Building an Elastic Query Engine on Disaggregated Storage

Midhul

Justin

Rachit

Dan

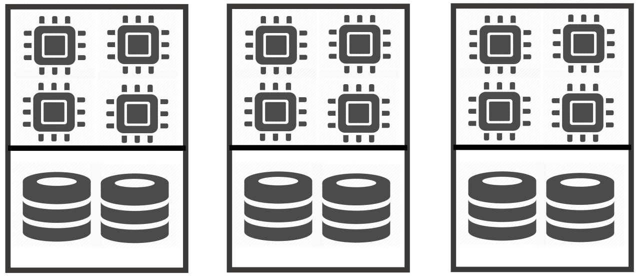
Ashish

Thierry

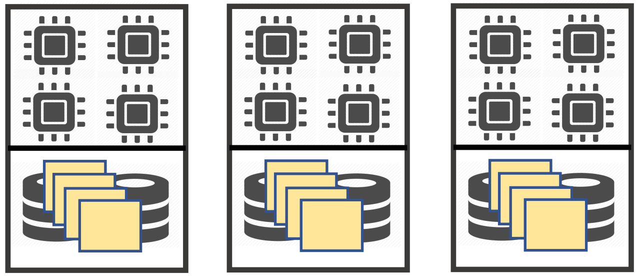
Cornell University

snowflake

Traditional Shared-Nothing Architectures

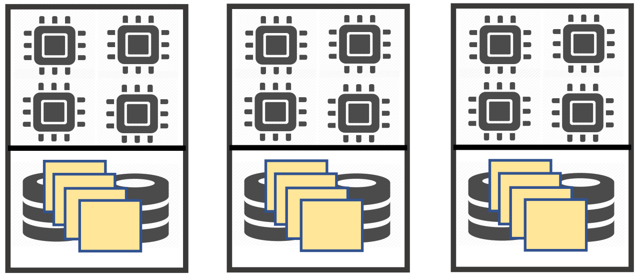


Traditional Shared-Nothing Architectures



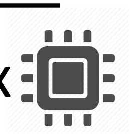
- Data partitioned across servers
- Each server handles its own partition

Traditional Shared-Nothing Architectures

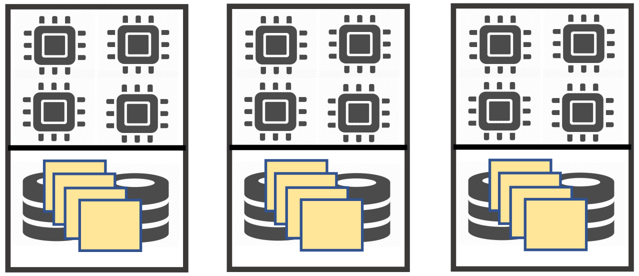


- Data partitioned across servers
- Each server handles its own partition

Workload A

50x  20x 

Traditional Shared-Nothing Architectures



- Data partitioned across servers
- Each server handles its own partition

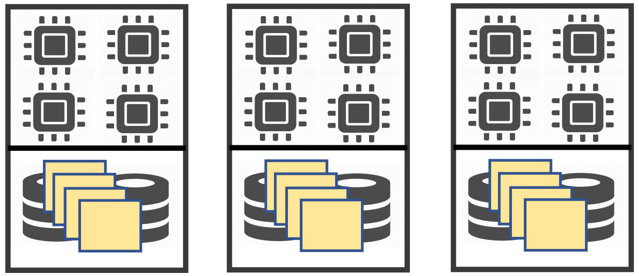
Workload A

50x 20x

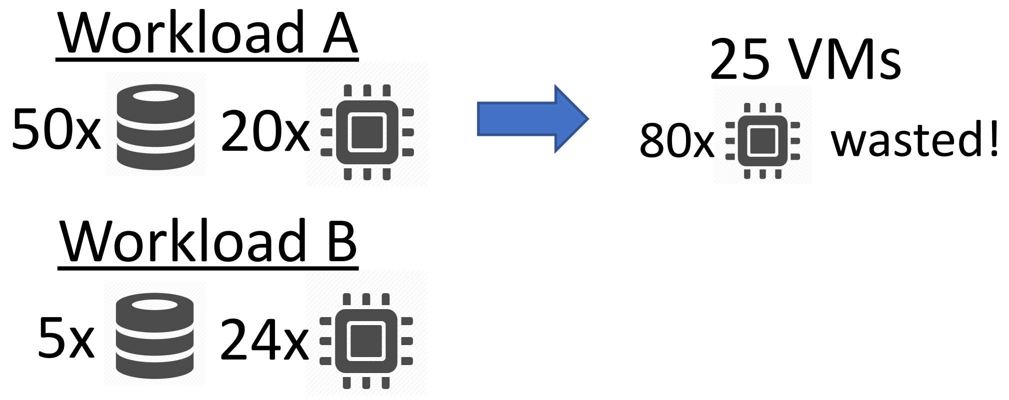
25 VMs

80x wasted!

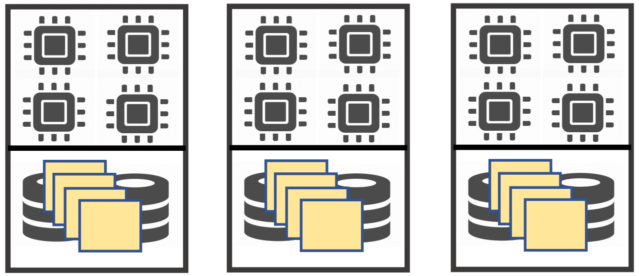
Traditional Shared-Nothing Architectures



- Data partitioned across servers
- Each server handles its own partition



Traditional Shared-Nothing Architectures



- Data partitioned across servers
- Each server handles its own partition

<u>Workload A</u>

50x 20x → 25 VMs

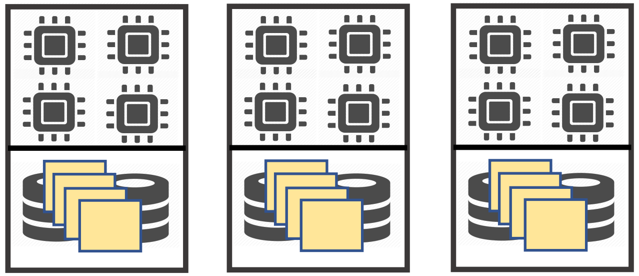
80x wasted!

<u>Workload B</u>

5x 24x → 6 VMs

7x wasted!

Traditional Shared-Nothing Architectures



- Data partitioned across servers
- Each server handles its own partition

Workload A

50x 20x → 25 VMs

80x wasted!

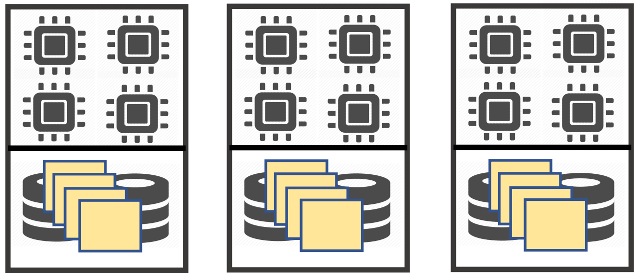
Workload B

5x 24x → 6 VMs

7x wasted!

Hardware-workload mismatch!

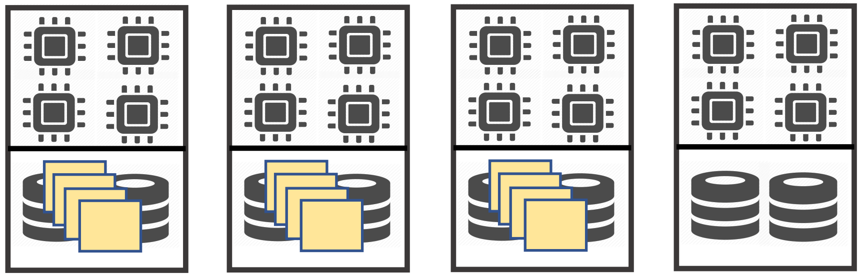
Traditional Shared-Nothing Architectures



- Data partitioned across servers
- Each server handles its own partition

Hardware-workload mismatch!

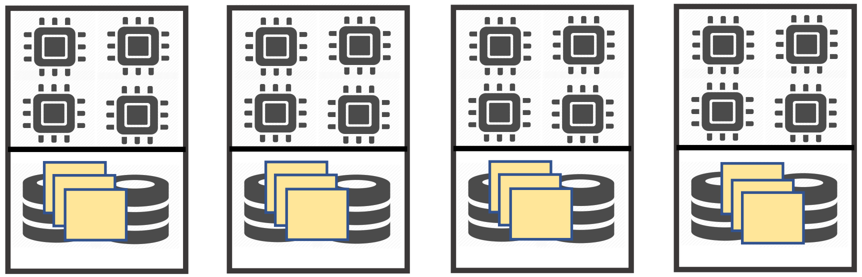
Traditional Shared-Nothing Architectures



- Data partitioned across servers
- Each server handles its own partition

Hardware-workload mismatch!

Traditional Shared-Nothing Architectures

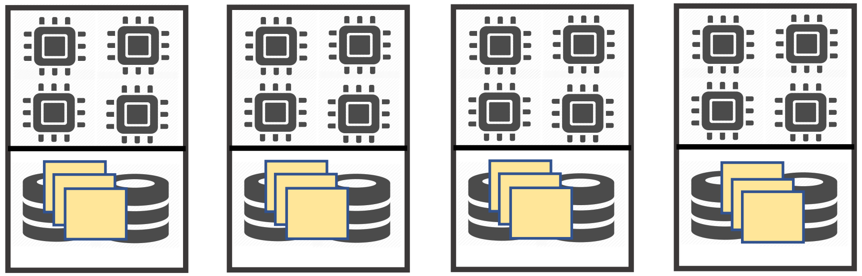


- Data partitioned across servers
- Each server handles its own partition

Hardware-workload mismatch!

Data re-shuffle during elasticity!

Traditional Shared-Nothing Architectures



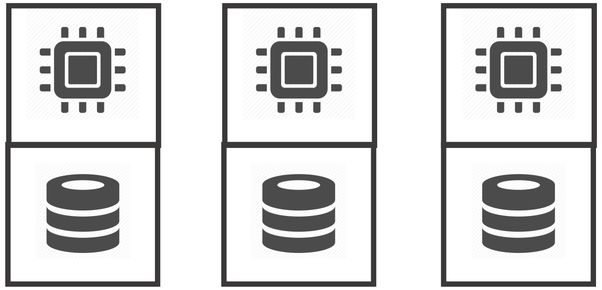
- Data partitioned across servers
- Each server handles its own partition

Hardware-workload mismatch!

Data re-shuffle during elasticity!

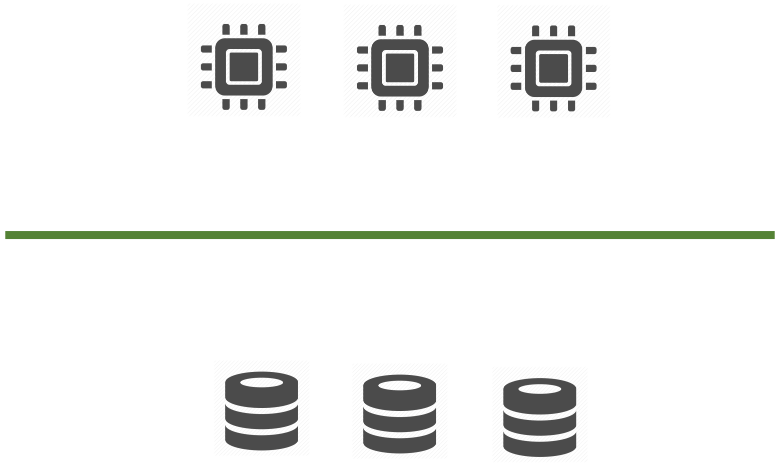
Fundamental issue in shared-nothing architectures: Tight coupling of compute & storage

Query Engines on Disaggregated Storage



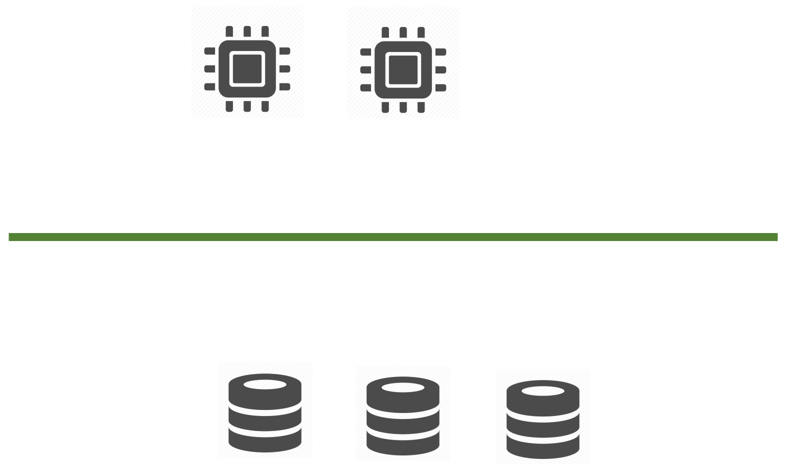
- **Decouple compute** and **persistent storage**
- Independent scaling of resources

Query Engines on Disaggregated Storage



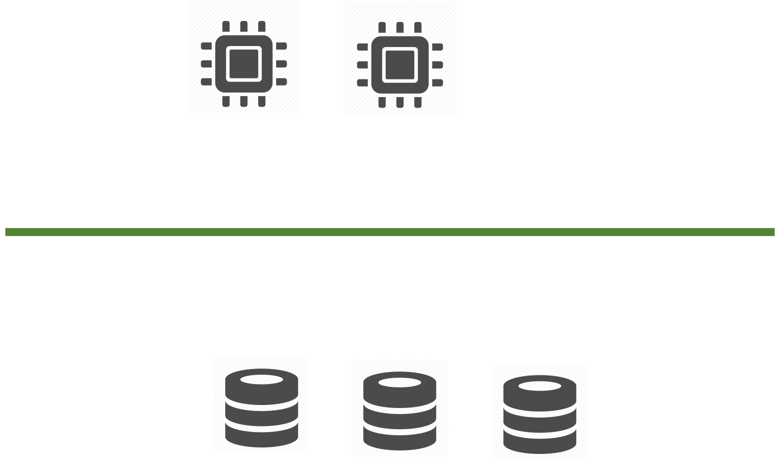
- **Decouple compute** and **persistent storage**
- Independent scaling of resources

Query Engines on Disaggregated Storage



- **Decouple compute** and **persistent storage**
- Independent scaling of resources

Query Engines on Disaggregated Storage



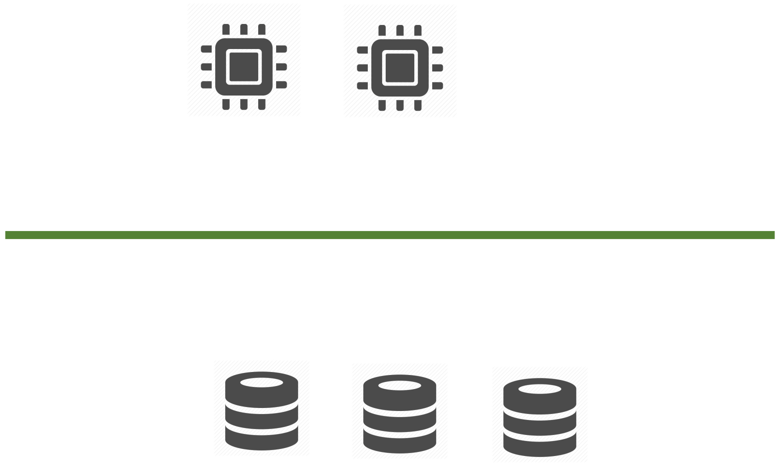
- **Decouple compute** and **persistent storage**
- Independent scaling of resources

