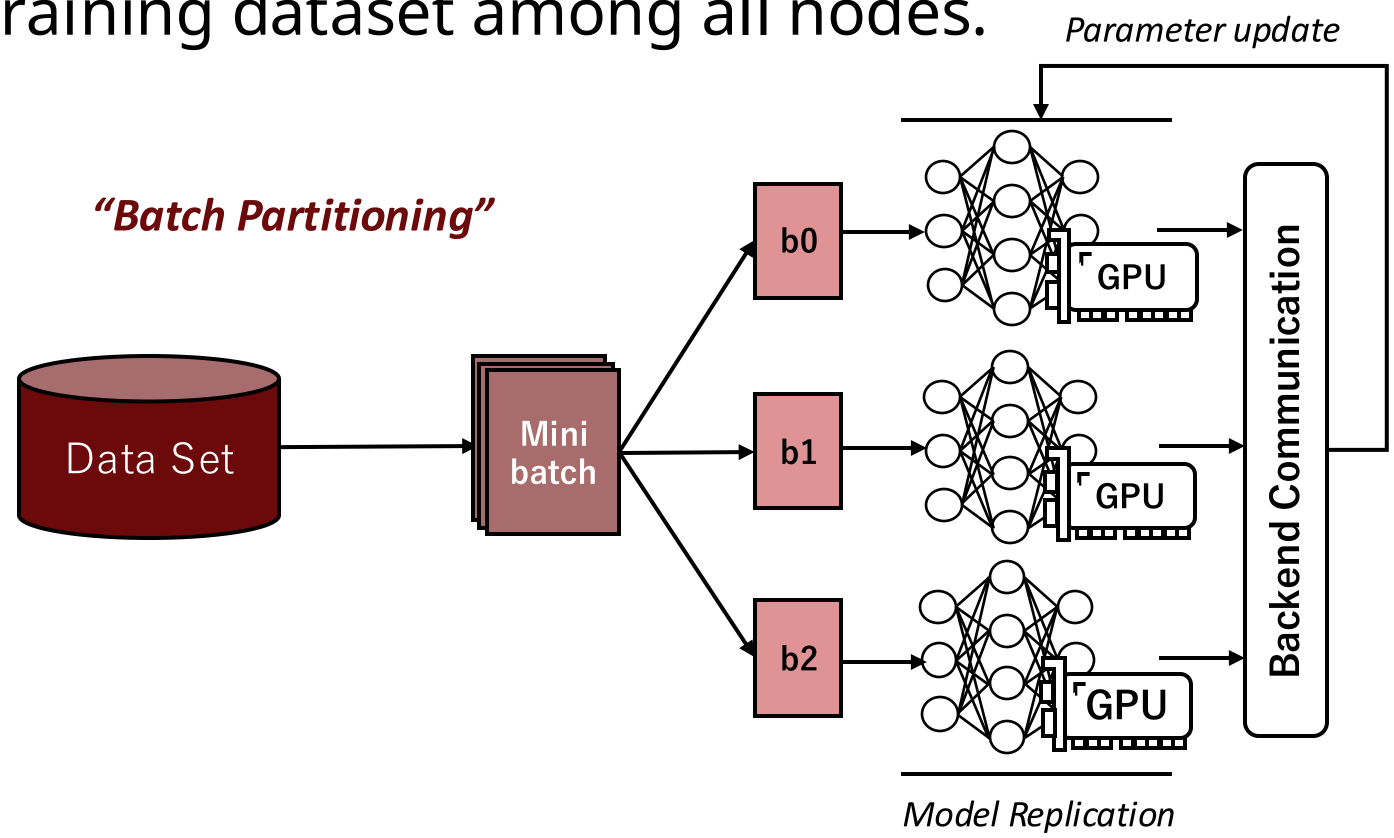
Contents

- Background
- Problem Definition
- Design & Implementation
- Evaluation
- Conclusion

Background

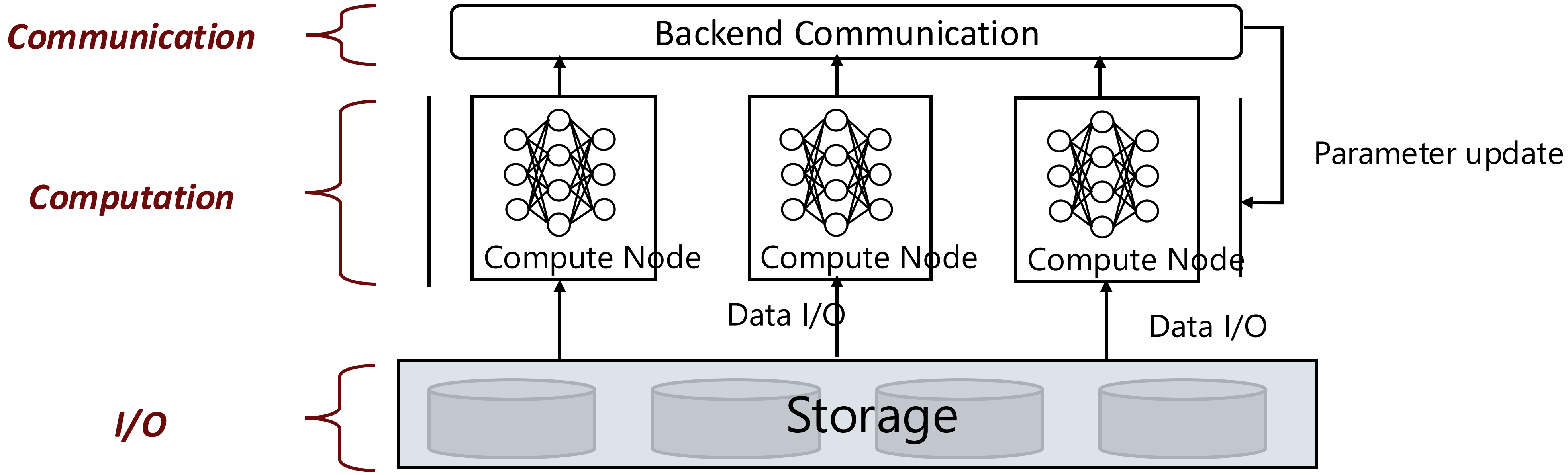
Distributed Deep Learning in HPC

- The Data Parallel Approach
  - Replicates the deep learning model across nodes while distributing the training dataset among all nodes.



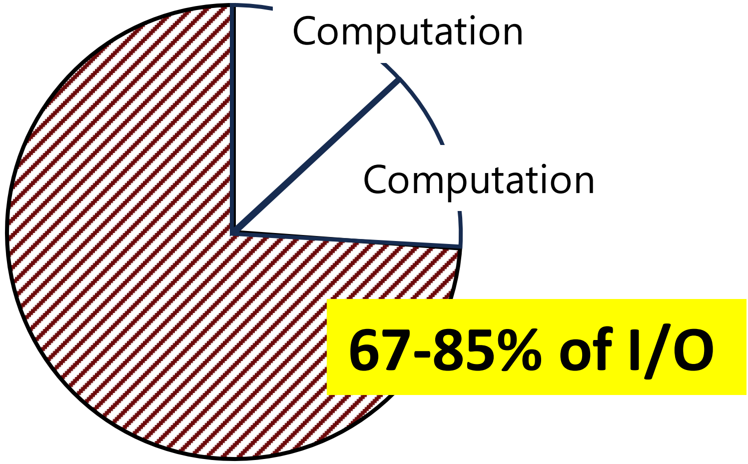
Distributed Deep Learning in HPC

- Three key aspects of Distributed Deep Learning
  - I/O, Computation, Communication
- Most prior research efforts have concentrated on improving computation and communication.



Distributed Deep Learning in HPC

- However, the optimizations in computation and communication, along with the development of modern computational accelerators and network technologies, have shifted the bottleneck towards I/O.
- ***I/O accounts for 67-85% of total training time[1] .***
    - ***Training ResNet50 on ImageNet: 85% of training runtime is IO overhead***



[1] N. Dryden, R. Böhringer, T. Ben-Nun, and T. Hoefler, "Clairvoyant Prefetching for Distributed Machine Learning I/O," Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21), 2021.

Characteristics of Deep Learning Image Dataset

- ***"Large Number of Small Files"***
    - ***ImageNet-1K:*** *1.28 million images (-150KB)*
    - ***ImageNet-21K:*** *11 million images (-163KB)*
    - ***OpenImages:*** *9 million images (-150KB)*
    - ***Google Landmarks Dataset v2:*** *5 million images (-200KB)*
    - ***Places365:*** *10 million images (-150KB)*

- The HPC I/O subsystem is ***not designed to efficiently handle the large-scale data I/O access required by deep learning frameworks.***

Optimizing I/O for Deep Learning Workloads

- ***NoPFS***[1]
  - Optimizes prefetching and caching by predicting data access patterns, reducing latency in training I/O.
- ***DeepIO***[2]
  - Minimizes backend storage reads by keeping data in memory, focusing on reducing read latency and boosting I/O efficiency for distributed training.
- ***HVAC***[3]
  - Caches data on node-local NVMe, specifically reducing repetitive I/O reads during training.

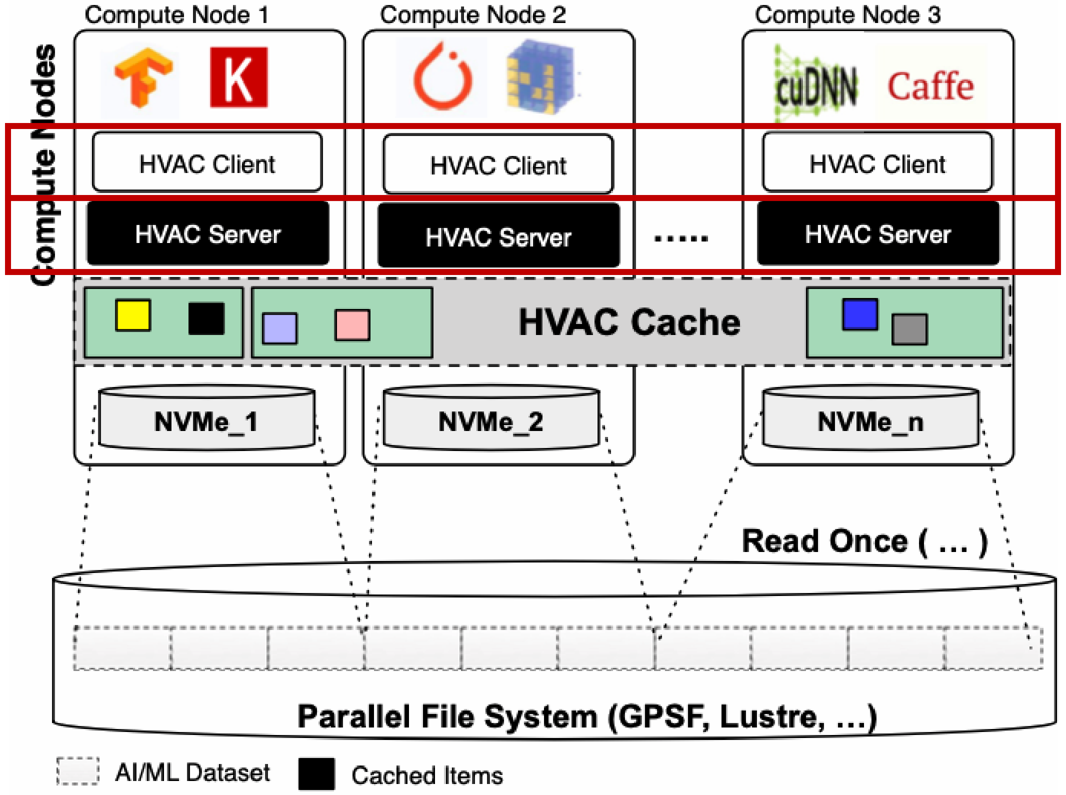
[2] Y. Zhu, "Entropy-Aware I/O Pipelining for Large-Scale Deep Learning on HPC Systems," Proceedings of the IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2018.
[3] A. Khan, A. K. Paul, C. Zimmer, S. Oral, S. Dash, S. Atchley, and F. Wang, "HVAC: Removing I/O Bottleneck for Large-Scale Deep Learning Applications," Proceedings of the IEEE International Conference on Cluster Computing (Cluster), 2022.