Design aspects

Data-driven insights

Future Directions

Query Engines on Disaggregated Storage



- **Decouple compute** and **persistent storage**
- Independent scaling of resources

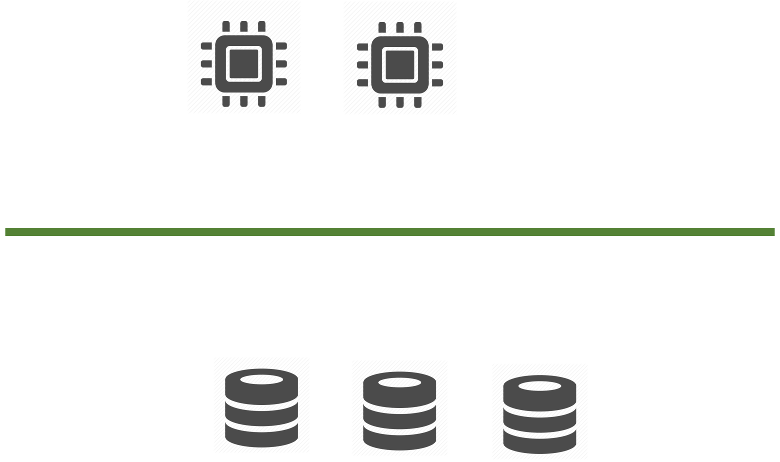
Design aspects

Data-driven insights

Future Directions

- Warehousing as a service
- In production for over 5 years
- 1000s of customers, millions of queries / day

Query Engines on Disaggregated Storage



- **Decouple compute** and **persistent storage**
- Independent scaling of resources

Design aspects

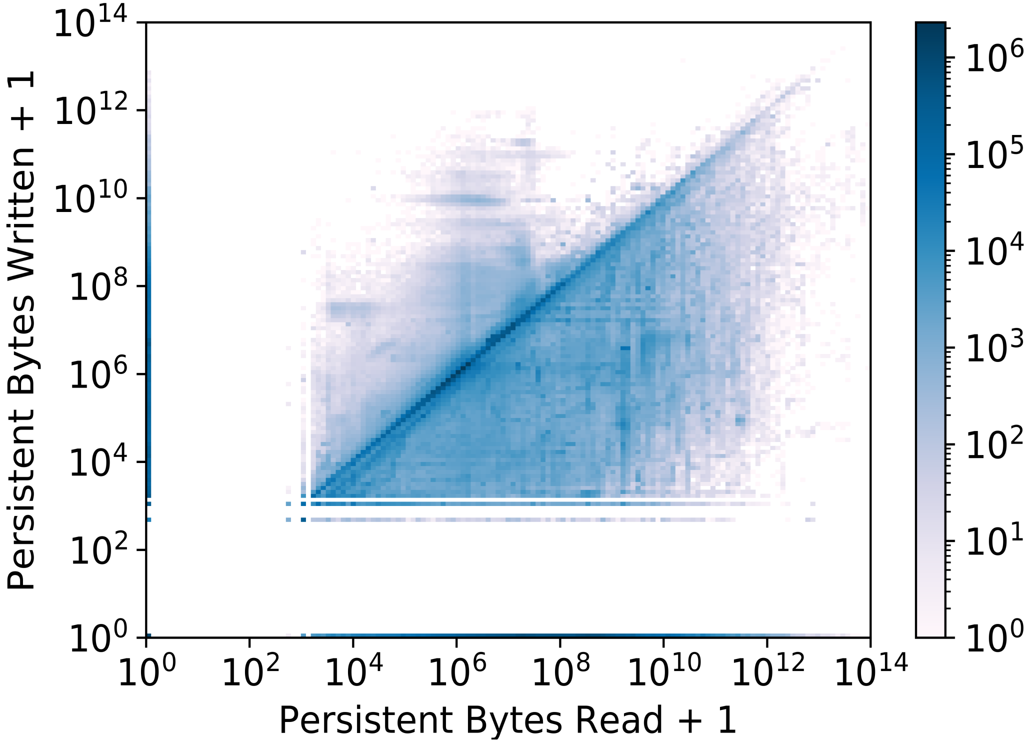
Data-driven insights

Future Directions

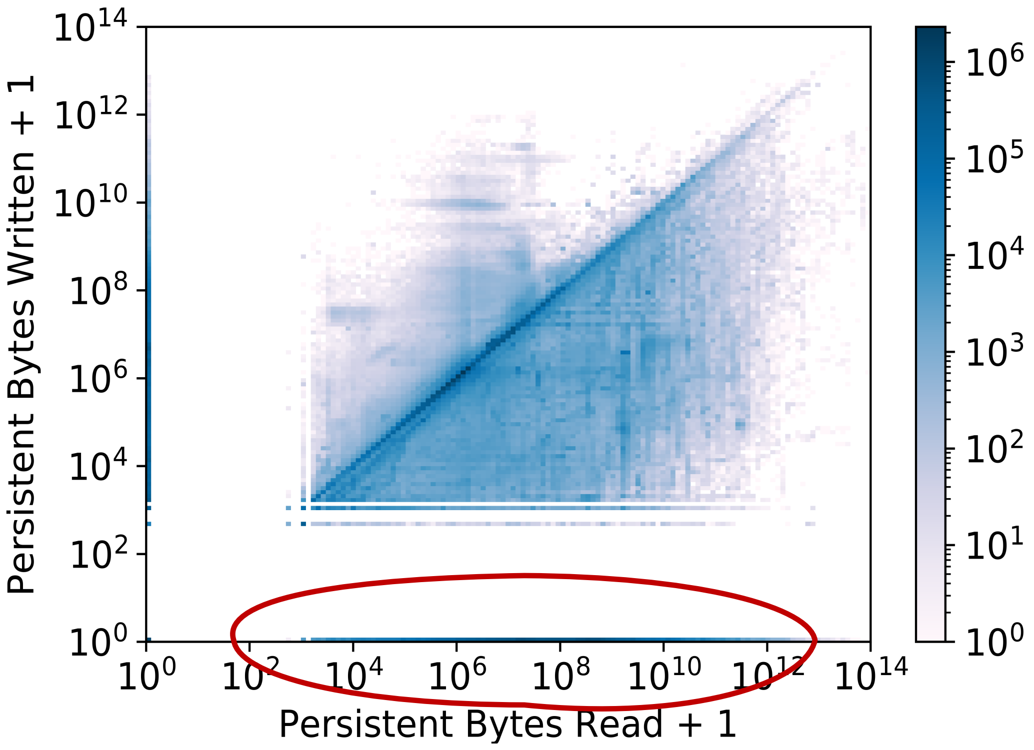
- Warehousing as a service
- In production for over 5 years
- 1000s of customers, millions of queries / day

Statistics from 70 million queries over 14 day period

Diversity of Queries

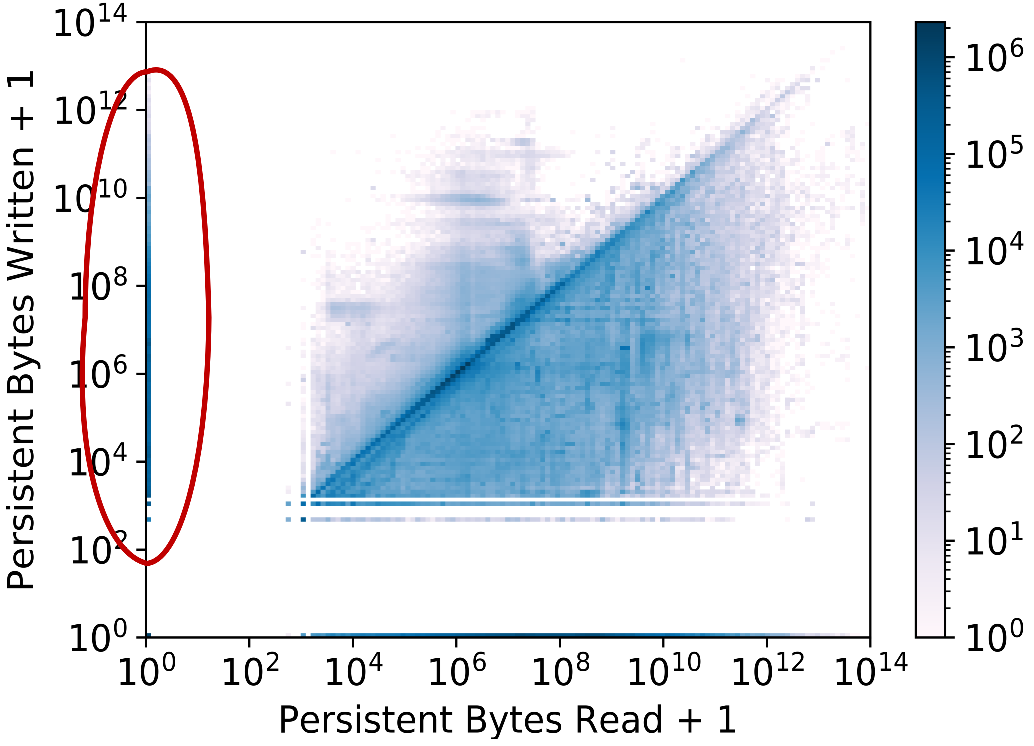


Diversity of Queries



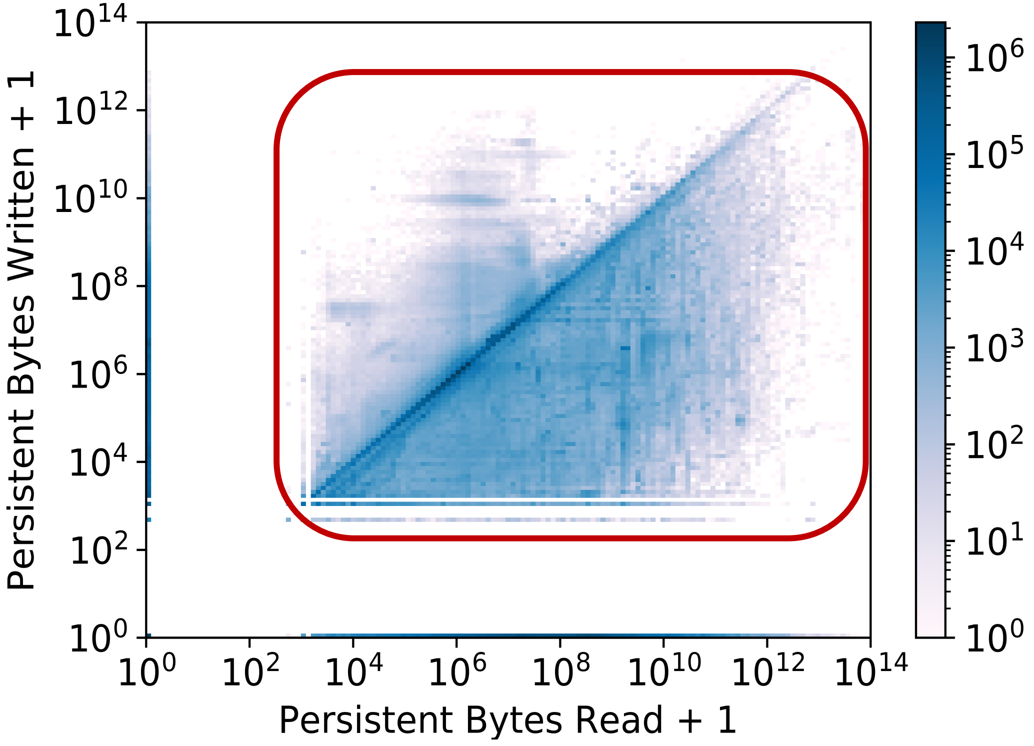
- Read-Only -> 28%

Diversity of Queries



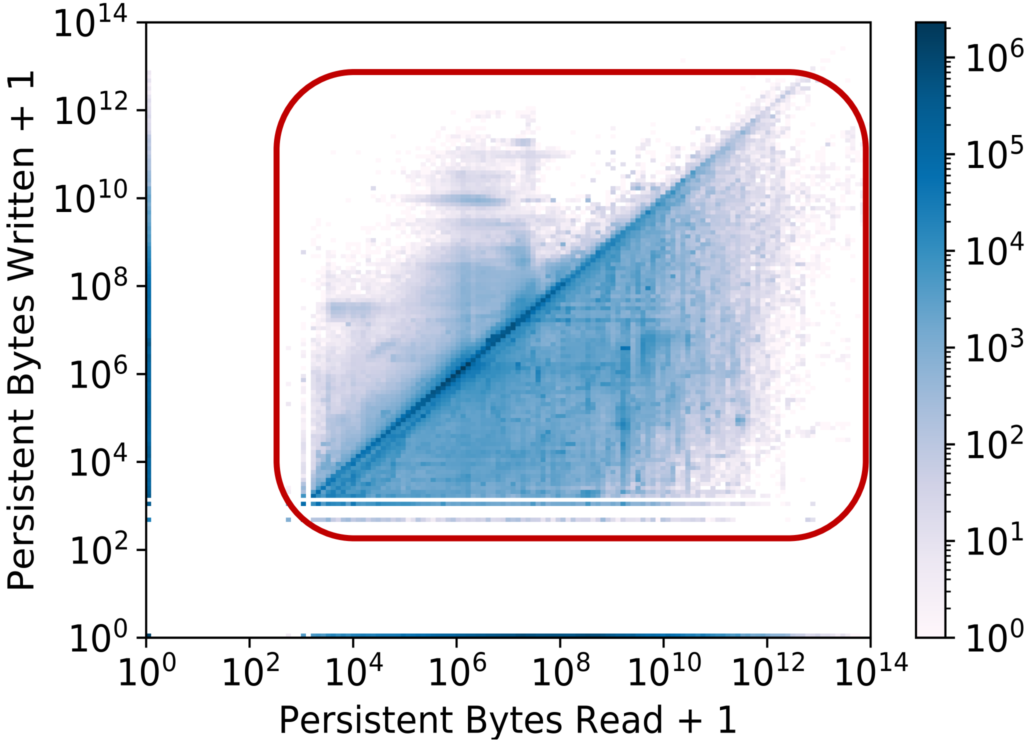
- Read-Only -> 28%
- Write-Only -> 13%

Diversity of Queries



- Read-Only -> 28%
- Write-Only -> 13%
- Read-Write -> 59%

Diversity of Queries

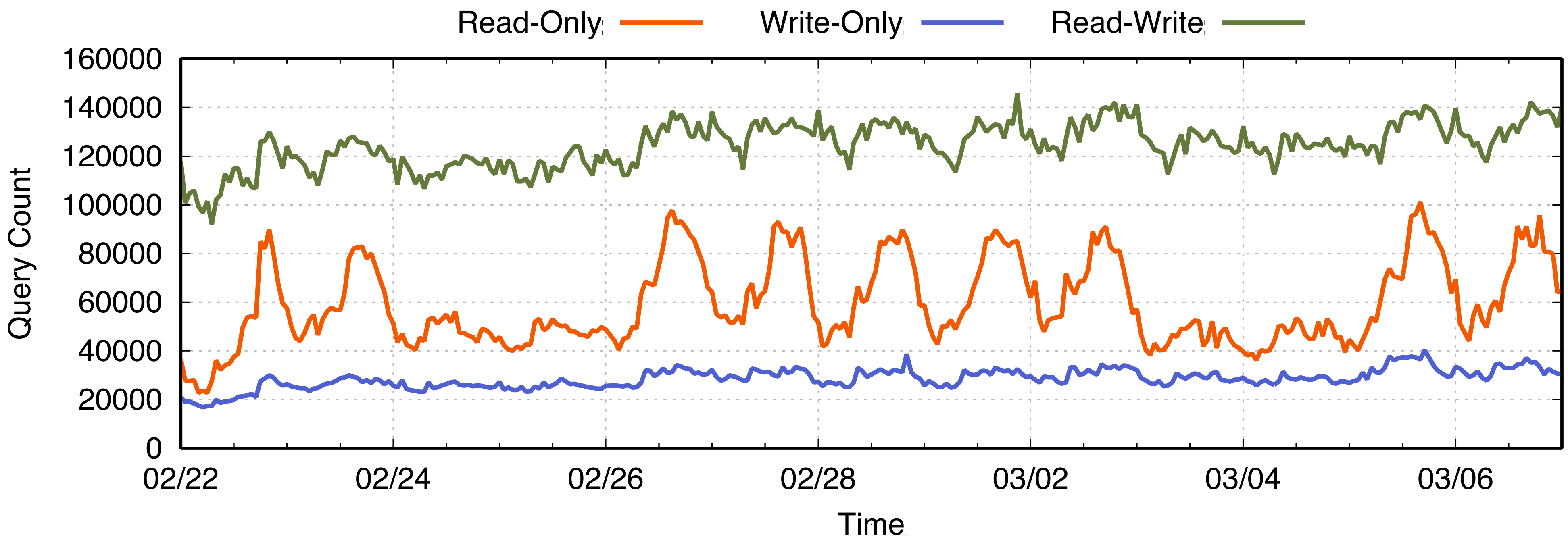


- Read-Only -> 28%
- Write-Only -> 13%
- Read-Write -> 59%

Three distinct query classes

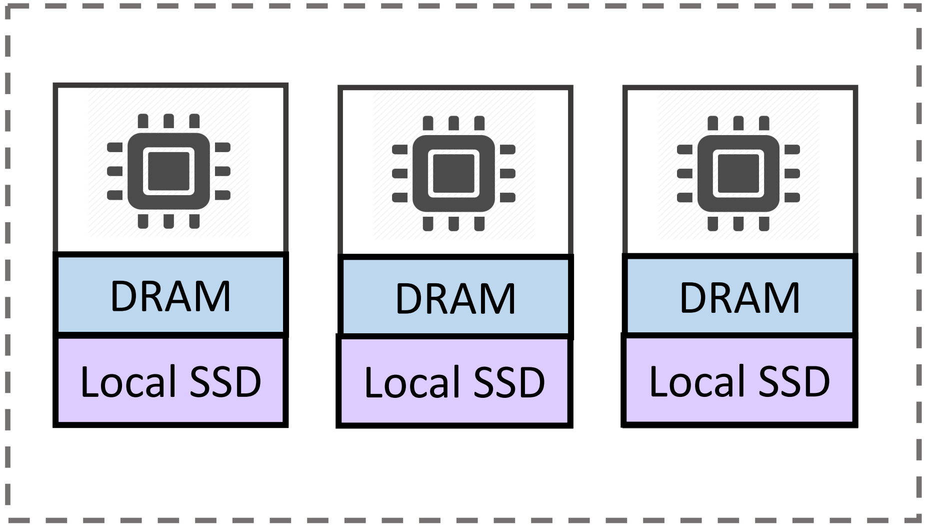
Persistent data read/written varies over several orders of magnitude within each class

Query distribution over time



Read-Only query load varies significantly over time

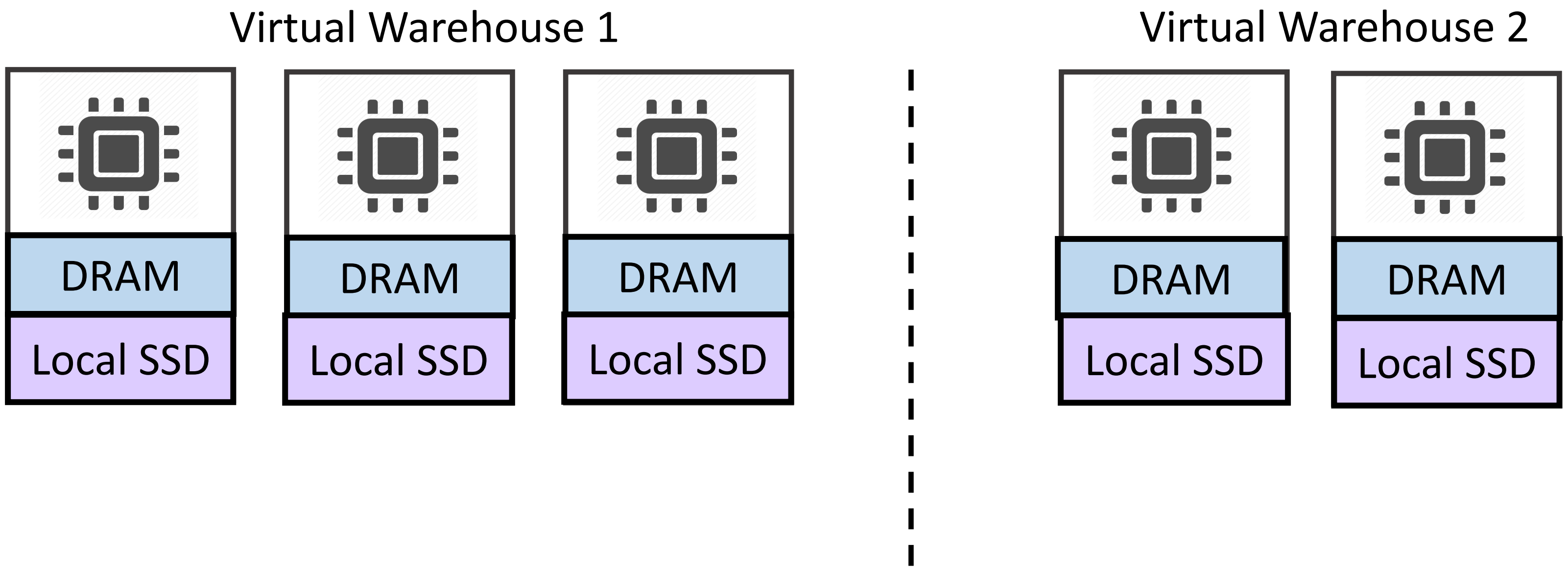
High-level architecture



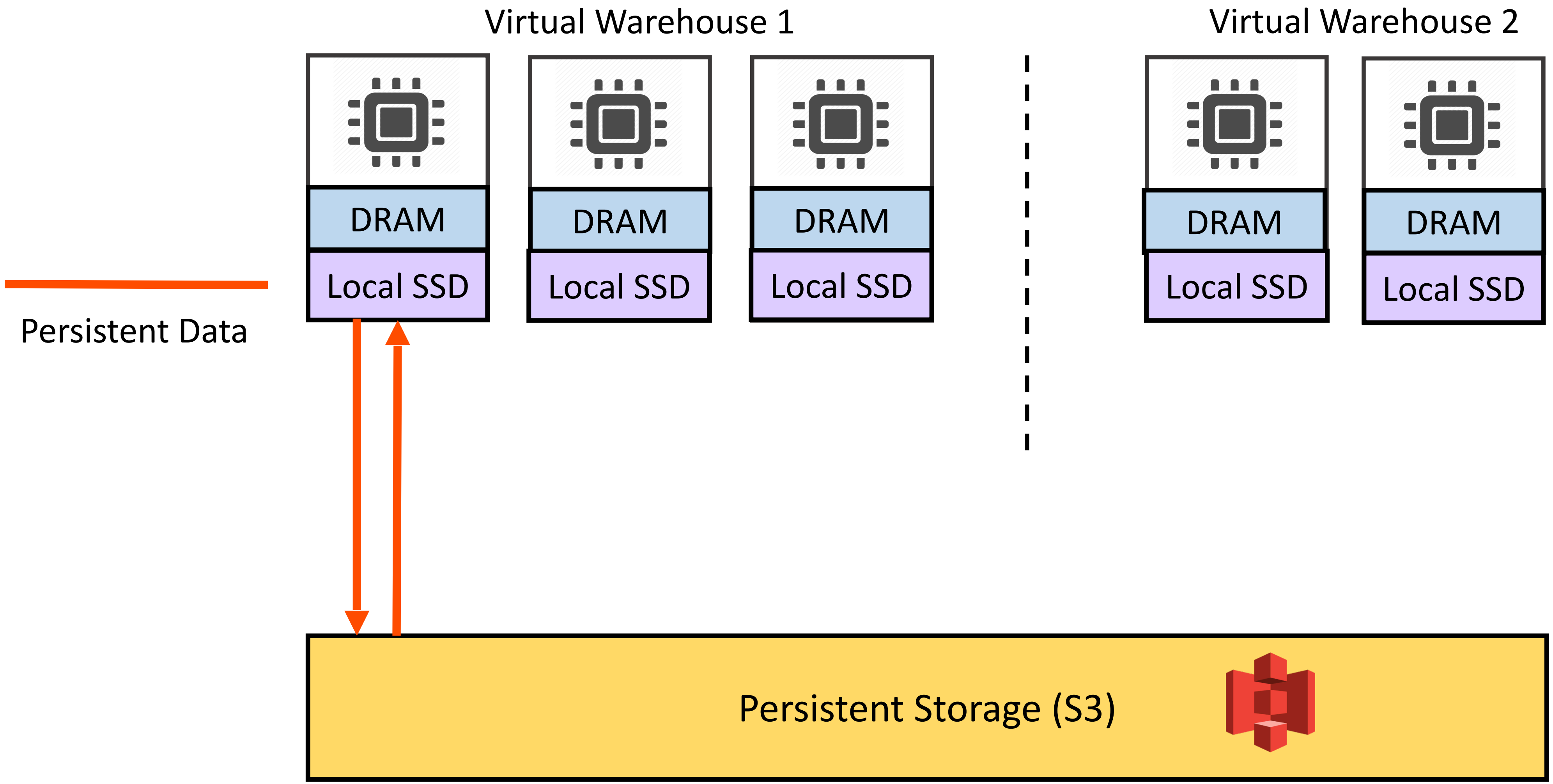
Virtual Warehouse

- Abstraction for computational resources
- Under the hood -> Set of VMs
- Distributed execution of queries