HVAC: High-Velocity AI Cache

- ***HVAC***, *a transparent read-only caching layer for large-scale supercomputers using node-local NVMe.*
- ***Scalability***
  - Designed to scale across thousands of compute nodes on leadership-class supercomputers like Summit and Frontier.
  - ***Avoids additional metadata bottlenecks*** and storage overhead.
- ***Client-Server Library Architecture***
  - Intercepts <open-read-close> file I/O operations via ***LD_PRELOAD*** using a shared library approach.
  - Data is cached to distributed node-local storage.
  - Utilizes distributed hashing to determine the location of cached content across nodes. → ***No Repeated Access to PFS***

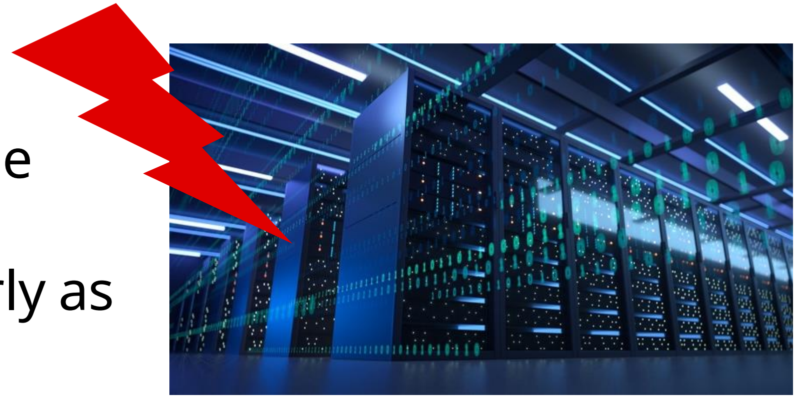
HVAC Overview

- ***HVAC Server***
  - Builds a caching layer on ***node-local fast storage.***
  - Handles system calls forwarded by HVAC clients.
  - Reads files from node-local storage if available, or retrieves files from the PFS and caches them to node-local storage.
- ***HVAC Client***
  - ***Intercepts system calls*** directed to the PFS and ***redirects*** them to the HVAC server.

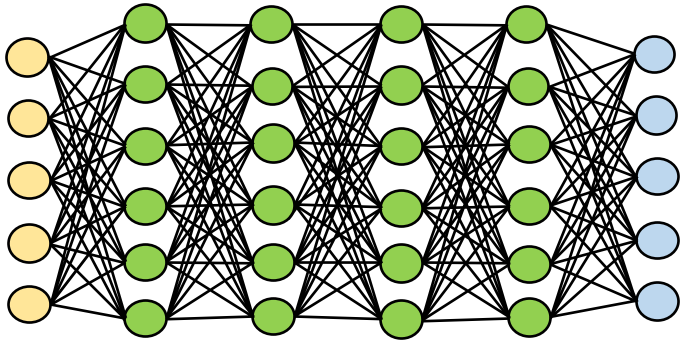


Increasing Node Failures in HPC and AI Workloads

- ***Node Failure Rates Rise with Complexity***
  - Larger, more complex HPC systems have higher node failure rates.
  - Failure rates scale with system size, increasing linearly as more processors are added[4] .



- ***Intensive Workloads Increase Failure Probability***
  - High-demand tasks, such as large-scale deep learning jobs, also increase the risk of failure.
  - Running multiple nodes for deep learning exacerbates the likelihood of failure event.



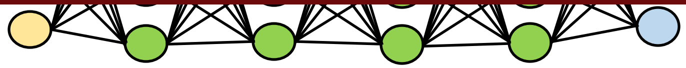
[4] B. Schroeder and G. A. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337-350, Oct.-Dec. 2010.

Increasing Node Failures in HPC and AI Workloads

- ***Node Failure Rates Rise with Complexity***

However, HVAC currently ***lacks fault tolerance support***, which limits its resilience in handling node failures.

the likelihood of failure event.



[4] B. Schroeder and G. A. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337-350, Oct.-Dec. 2010.

Failures in HVAC

- Even a single node failure can halt the entire training process, despite fault-tolerance support in the DL framework.
  - E.g. Elastic Scaling - *Horovod Elastic Run, MPI ULFM*
- This happens because I/O flows are controlled by HVAC.

→ The job must be restarted.

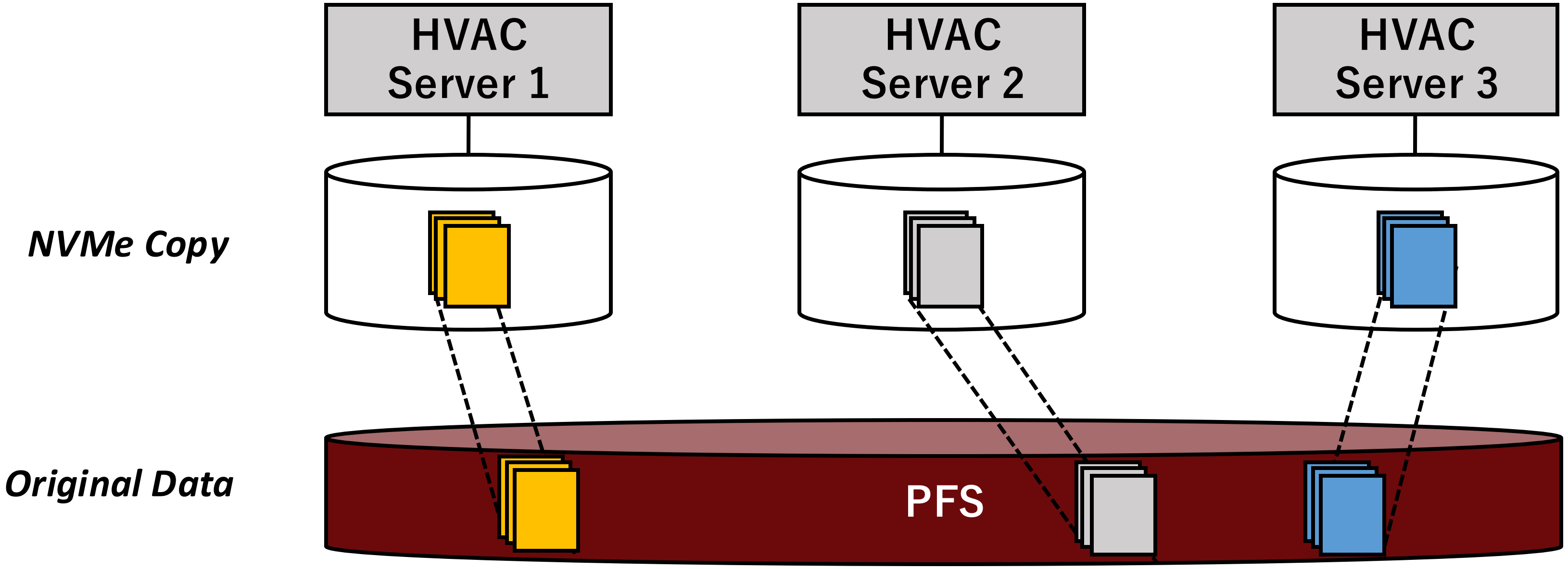
Failures in HVAC

- Even a single node failure can halt the entire training process,

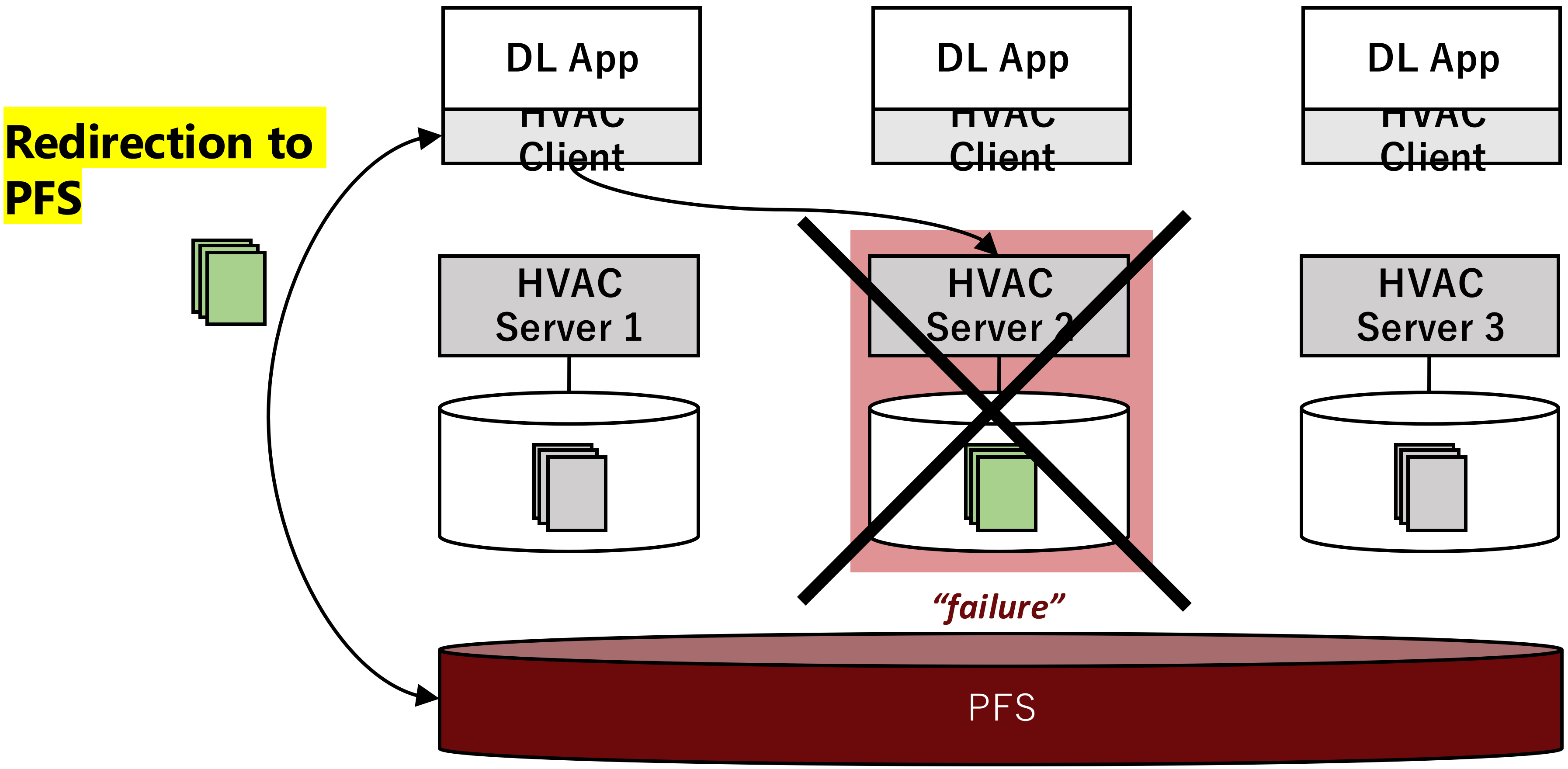
Therefore, it is crucial to ***ensure fault tolerance*** in the ***HVAC layer*** to prevent training interruptions!