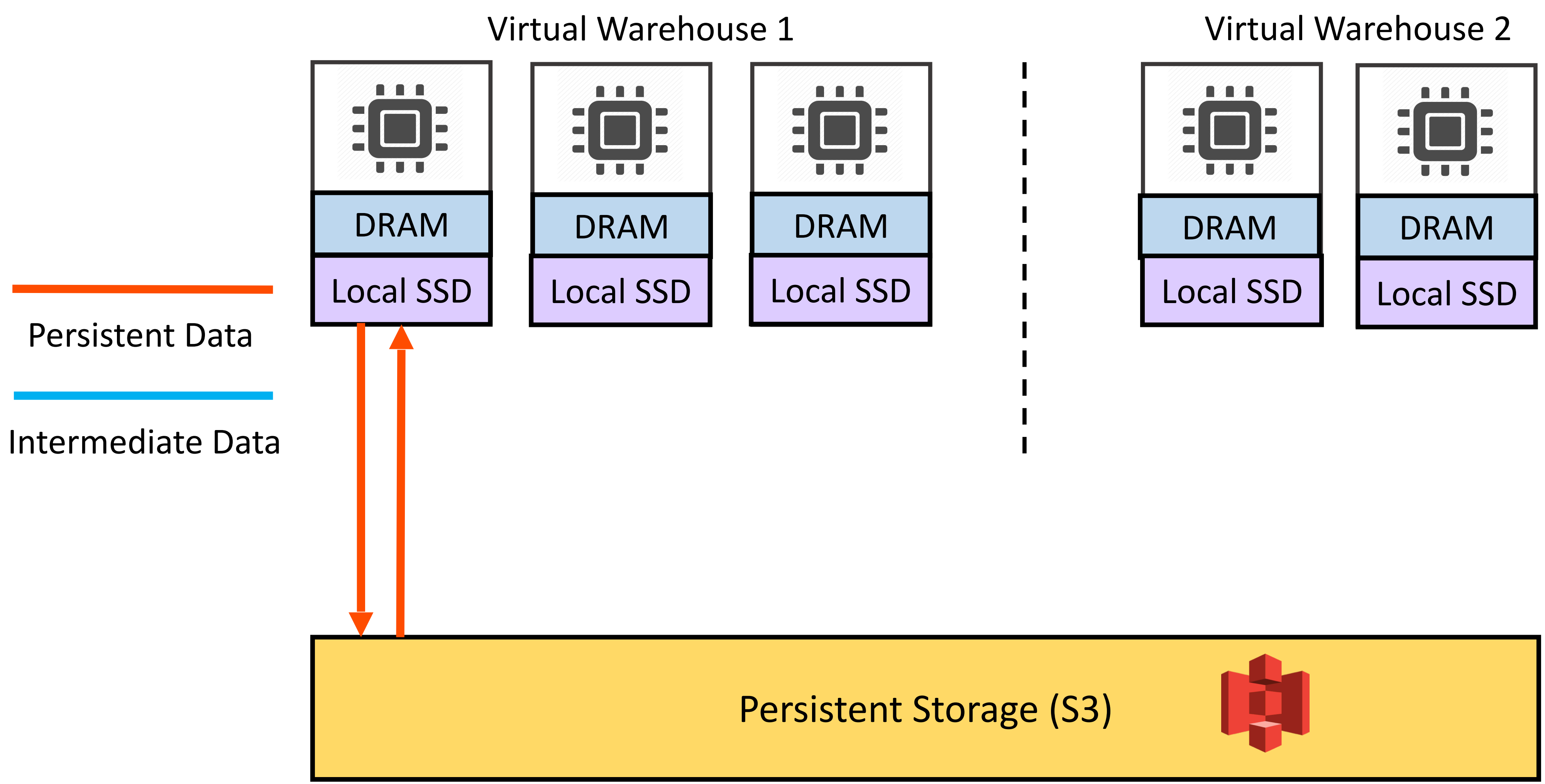
High-level architecture



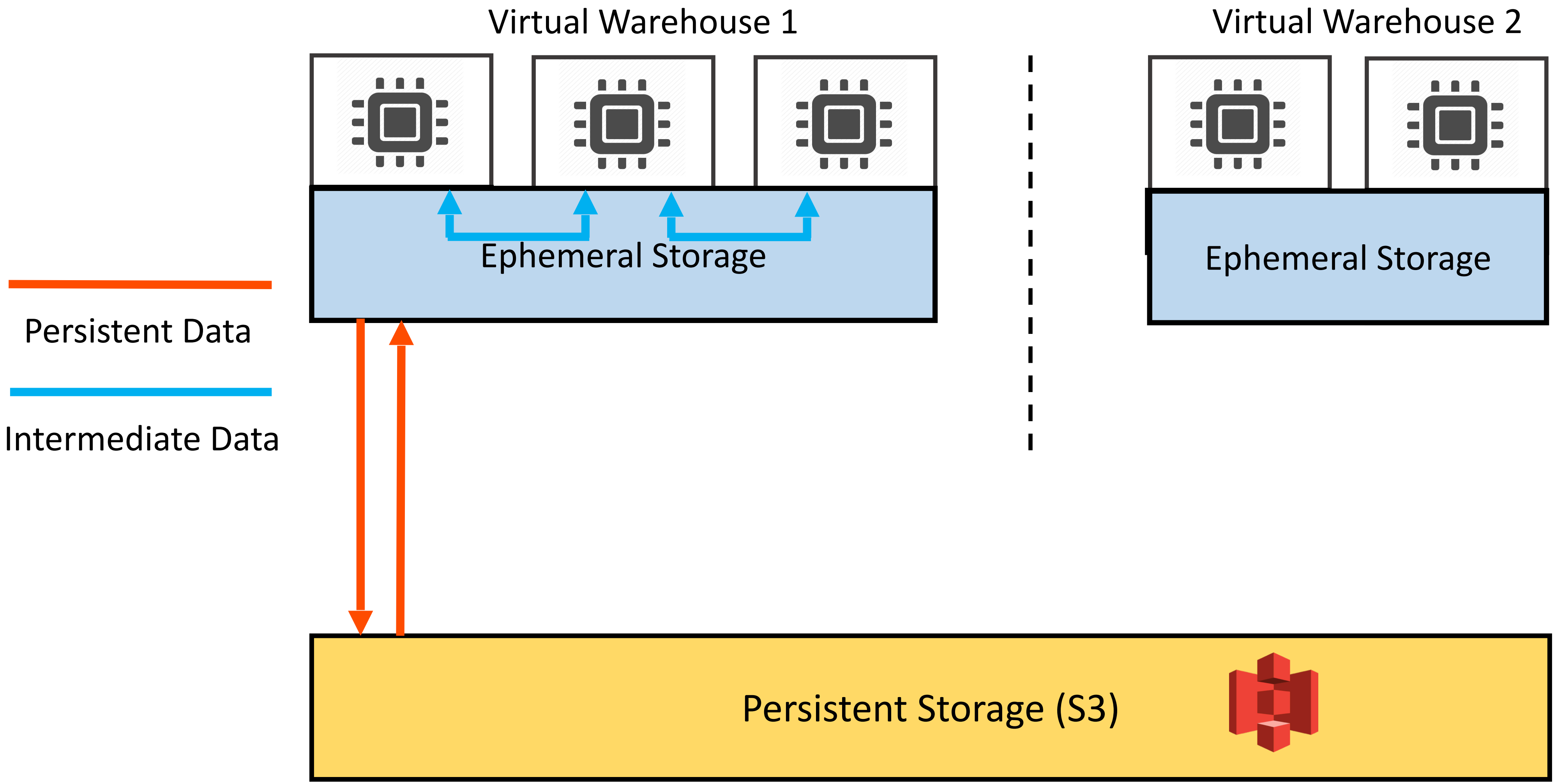
High-level architecture



High-level architecture

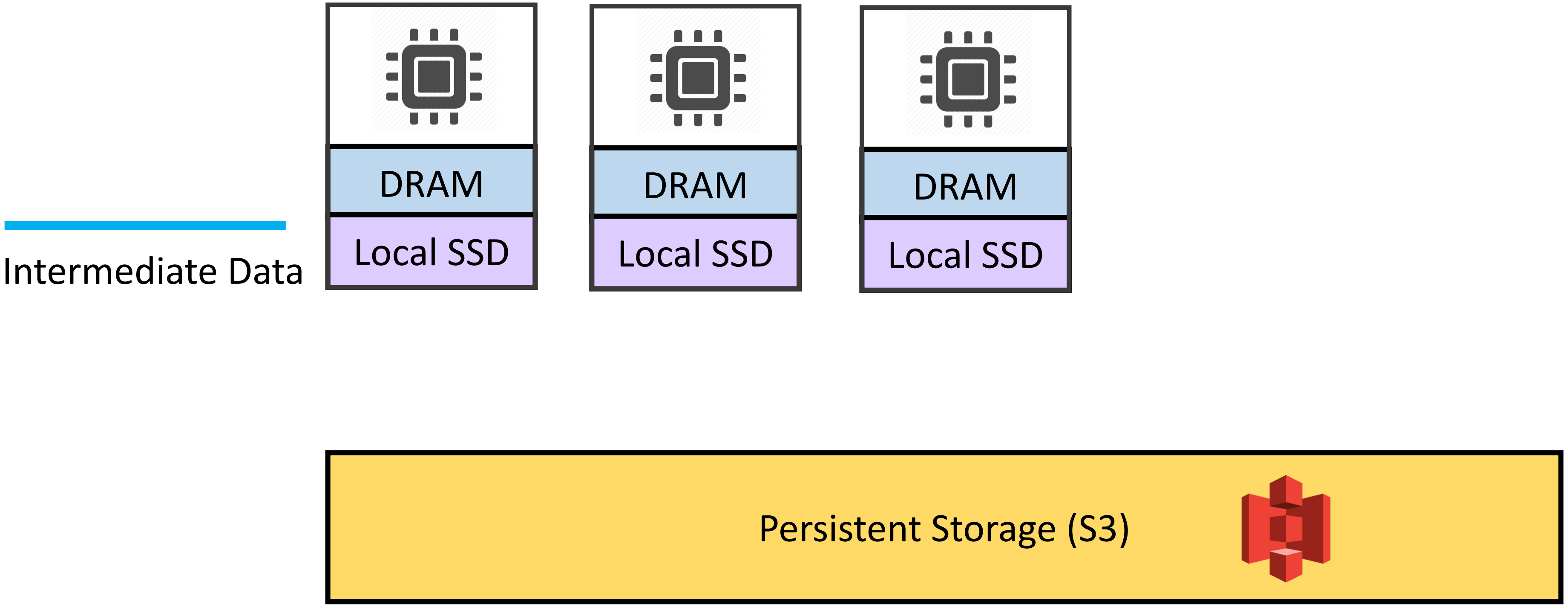


High-level architecture

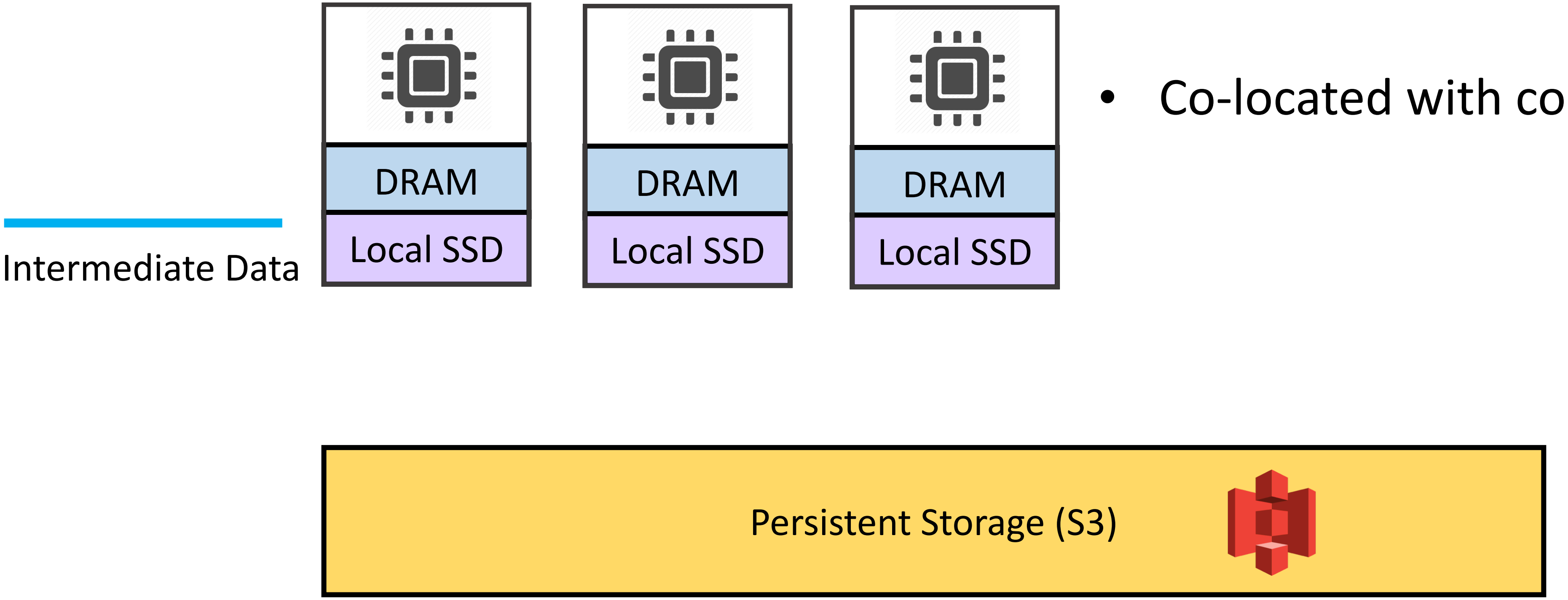


Ephemeral Storage System

Key Features



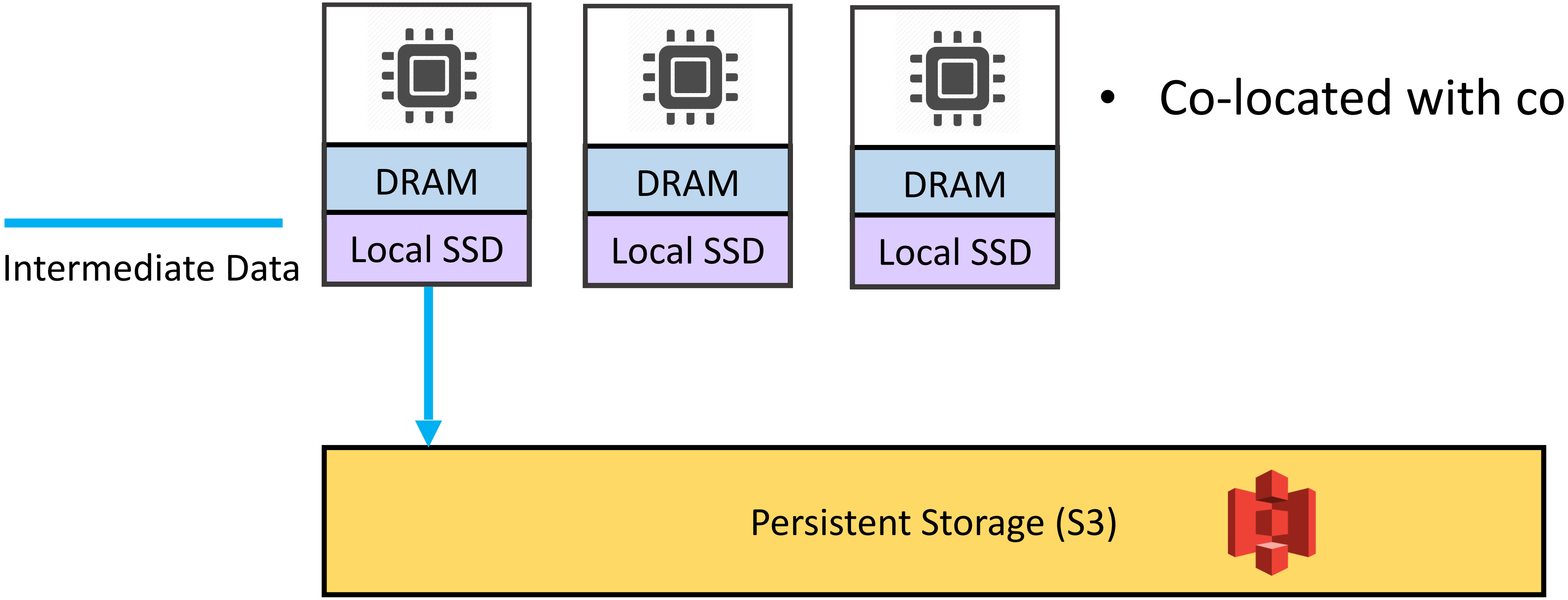
Ephemeral Storage System



Key Features

- Co-located with compute in VWs

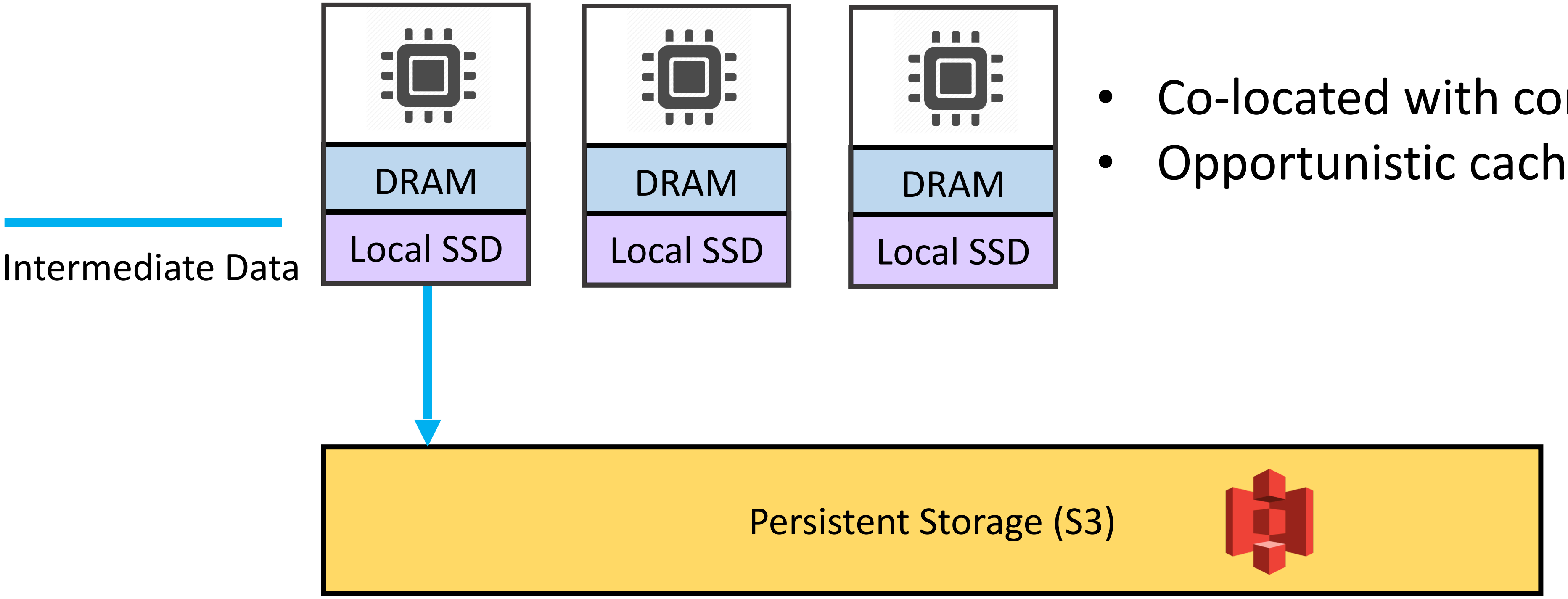
Ephemeral Storage System



Key Features

- Co-located with compute in VWs

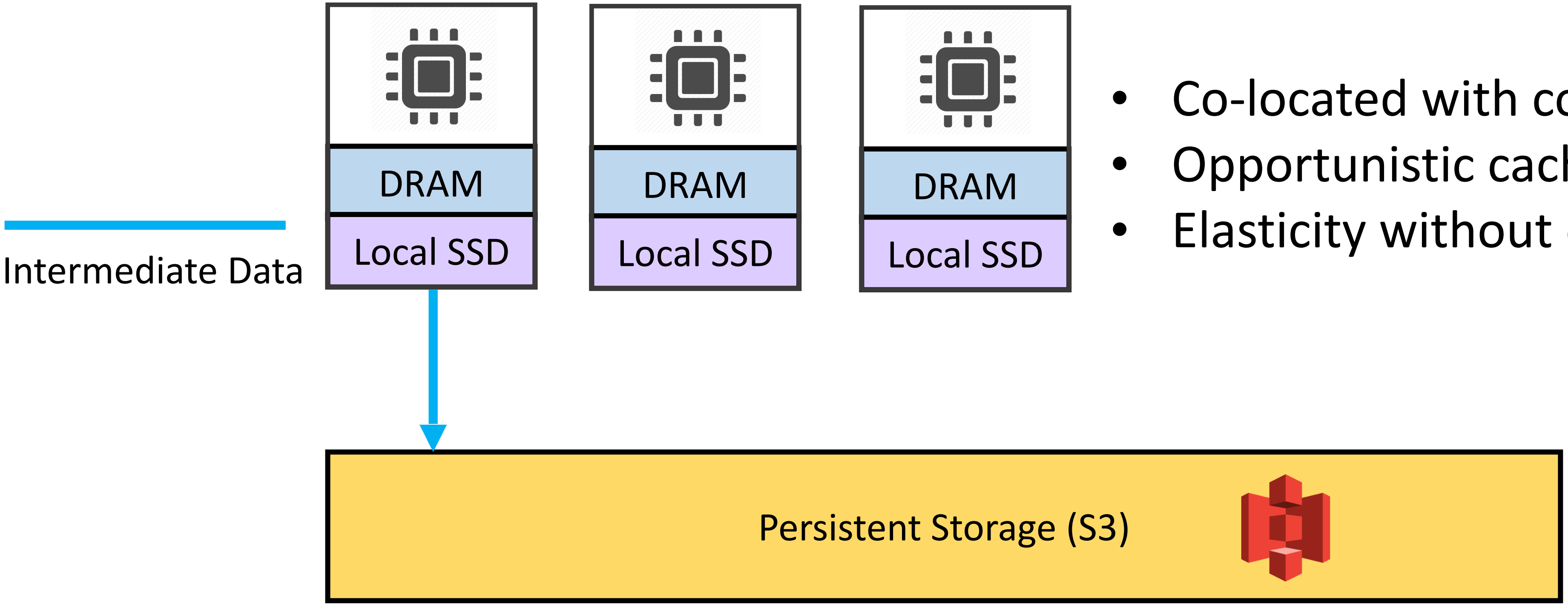