Naïve Approach

- Because HVAC functions as a caching layer, the original data resides in the PFS.
- I/O requests to failed nodes → redirected to PFS!

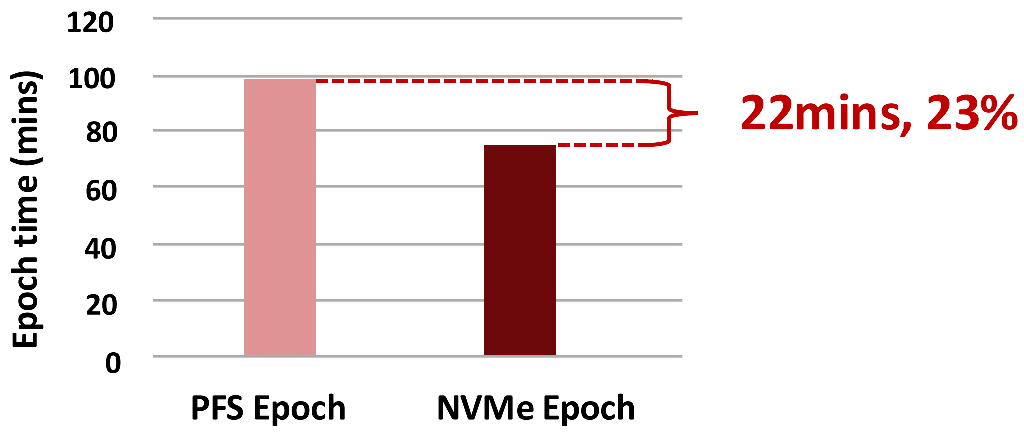


I/O Redirection to PFS



Limitations of PFS I/O Redirection

“Frequent future PFS access”

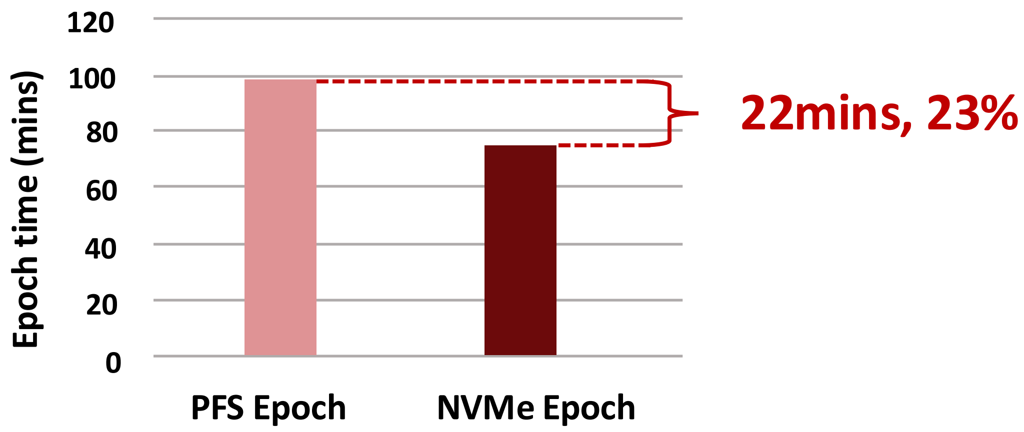


- 22 mins per epoch x 5 epochs → ***nearly 2 hours***
- Partial data reads from PFS still cause delays

* The data is from A. Khan, A. K. Paul, C. Zimmer, S. Oral, S. Dash, S. Atchley, and F. Wang, "HVAC: Removing I/O Bottleneckfor Large-Scale Deep Learning Applications," Proceedings of the IEEE International Conference on Cluster Computing (Cluster), 2022.

Limitations of PFS I/O Redirection

"Frequent future PFS access"

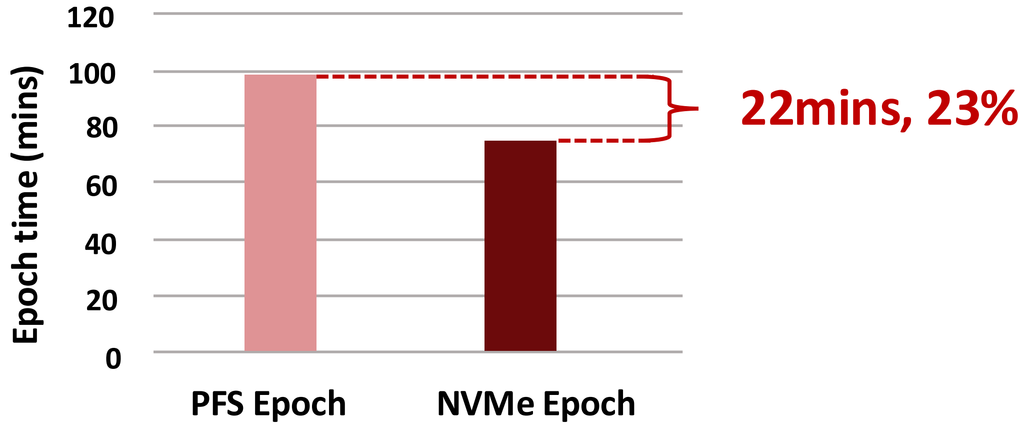


- 22 mins per epoch x 5 epochs → ***nearly 2 hours***
- Partial data reads from PFS still cause delays
- **Straggler Problem:** Even if a few nodes access PFS, deep learning synchronization at each iteration forces other nodes to wait → reducing parallelism and scalability.

* The data is from A. Khan, A. K. Paul, C. Zimmer, S. Oral, S. Dash, S. Atchley, and F. Wang, "HVAC: Removing I/O Bottleneckfor Large-Scale Deep Learning Applications," Proceedings of the IEEE International Conference on Cluster Computing (Cluster), 2022.

Limitations of PFS I/O Redirection

"Frequent future PFS access"



- 22 mins per epoch x 5 epochs → ***nearly 2 hours***
- Partial data reads from PFS still cause delays
- **Straggler Problem:** Even if a few nodes access PFS, deep learning synchronization at each iteration forces other nodes to wait → reducing parallelism and scalability.
- **Job Time Limitations:** Risk of exceeding pre-defined job time.

* The data is from A. Khan, A. K. Paul, C. Zimmer, S. Oral, S. Dash, S. Atchley, and F. Wang, "HVAC: Removing I/O Bottleneckfor Large-Scale Deep Learning Applications," Proceedings of the IEEE International Conference on Cluster Computing (Cluster), 2022.

Limitations of PFS I/O Redirection

"Re-caching Mechanism"

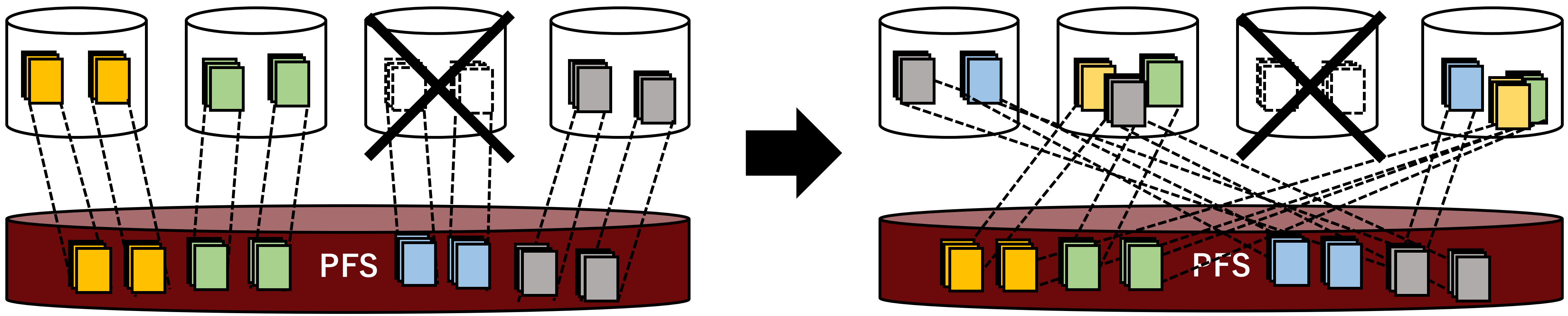
Read from PFS once, then access re-cached data for future requests

- **Job Time Limitations:** Risk of exceeding pre-defined job time.

* The data is from A. Khan, A. K. Paul, C. Zimmer, S. Oral, S. Dash, S. Atchley, and F. Wang, "HVAC: Removing I/O Bottleneckfor Large-Scale Deep Learning Applications," Proceedings of the IEEE International Conference on Cluster Computing (Cluster), 2022.

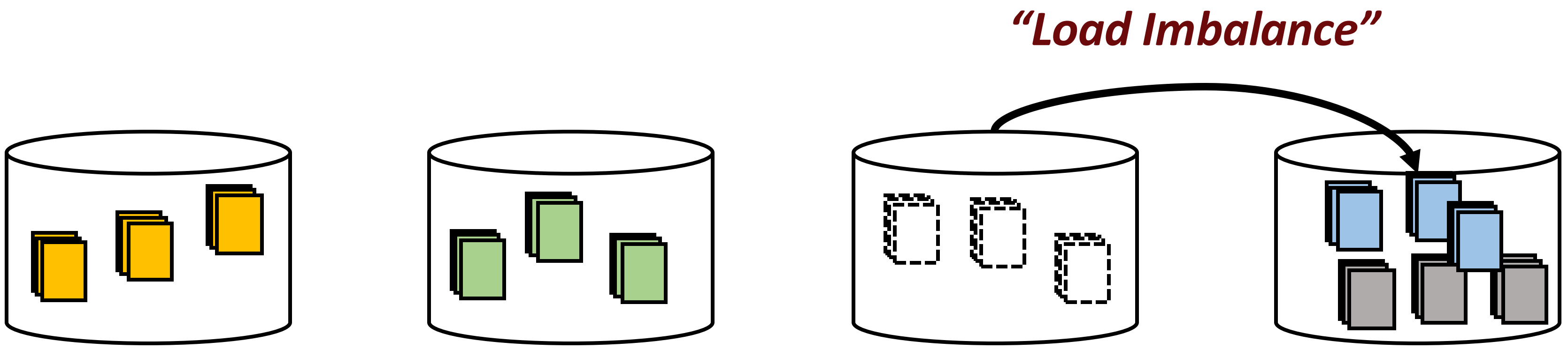
Challenge: Handling Data Redistribution

- ***Original HVAC Implementation - Static Hash Partitioning***
  - Data paths are converted to key values and distributed across nodes using a modulo operation.
  - On node failure, recalculating hash values for ***N−1*** nodes ***causes extensive data redistribution.***



Challenge: Handling Data Redistribution

- ***Additional Hash Functions***
  - Reduces data movement but doesn’t address multiple unpredictable failures.
- ***Range Partitioning***
  - Can handle multiple node failures, but ***balancing data distribution*** remains challenging.



Problem Definition

- How can we ***track data locations*** that change after re-caching?
- How can we ***redistribute*** lost data ***evenly*** across remaining active nodes?

Design & Implementation

Design of FT-HVAC

- ***FT-HVAC:*** An ***I/O accelerated caching framework*** with ***fault tolerance*** for large-scale distributed deep learning.

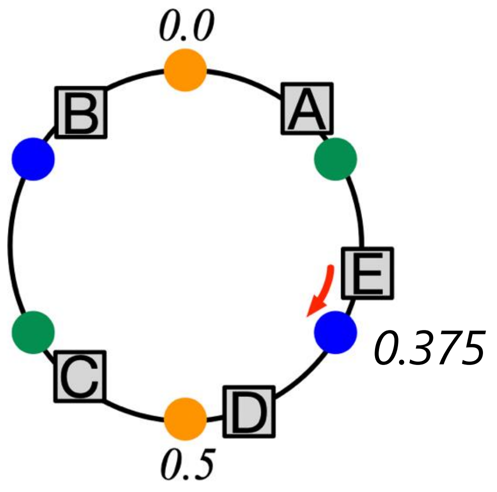
Design of FT-HVAC

- ***FT-HVAC:*** An ***I/O accelerated caching framework*** with ***fault tolerance*** for large-scale distributed deep learning.

1. Enable ***fault tolerance*** in HVAC.
2. Implement ***data recaching*** within the HVAC layer to ensure data availability and quick access during node failures.
3. Achieve ***load-balanced*** data recaching.

Elastic Recaching with Hash Ring

- Hash Ring Mechanism



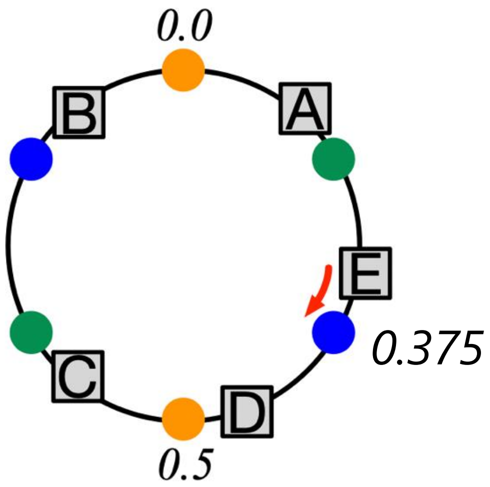
<Before Failure>

Node Table		
Node 0	*0.000*	*0.500*
Node 1	*0.375*	*0.875*
Node 2	*0.125*	*0.625*
	Virtual nodes	

Hash Table	
File A	*0.083427*
⋮	⋮
File E	*0.293853*

Elastic Recaching with Hash Ring

- Hash Ring Mechanism



<Before Failure>

Node Table

Node 0	*0.000*	*0.500*
Node 1	*0.375*	*0.875*
Node 2	*0.125*	*0.625*
	Virtual nodes	

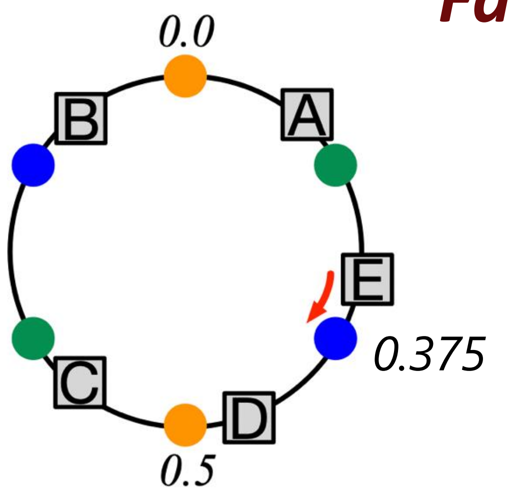
Hash Table

File A	*0.083427*
⋮	⋮
File E	*0.293853*

Elastic Recaching with Hash Ring

- Hash Ring Mechanism

"Failure at Node 1"



<Before Failure>

Node Table

Node 0	*0.000*	*0.500*
Node 1	*0.375*	*0.875*
Node 2	*0.125*	*0.625*

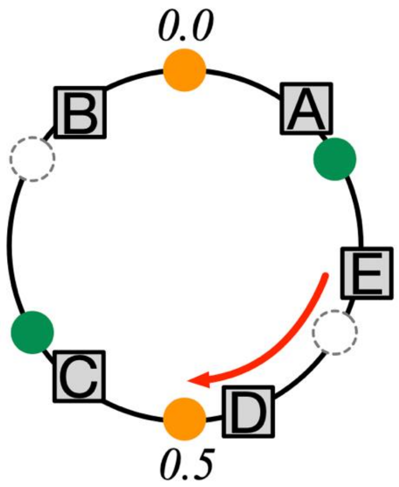
Virtual nodes

Hash Table

File A	*0.083427*
⋮	⋮
File E	*0.293853*

Elastic Recaching with Hash Ring

- Hash Ring Mechanism



<After Failure>

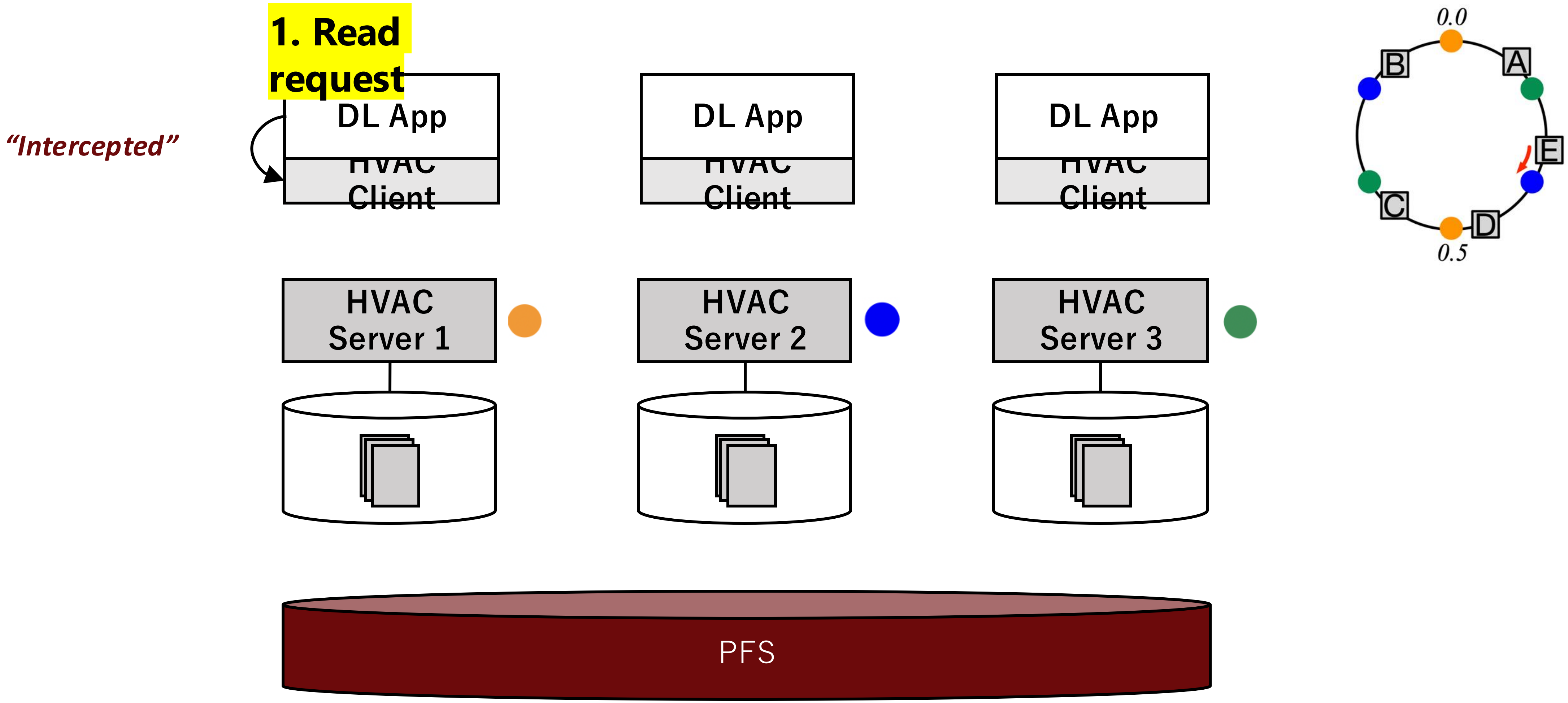
Node Table

Node 0	*0.000*	*0.500*
~~***Node 1***~~	~~*0.375*~~	~~*0.875*~~
Node 2	*0.125*	*0.625*
	Virtual nodes	