Ephemeral Storage System



Key Features

- Co-located with compute in VWs
- Opportunistic caching of persistent data

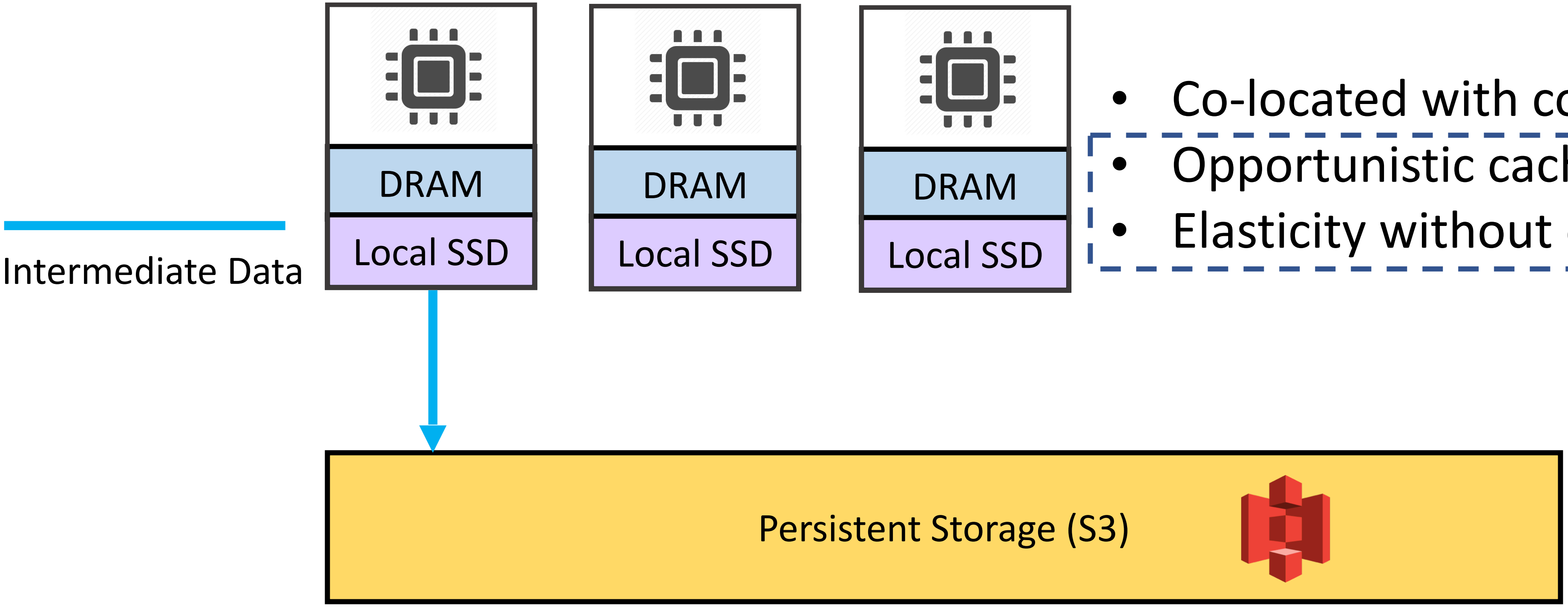
Ephemeral Storage System



Key Features

- Co-located with compute in VWs
- Opportunistic caching of persistent data
- Elasticity without data re-shuffle

Ephemeral Storage System

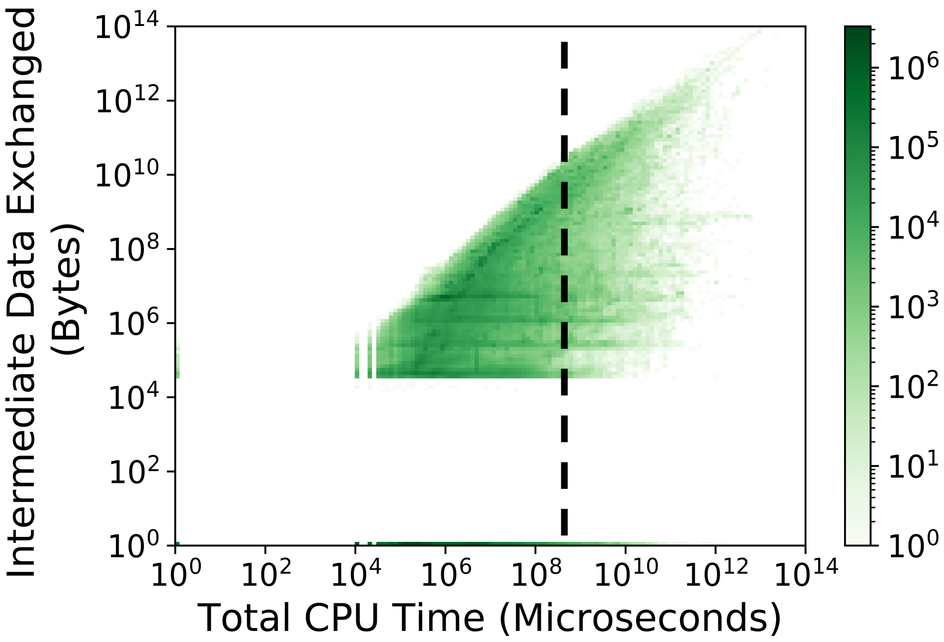


Key Features

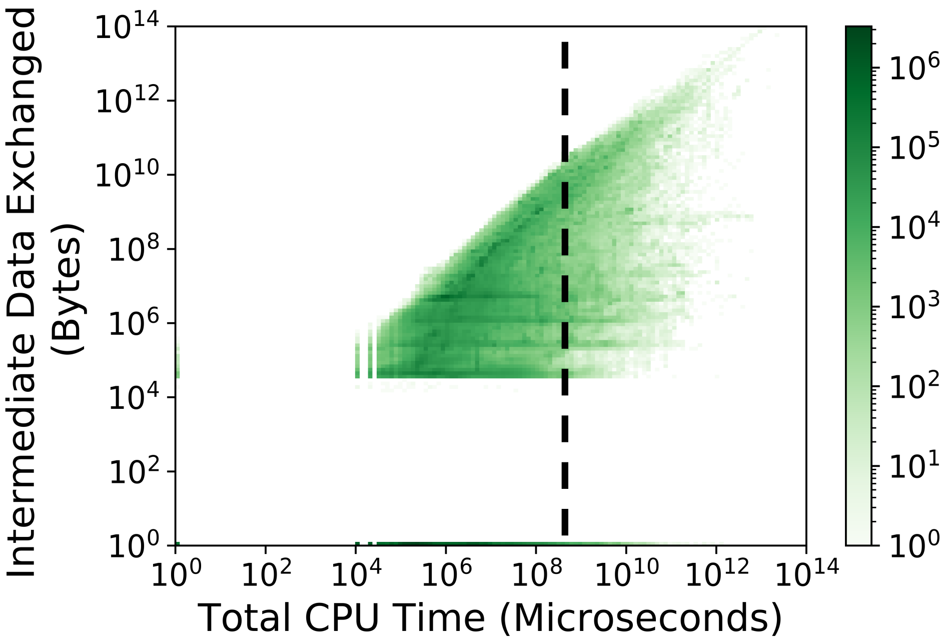
- Co-located with compute in VWs
- Opportunistic caching of persistent data
- Elasticity without data re-shuffle