Hash Table

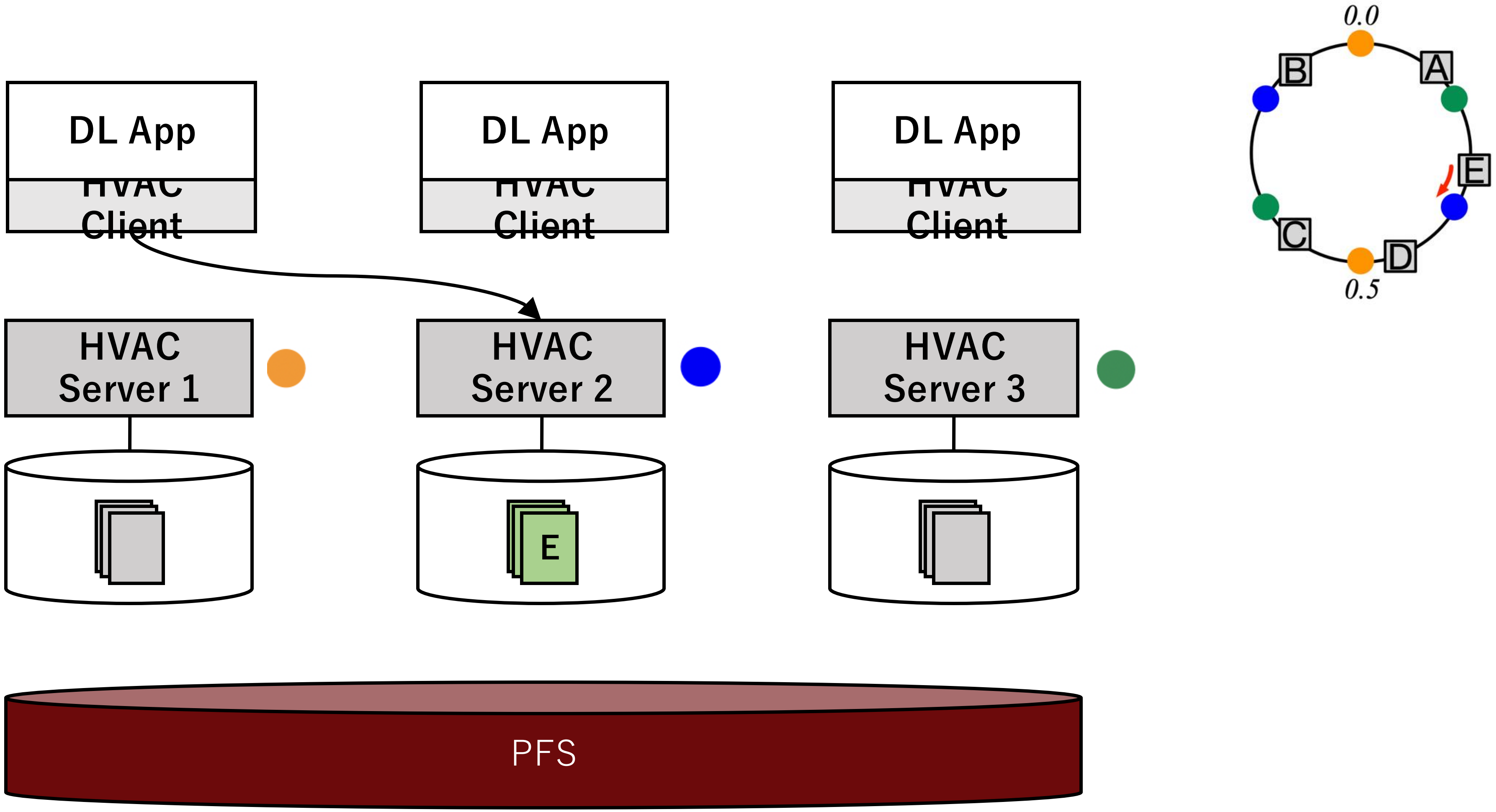
File A	*0.083427*
⋮	⋮
File E	*0.293853*

Elastic Recaching with Hash Ring

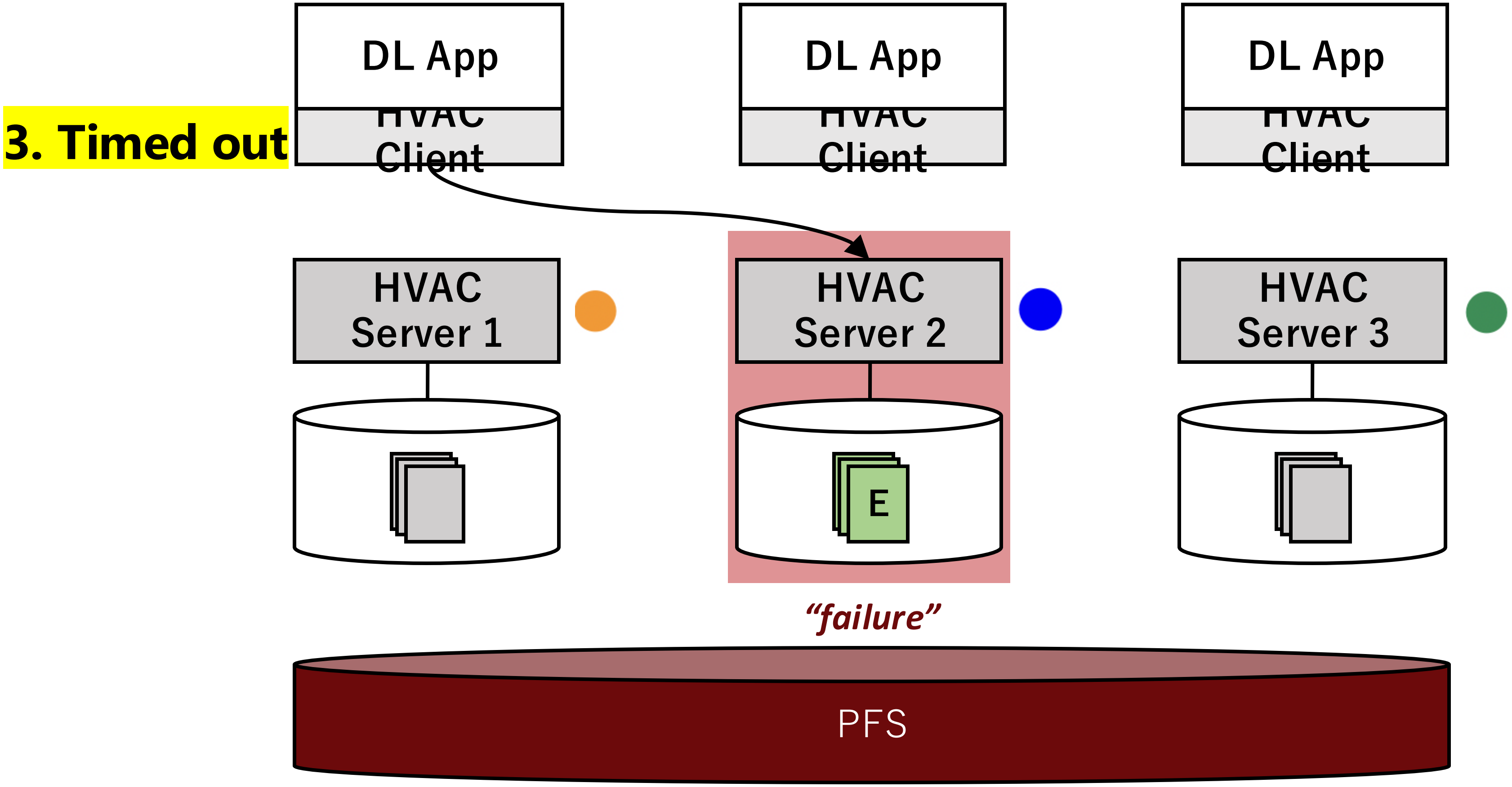


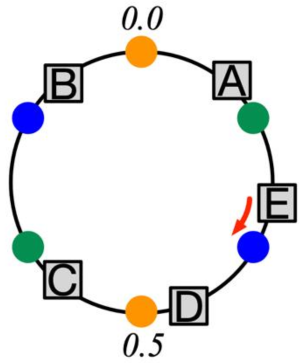
Elastic Recaching with Hash Ring

2. Request to Server

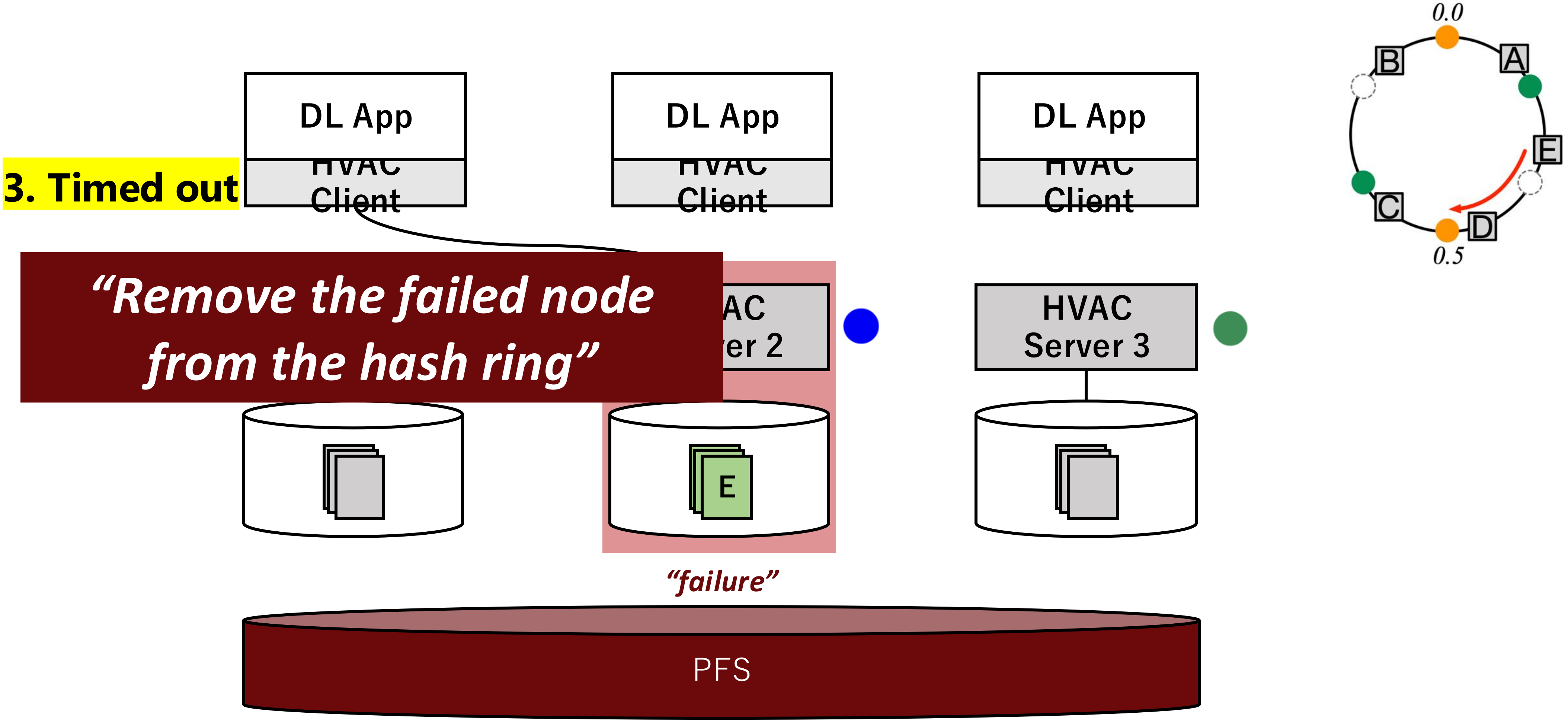


Elastic Recaching with Hash Ring



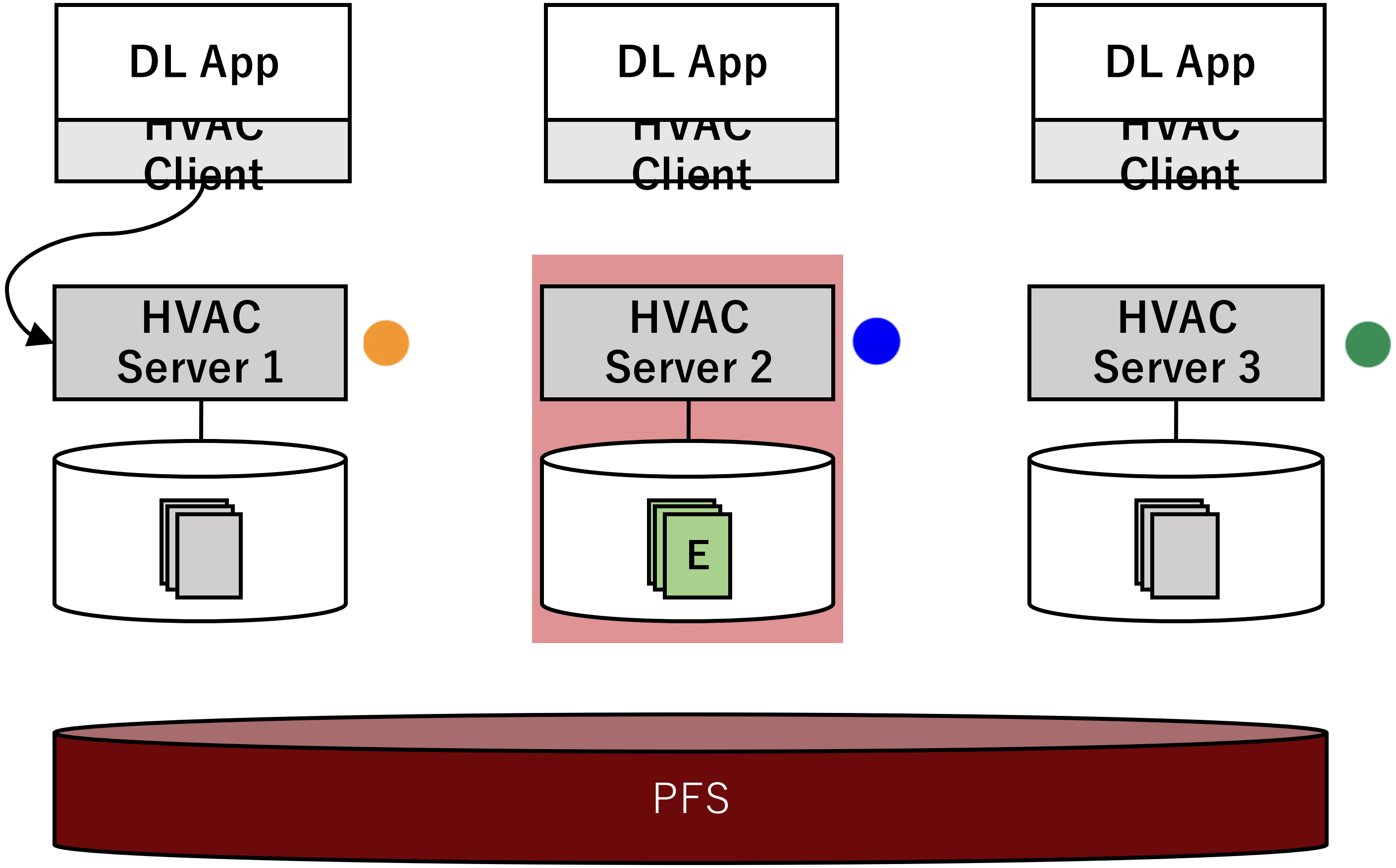


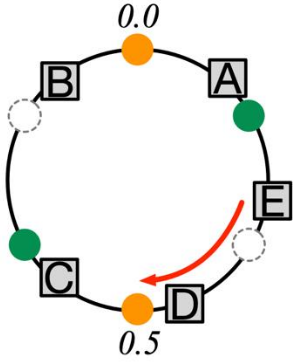
Elastic Recaching with Hash Ring



Elastic Recaching with Hash Ring

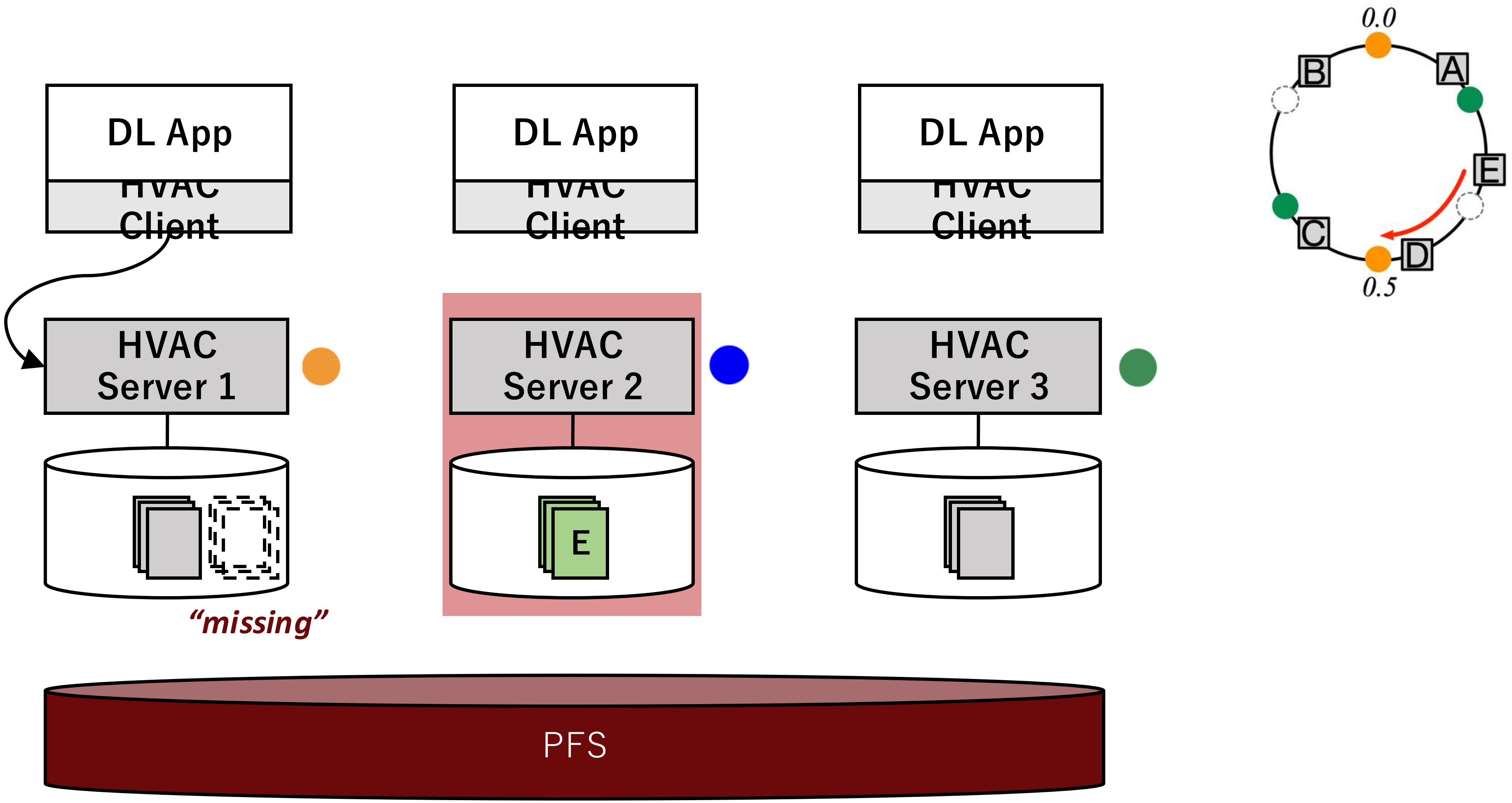
4. Retry hash calculation



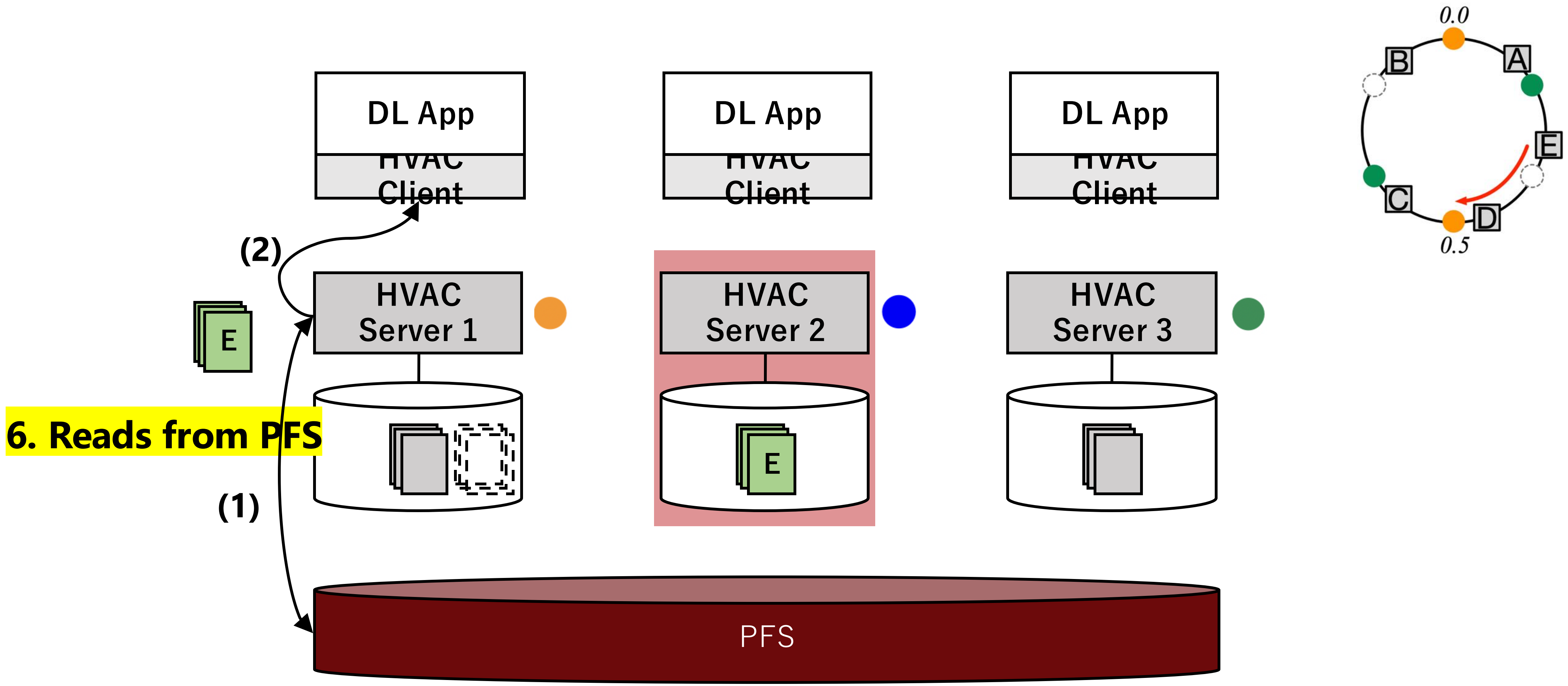


Elastic Recaching with Hash Ring

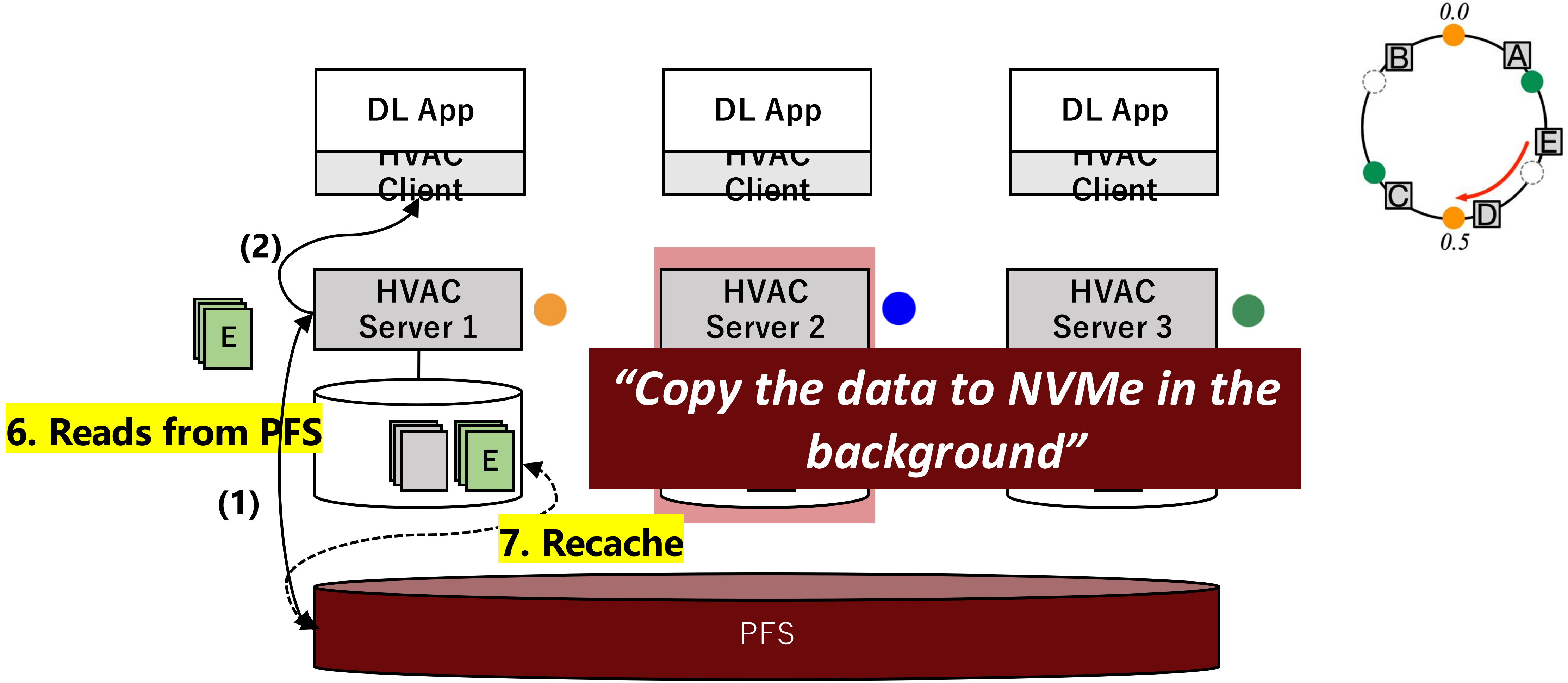
5. Checks the data



Elastic Recaching with Hash Ring

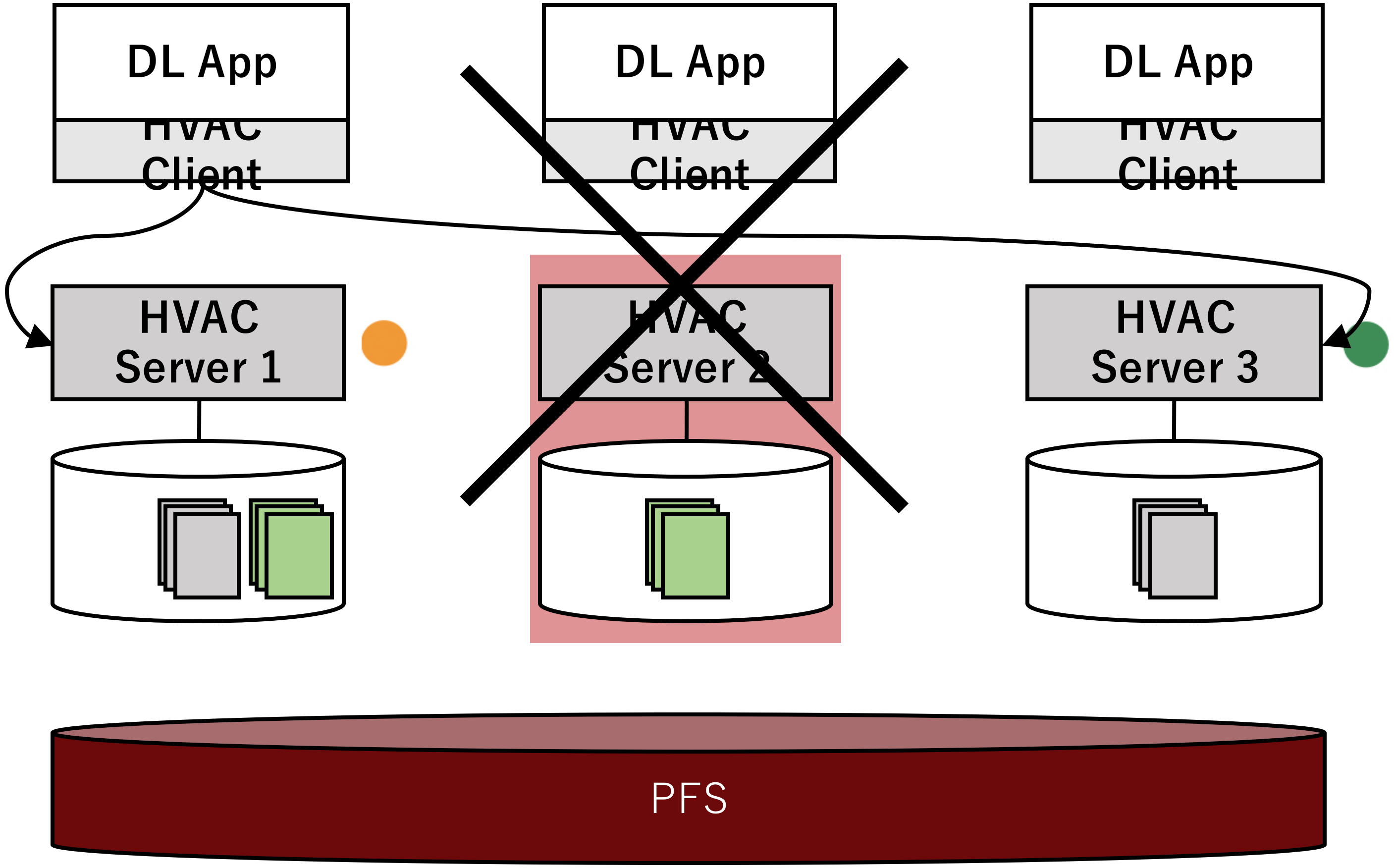


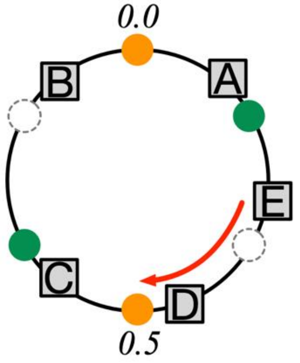
Elastic Recaching with Hash Ring



"Copy the data to NVMe in the background"

Elastic Recaching with Hash Ring





Evaluation

Node Failure Analysis on Frontier

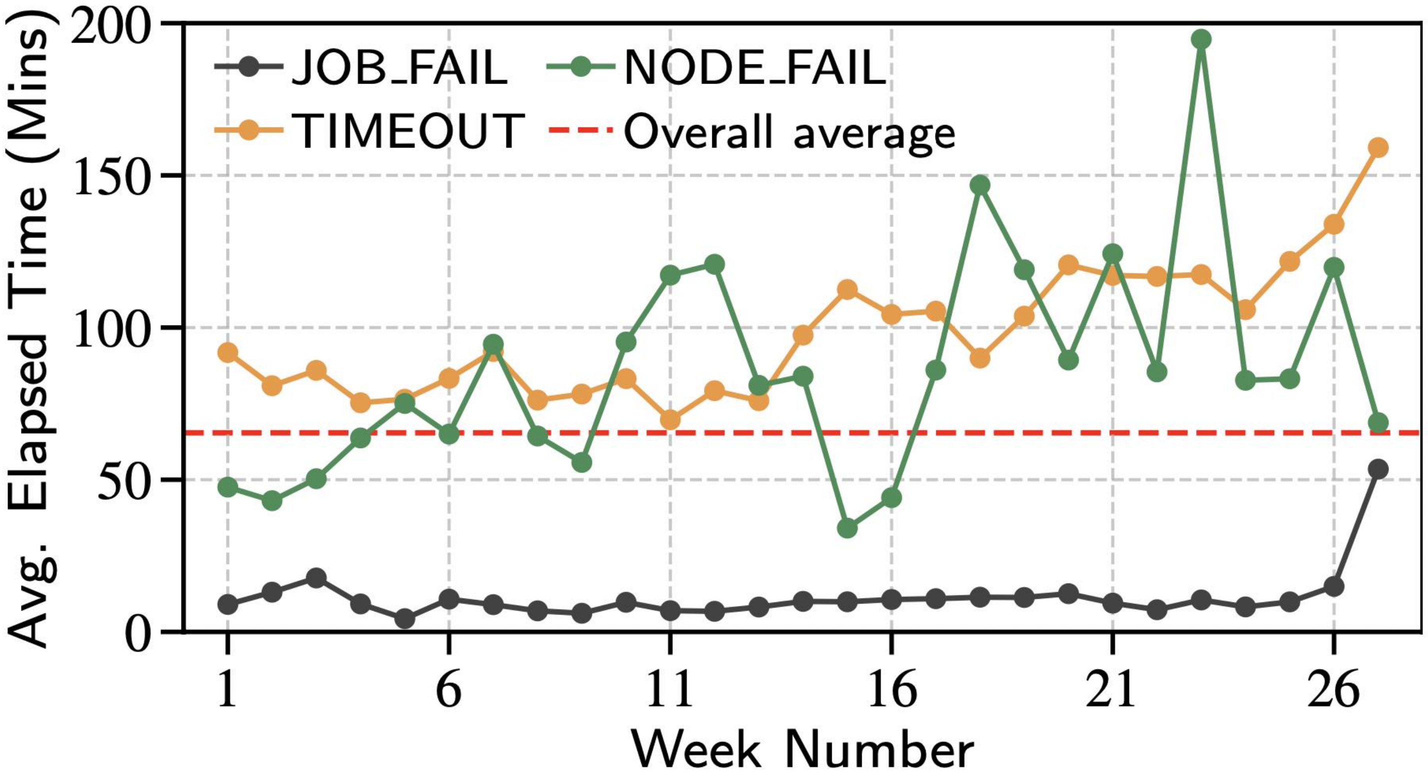
- Analyzed job scheduler logs for six months following the production launch of the ORNL's Frontier cluster.
- Focused on three types of job failures: ***"Job Fail", "Node Fail",*** and ***"Timeout".***

Type	Count	Failure ratio	Overall ratio
Total Jobs	181,933	N/A	100%
Total Failures	45,556	100%	25.04%
Node Fail	1,174	2.58%	0.65%
Timeout	20,464	44.92%	11.25%
Job Fail	23,918	52.50%	13.15%

- ***Job Fail:*** *Due to code errors, data/environment issues, or external malfunctions.*
- ***Node Fail:*** *Caused by hardware, network, software bugs, or overload.*
- ***Timeout:*** *Job exceeded set time limit, often due to complexity or resource/network constraints.*

Node Failure Analysis on Frontier

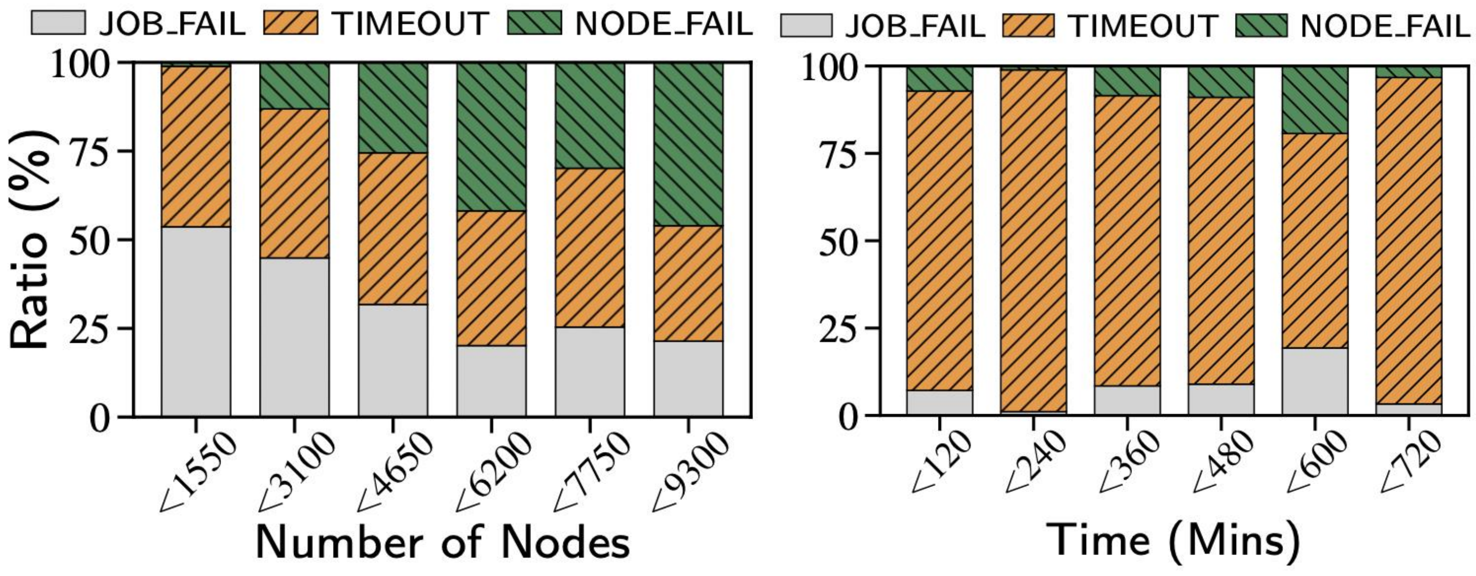
- ***Average Runtime*** of ***Failed Jobs*** on Frontier



- *Failed jobs typically run for an average of* ***over 1 hour****, sometimes reaching* ***2-3 hours.***
- *Long-running job failures →* ***significant loss of computing resources and time.***
- *Job failures have occurred* ***consistently*** *on a weekly basis.*

Node Failure Analysis on Frontier

- Relationships between ***types of job failures*** and system variables.



(a) Node Scaling Impact

(b) Runtime Impact

- *(a) As the **number of nodes increases**, the rate of **"Node Failures" also rises.***
  - *E.g. With 7,750–9,300 nodes, "Node Failures" are **46.04%** of failures; including "Timeouts," they total 78.60%.*
- *(b) Execution time does not significantly impact the proportion of failure types.*

Node Failure Analysis on Frontier

- Relationships between ***types of job failures*** and system variables.