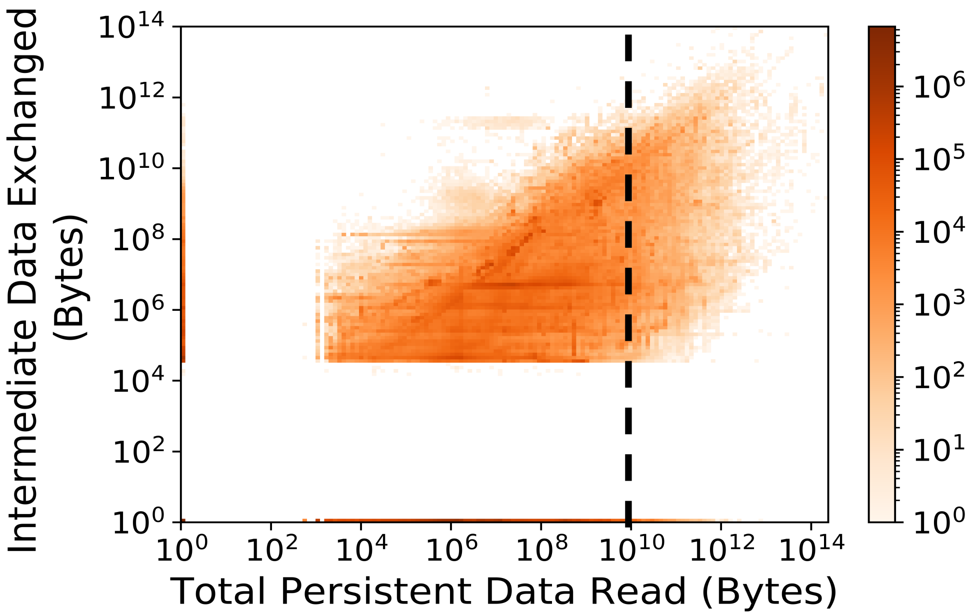
Intermediate Data Characteristics

Intermediate Data Characteristics

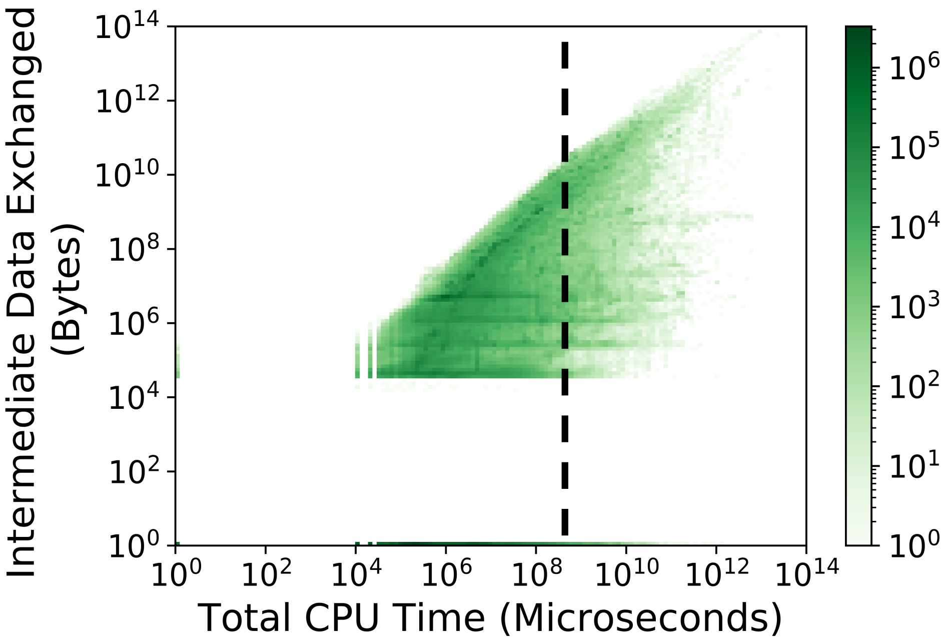


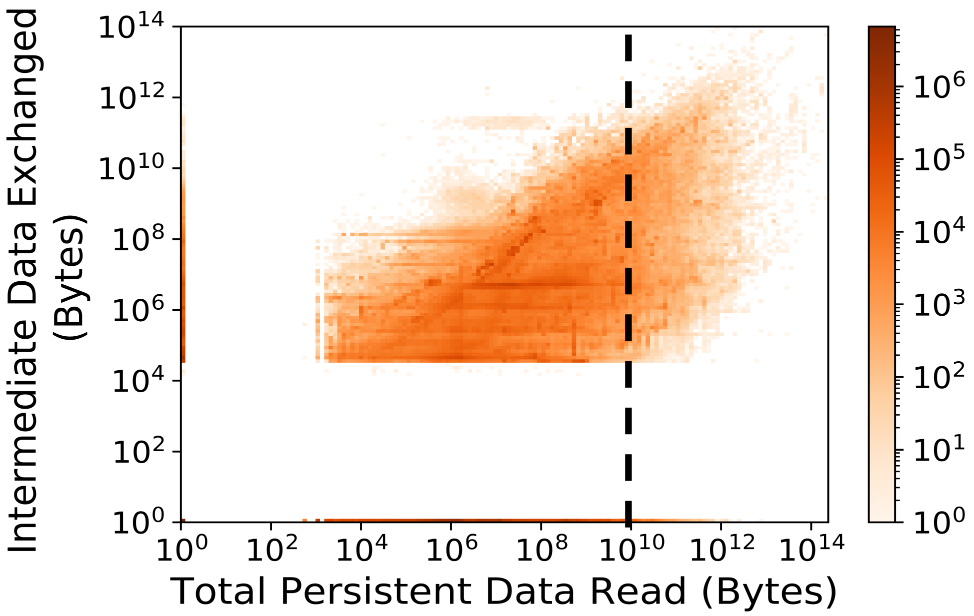
Intermediate Data Characteristics





Intermediate Data Characteristics





Intermediate data sizes -> variation over 5 orders of magnitude

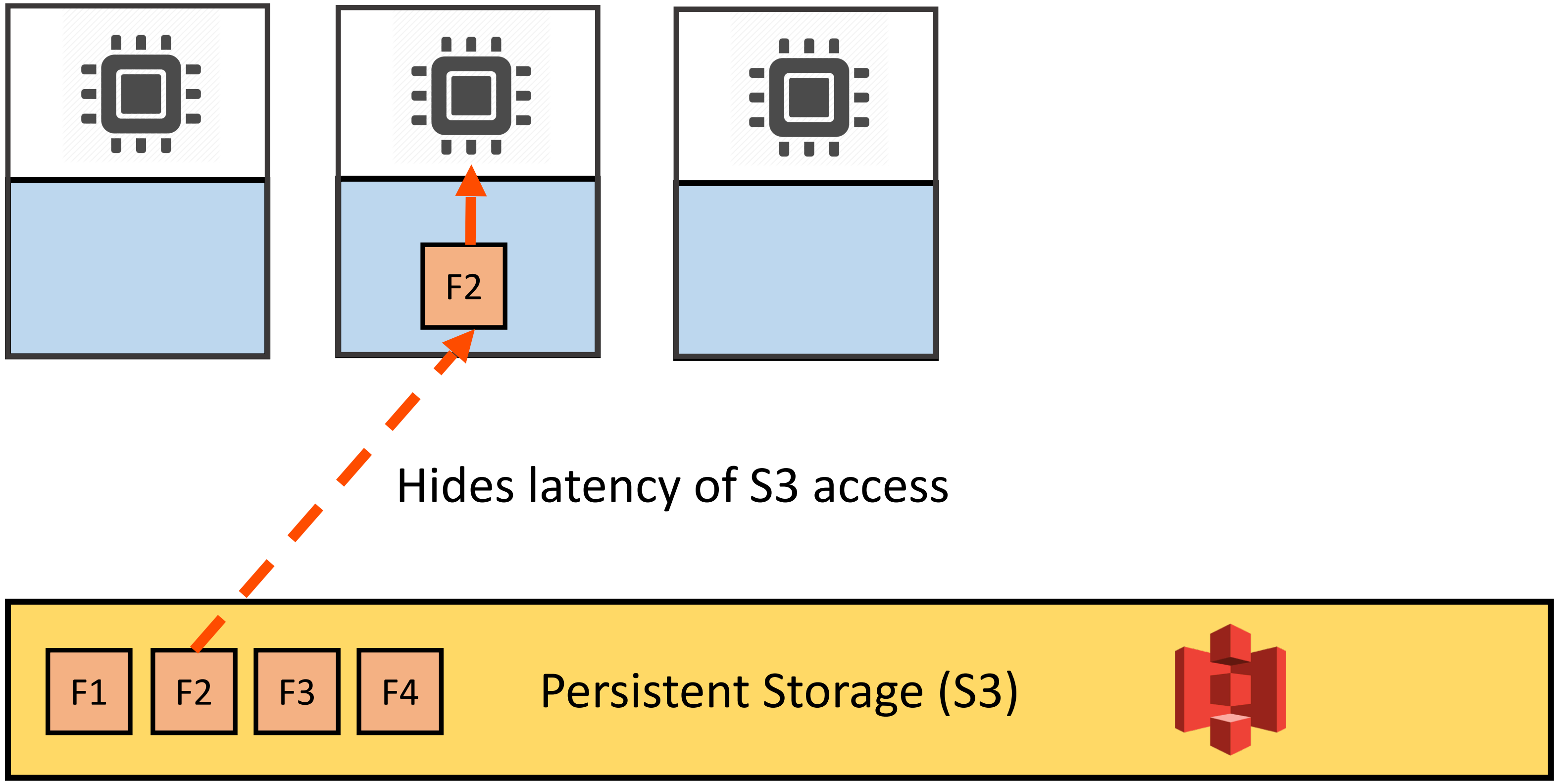
Difficult to predict intermediate data sizes upfront

Decouple compute & ephemeral storage?

Persistent Data Caching

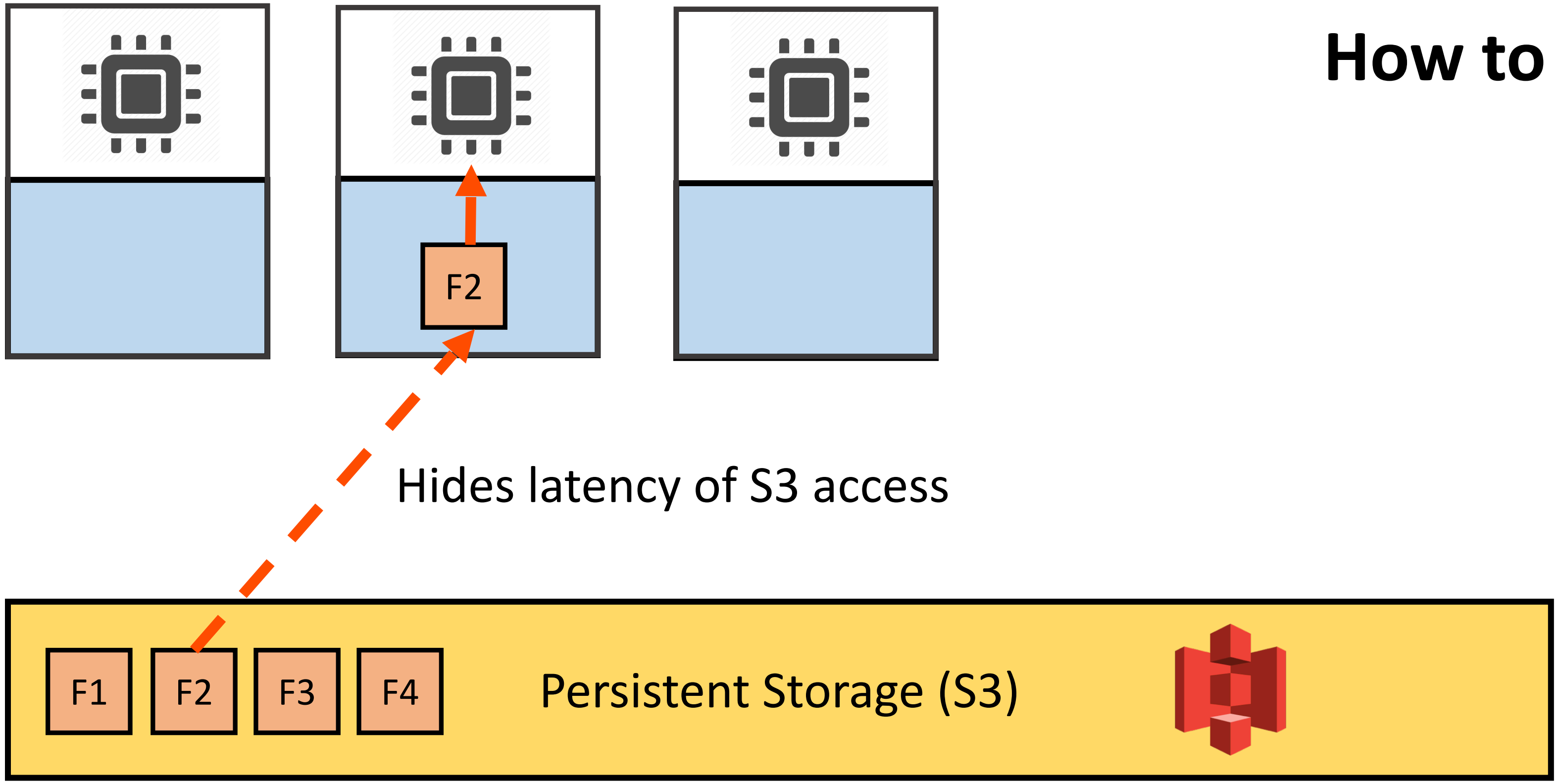
Persistent Data Caching

- Intermediate data volume -> Peak ↑ Average ↓
- Opportunistic caching of persistent data in ephemeral storage system



Persistent Data Caching

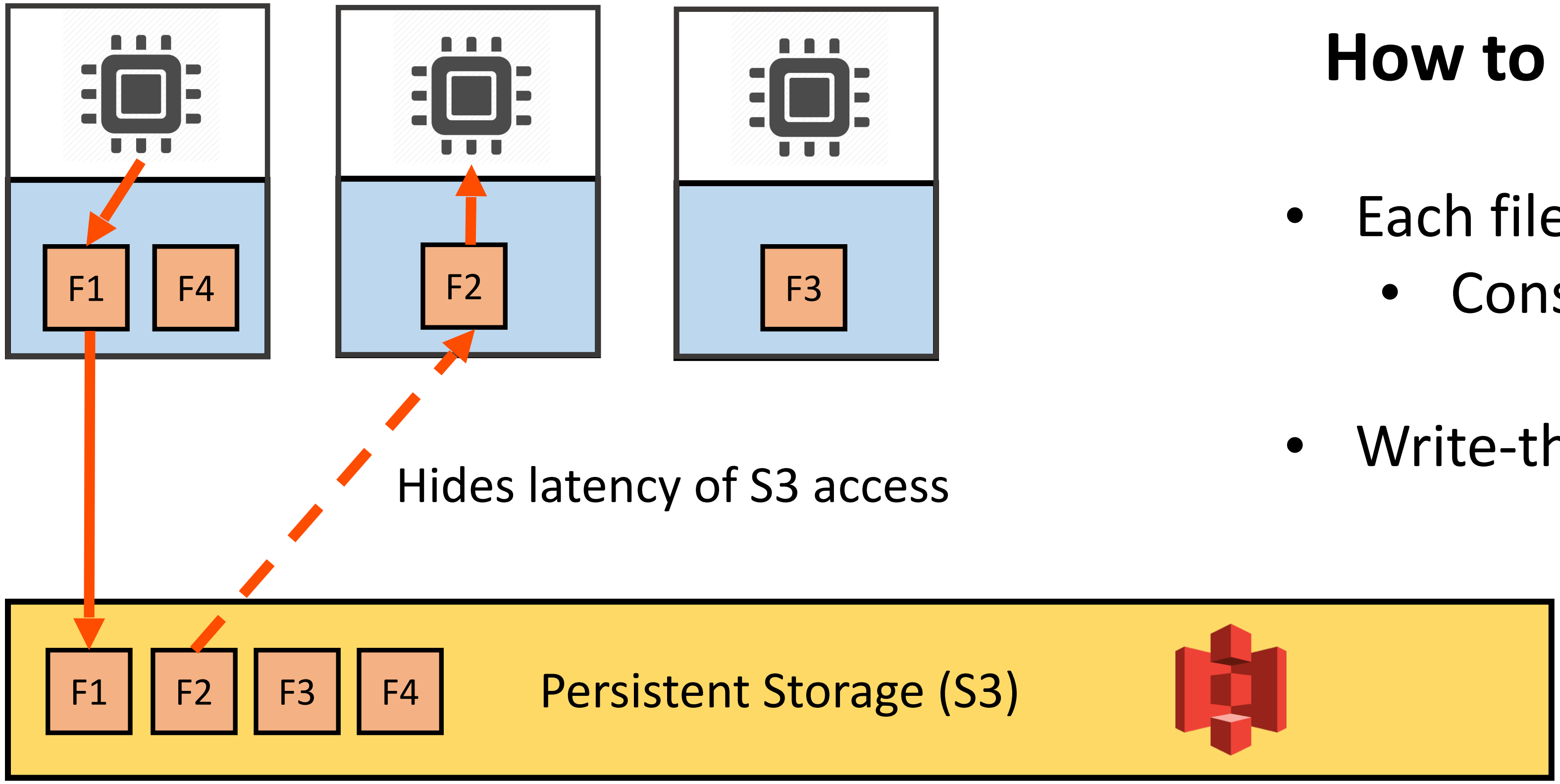
- Intermediate data volume -> Peak Average
- Opportunistic caching of persistent data in ephemeral storage system



How to ensure consistency?

Persistent Data Caching

- Intermediate data volume -> Peak ↑ Average ↓
- Opportunistic caching of persistent data in ephemeral storage system

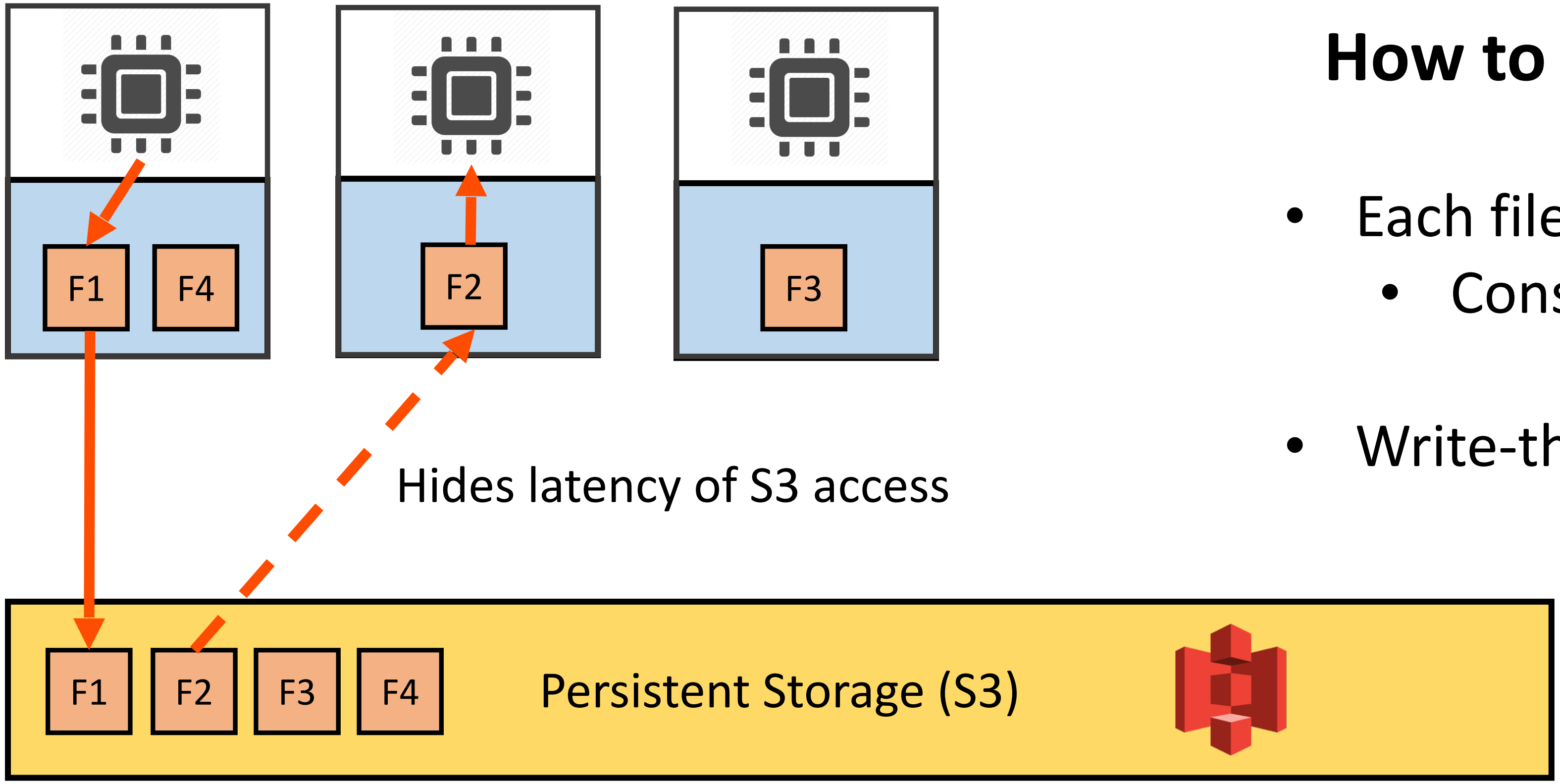


How to ensure consistency?

- Each file assigned to unique node
  - Consistent hashing
- Write-through caching

Persistent Data Caching

- Intermediate data volume -> Peak ↑ Average ↓
- Opportunistic caching of persistent data in ephemeral storage system



How to ensure consistency?

- Each file assigned to unique node
  - Consistent hashing
- Write-through caching

Analysis, Future Directions in paper

Elasticity

Elasticity

- **Persistent storage** – easy, offloaded to S3

Elasticity

- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, **pre-warmed pool of VMs**

Elasticity

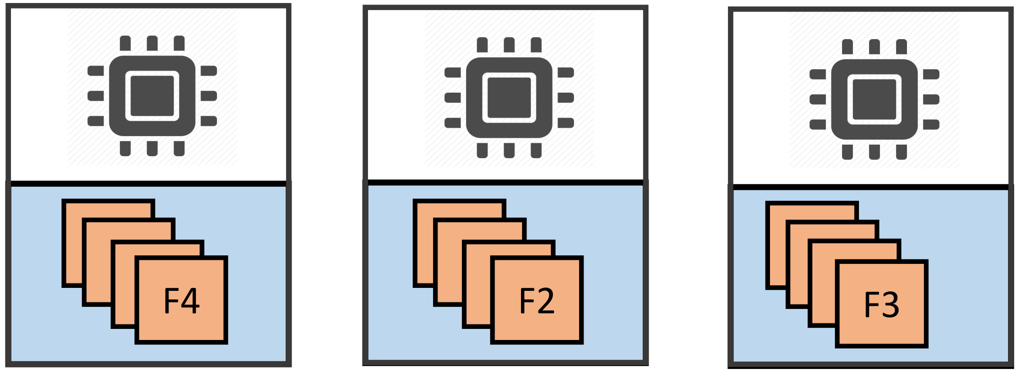
- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, **pre-warmed pool of VMs**
- **Ephemeral storage** – challenging, due to co-location with compute

  Back to shared-nothing architecture problem (data re-shuffle)

Elasticity

- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, **pre-warmed pool of VMs**
- **Ephemeral storage** – challenging, due to co-location with compute

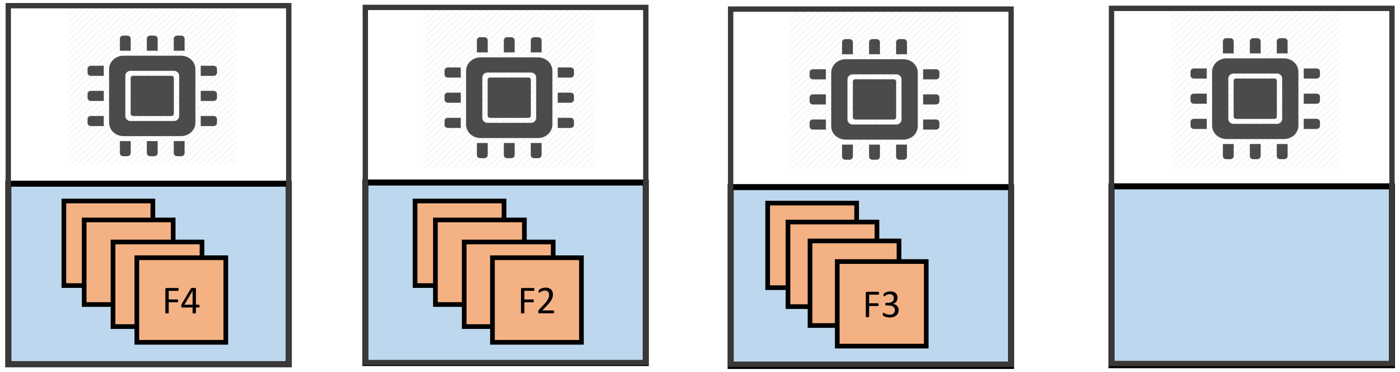
  Back to shared-nothing architecture problem (data re-shuffle)



Elasticity

- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, **pre-warmed pool of VMs**
- **Ephemeral storage** – challenging, due to co-location with compute

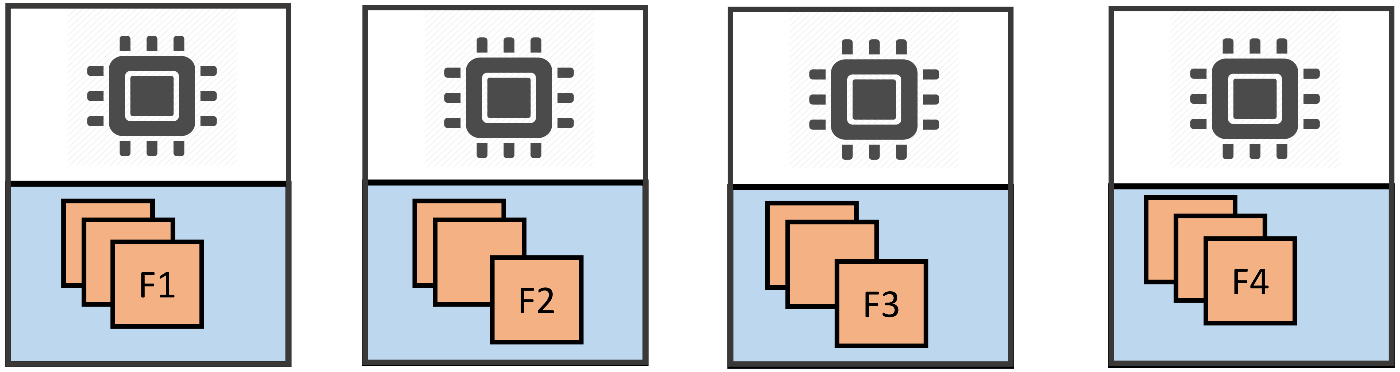
  Back to shared-nothing architecture problem (data re-shuffle)



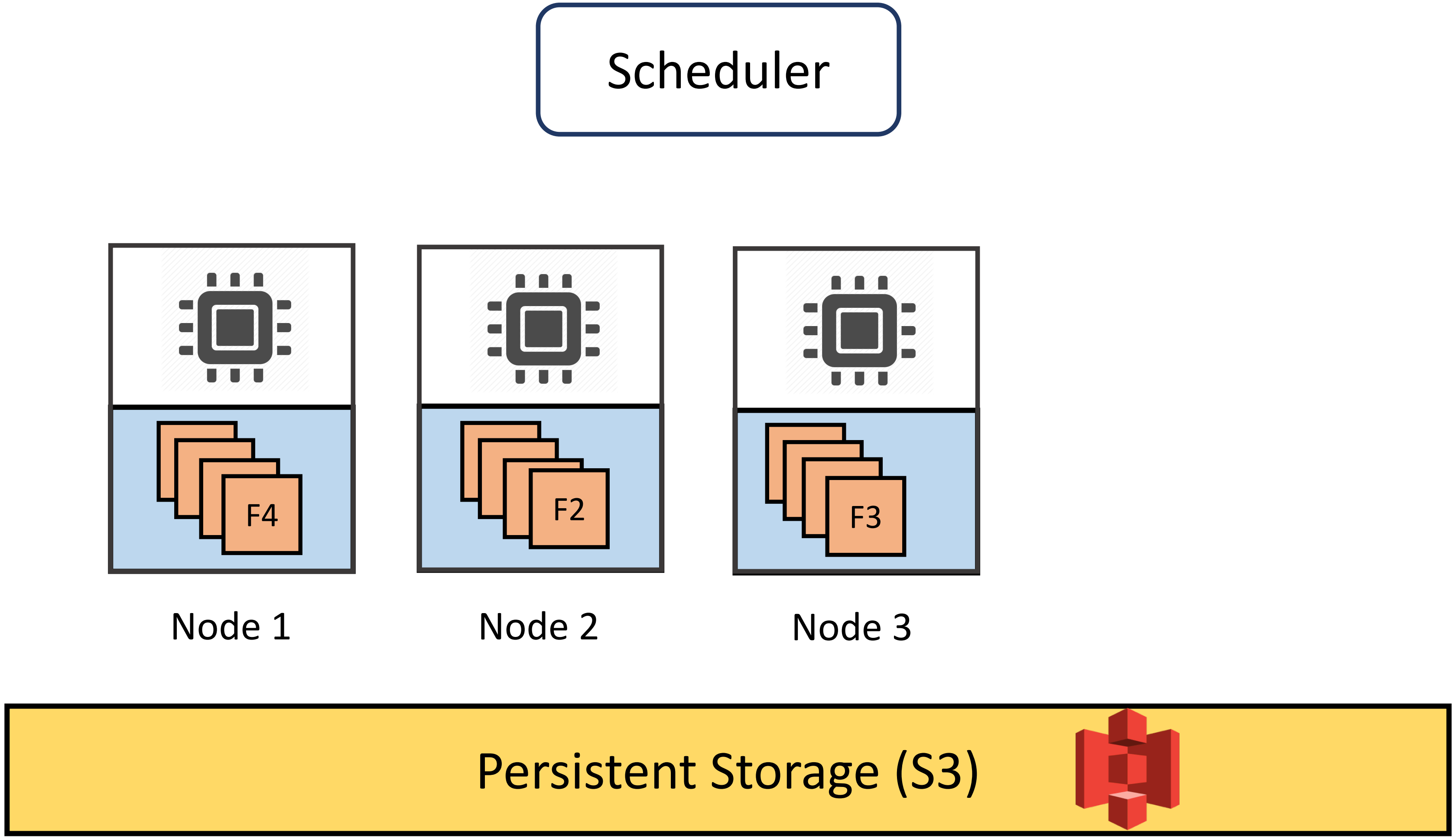
Elasticity

- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, **pre-warmed pool of VMs**
- **Ephemeral storage** – challenging, due to co-location with compute

  Back to shared-nothing architecture problem (data re-shuffle)