- *(a) As the* ***number of nodes increases****, the rate of* ***"Node Failures" also rises.***
  - *E.g. With 7,750–9,300 nodes, "Node Failures" are* ***46.04%*** *of failures; including "Timeouts," they total 78.60%.*

Failures are ***highly frequent*** in large-scale HPC systems,

Software systems that are not designed to handle such failures are especially vulnerable, often resulting in significant, unavoidable losses

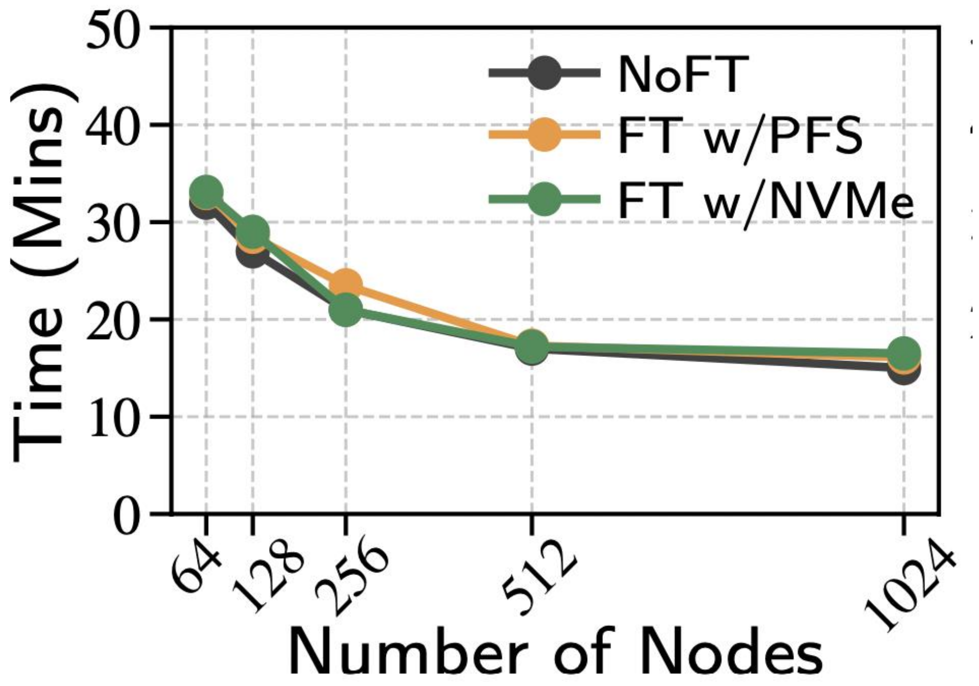
Experimental Setup

Attribute	Description
Supercomputer	Frontier
CPU	AMD Trento EPYC 7A53
GPU	8 x MI250X AMD with 64 GiB HBM
Memory Capacity	512 GiB DDR4
Node-local Storage	2 x 1.9 TB Samsung PM9A3 M.2 NVMe

- ***Application:*** Cosmoflow- MLPerf HPC v0.5 benchmark
- ***Framework:*** Horovod Elastic Run
- ***Dataset:*** 1.3TB cosmoUniverse dataset from NERSC ExaLearn group (524,288 training samples, 65,536 validation samples).
- ***File System:*** Orion (Lustre).
- ***Training Setup:*** 5 epochs, with 5 random failures injected after the first epoch.

Overhead Analysis

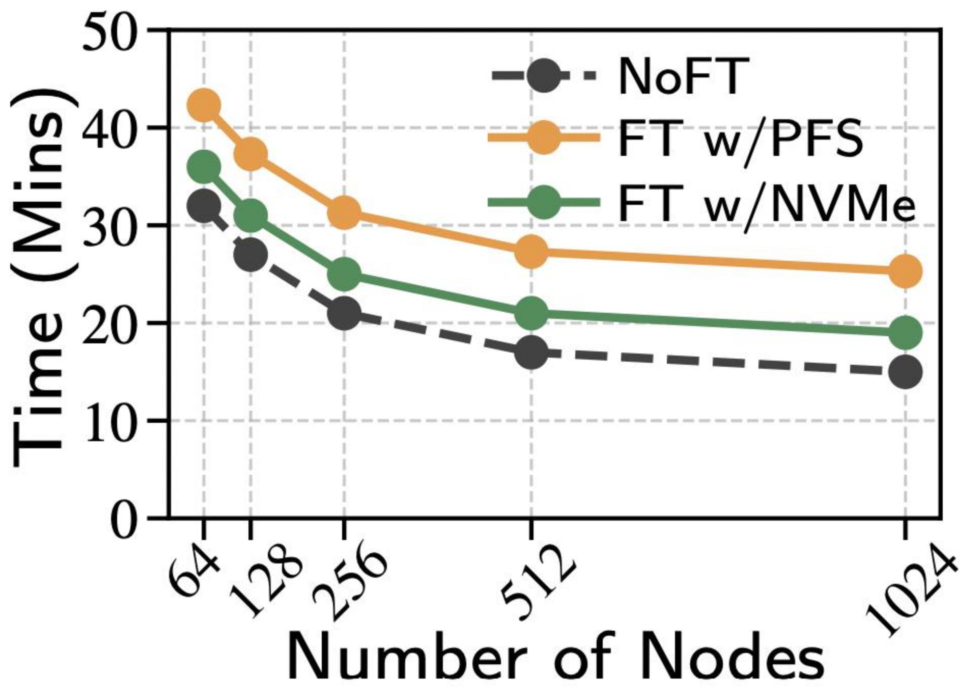
- Comparison between the original HVAC (NoFT) and two fault-tolerant approaches (FT w/PFS, FT w/NVMe) without any failure events.



- Overhead was minimal, with a maximum of 1-minute increase.
- Overhead resulted from additional data structures and conditional checks for the fault detection algorithm.

Overall Performance

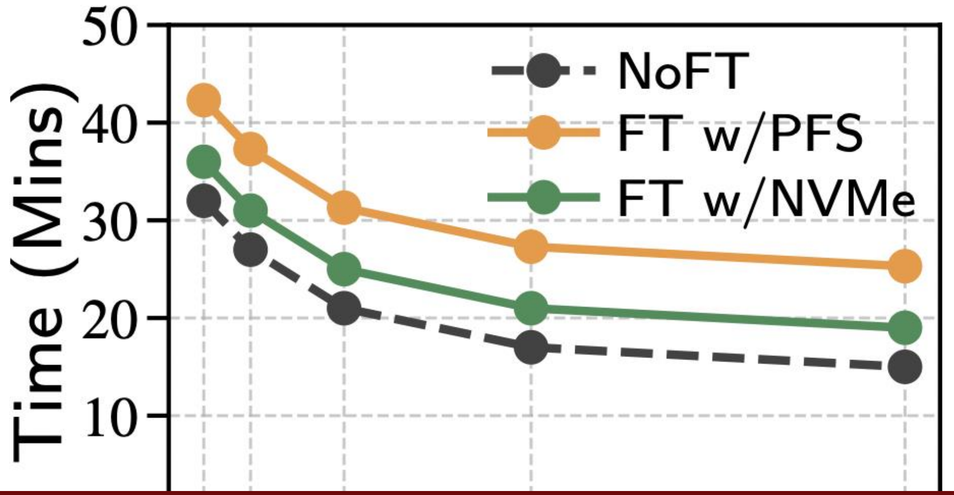
- Performance evaluation with failure events.



- Comparison to No Failure Scenario
- ***FT w/PFS:***
  - 64 nodes: 32.2% increase in training time.
  - 1024 nodes: 68.7% increase in training time.
- ***FT w/NVMe:***
  - 64 nodes: 12.5% increase in training time.
  - 1024 nodes: 26.7% increase in training time.

Overall Performance

- Performance evaluation with failure events.



- Comparison to No Failure Scenario
- ***FT w/PFS:***
  - 64 nodes: 32.2% increase in training time.
  - 1024 nodes: 68.7% increase in training time.
- ***FT w/NVMe:***
  - 64 nodes: 12.5% increase in training time.

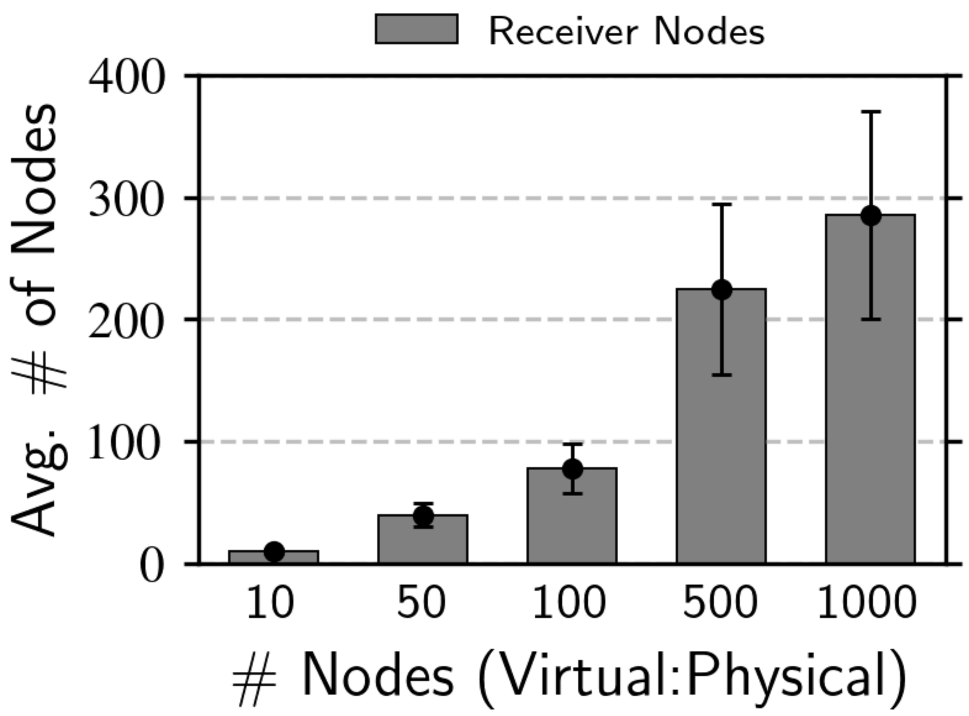
FT w/ NVMe offers reduction in training time compared with FT w/ PFS, from ***14.9% to 24.9%*** as the number of nodes increases, as ***NVMe accesses the PFS only once*** following a failure, while FT with PFS requires repeated accesses.

Load Balance Analysis

- In a 1024-node configuration with ***100 virtual nodes*** per physical node, a single node failure redistributes data to an average of ***80 nodes.***

Load Balance Analysis

- Analysis of varying virtual nodes per physical node to assess data distribution during failures.



- ***Simulation Setup:*** Conducted 500 simulations considering a 1024-node configuration.
- The Receiver Node metric represents the number of nodes receiving redistributed data.
- Increasing the number of virtual nodes improves data distribution but efficiency plateaus beyond 500.

Conclusion

- Node failures are ***common*** in leading-edge supercomputers.
- Such failures pose a ***high risk to DL applications*** on large-scale systems.
- FT-HVAC is a ***fault-tolerant, I/O-accelerated caching framework*** for distributed DL.
- FT-HVAC has demonstrated effective fault handling across 1024 nodes.
- The Elastic Recaching approach reduced training time by ***up to 24.9%*** compared to the I/O redirection method, while maintaining effective load balancing.

Questions?

- Seoyeong Lee / sylee2519@gmail.com
- Data-Intensive Computing & Systems Laboratory / https://discos.sogang.ac.kr

<Camera-ready paper>