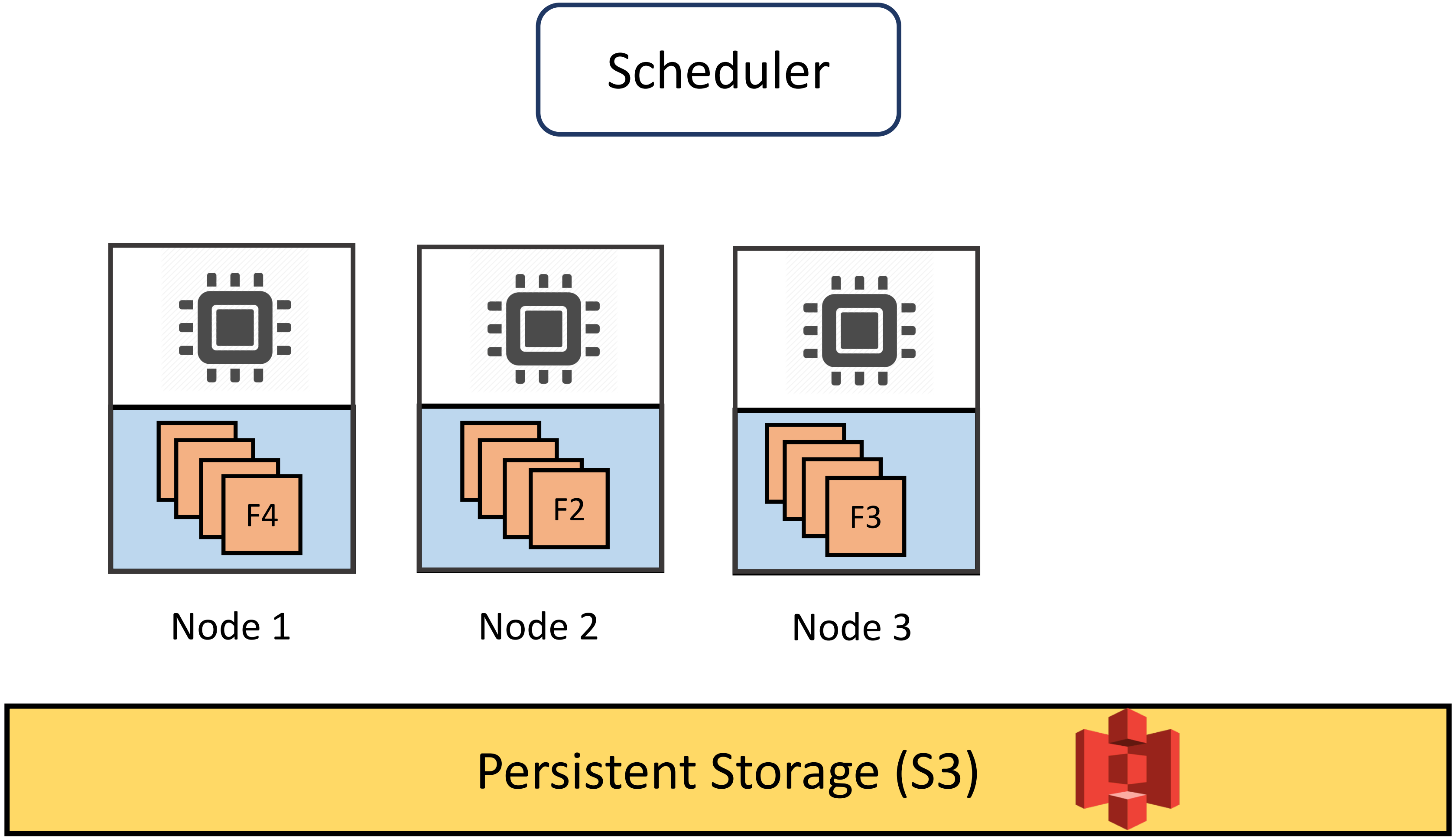


Lazy Consistent Hashing



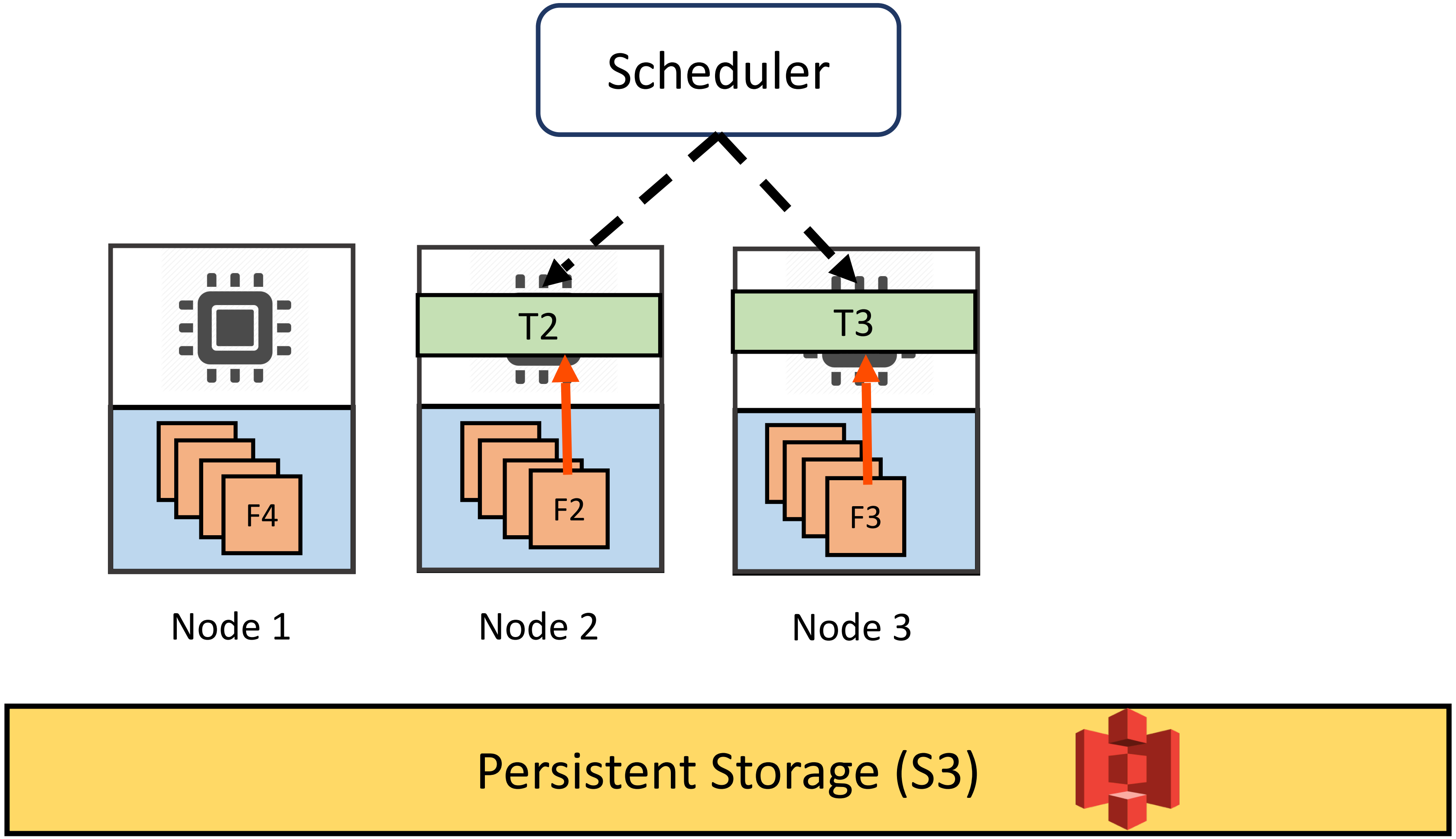
Lazy Consistent Hashing

Locality aware task scheduling

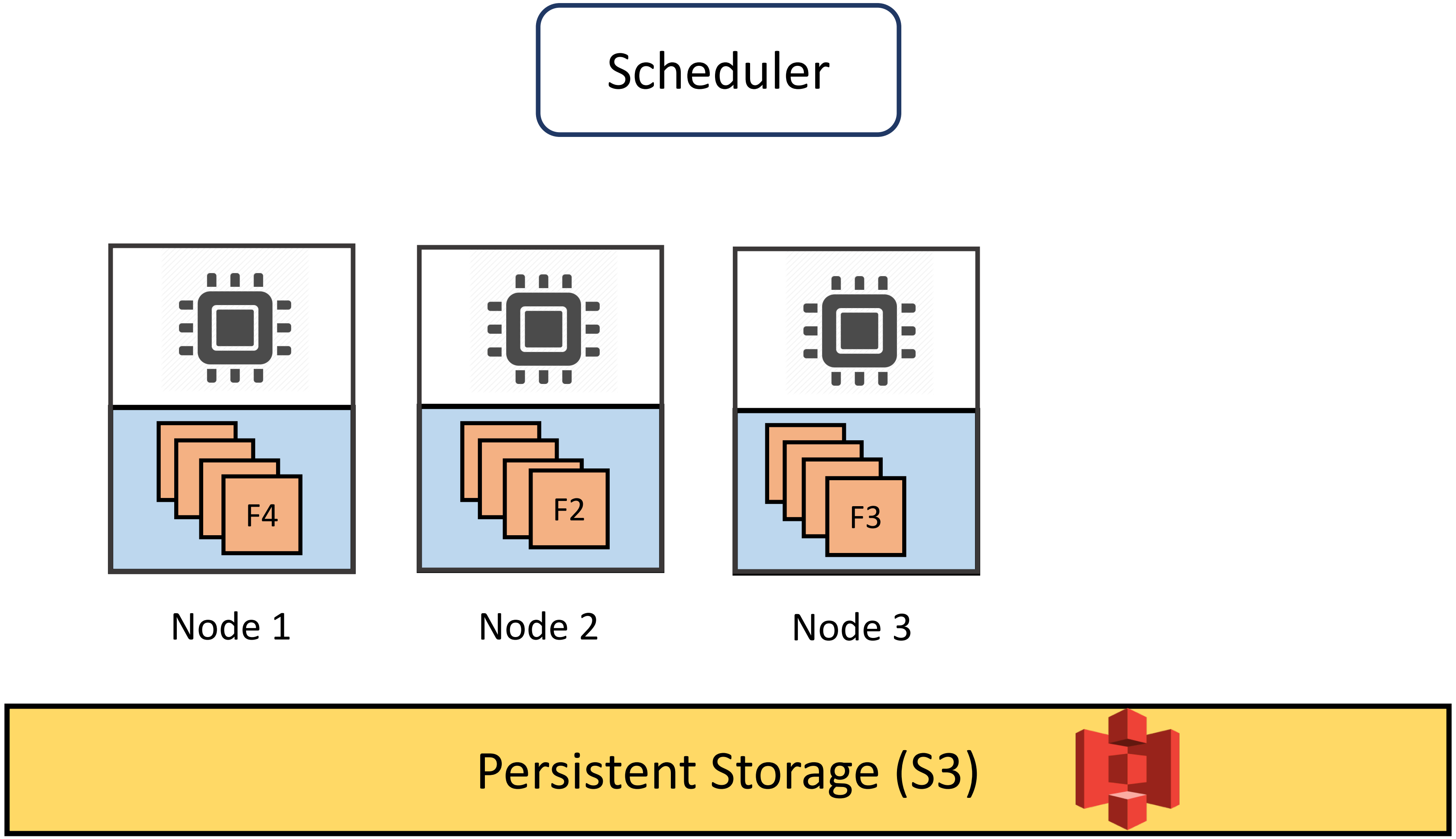


Lazy Consistent Hashing

Locality aware task scheduling

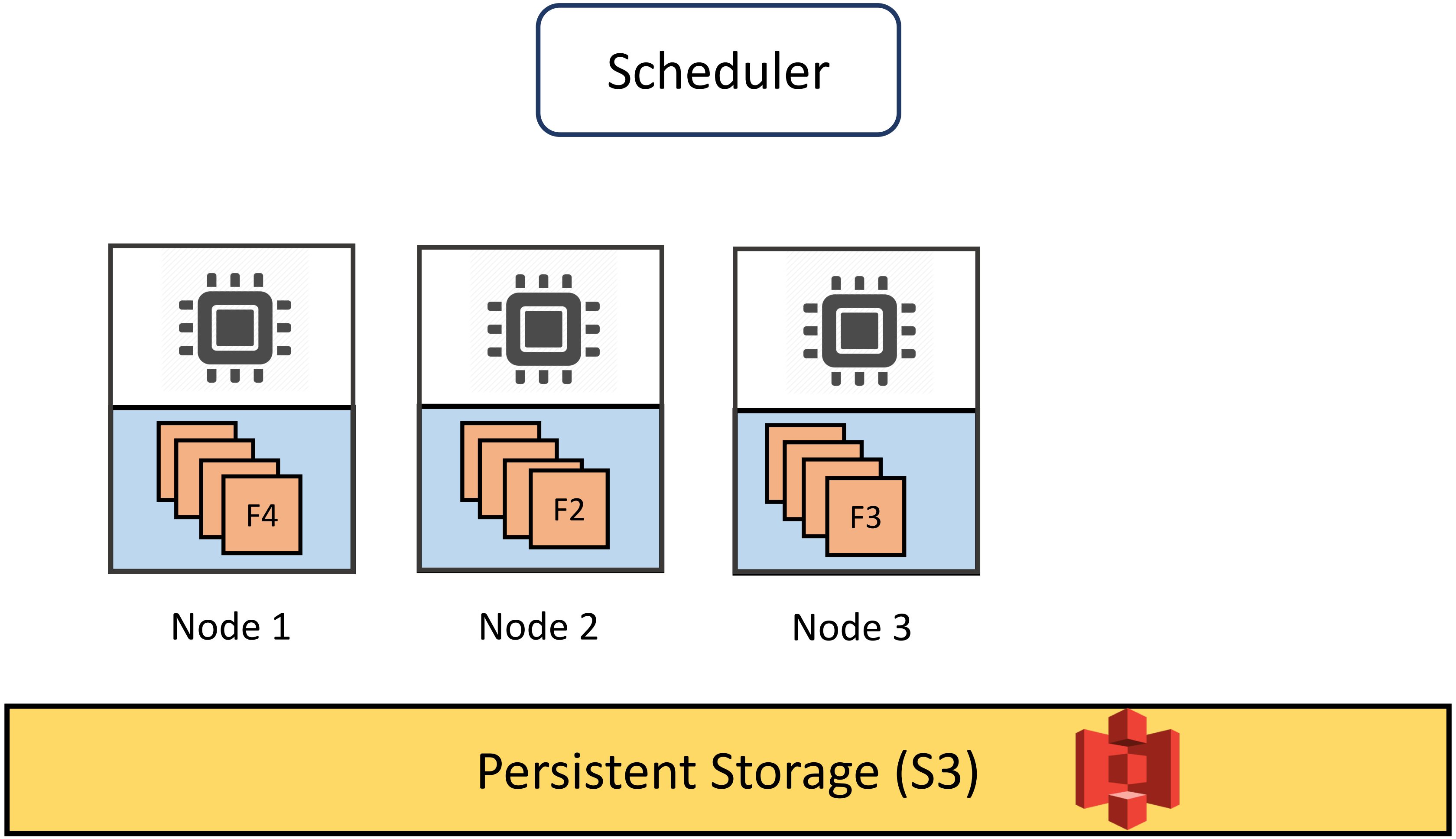


Lazy Consistent Hashing



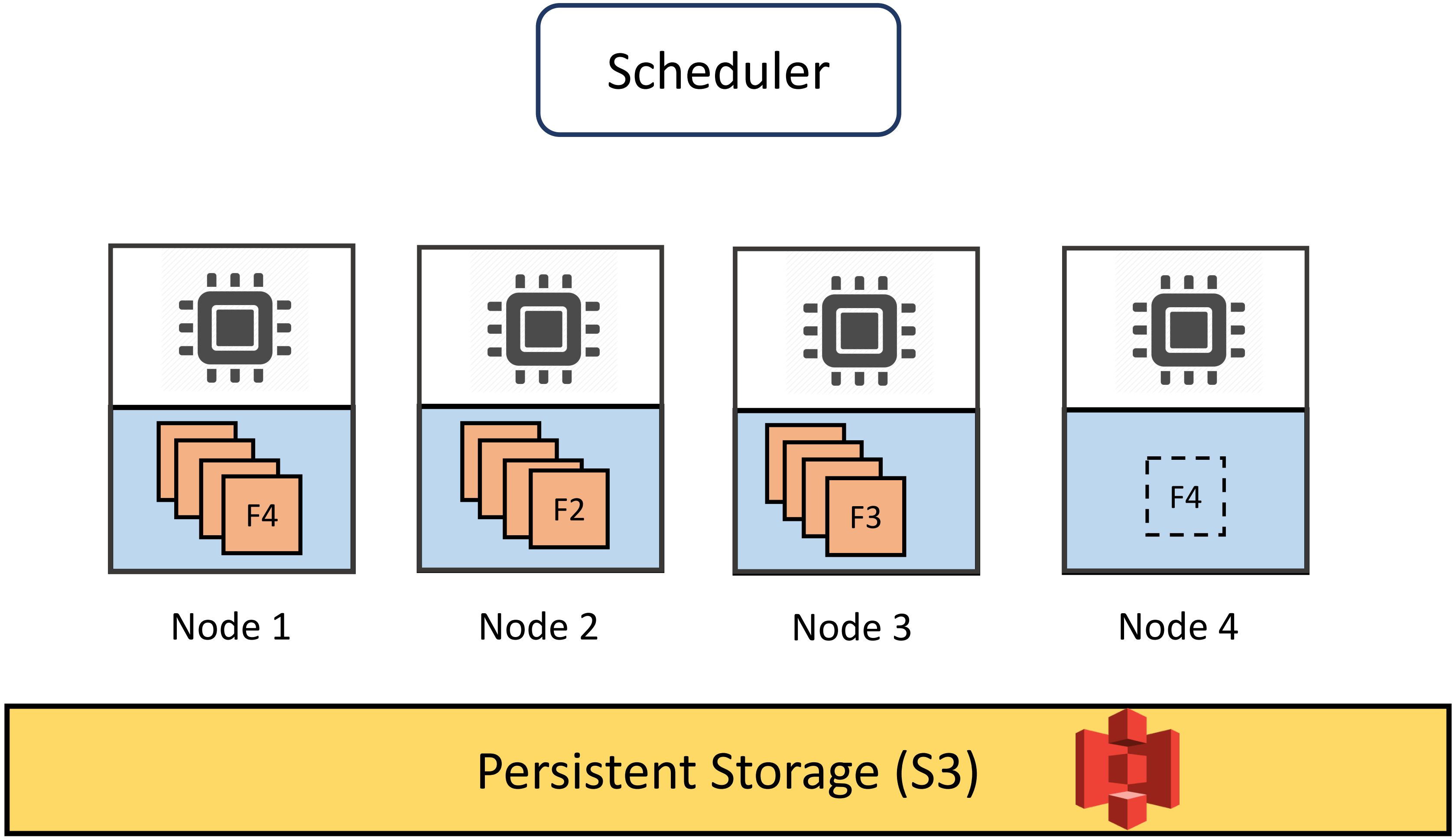
Lazy Consistent Hashing

Elasticity without data re-shuffle



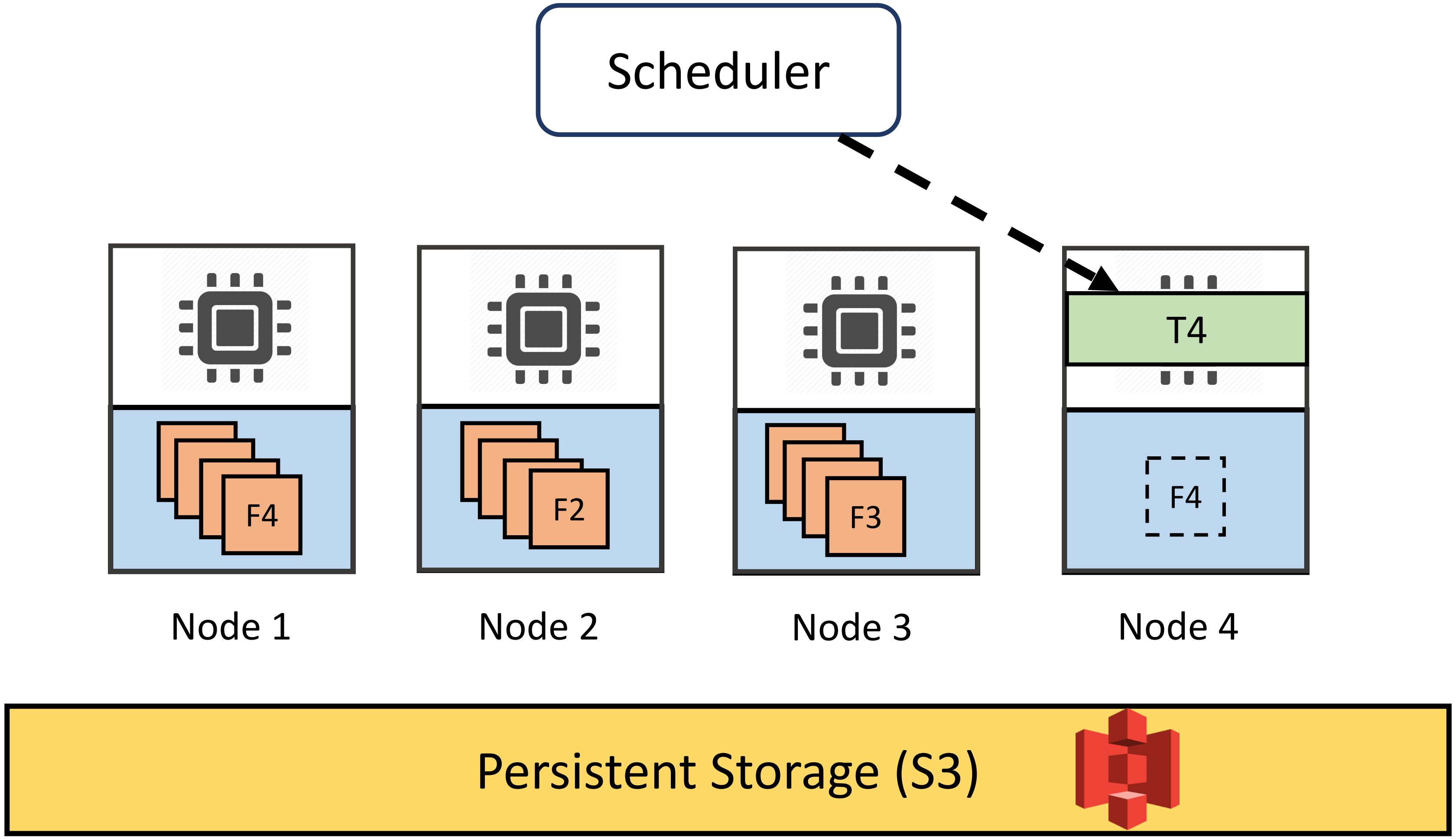
Lazy Consistent Hashing

Elasticity without data re-shuffle



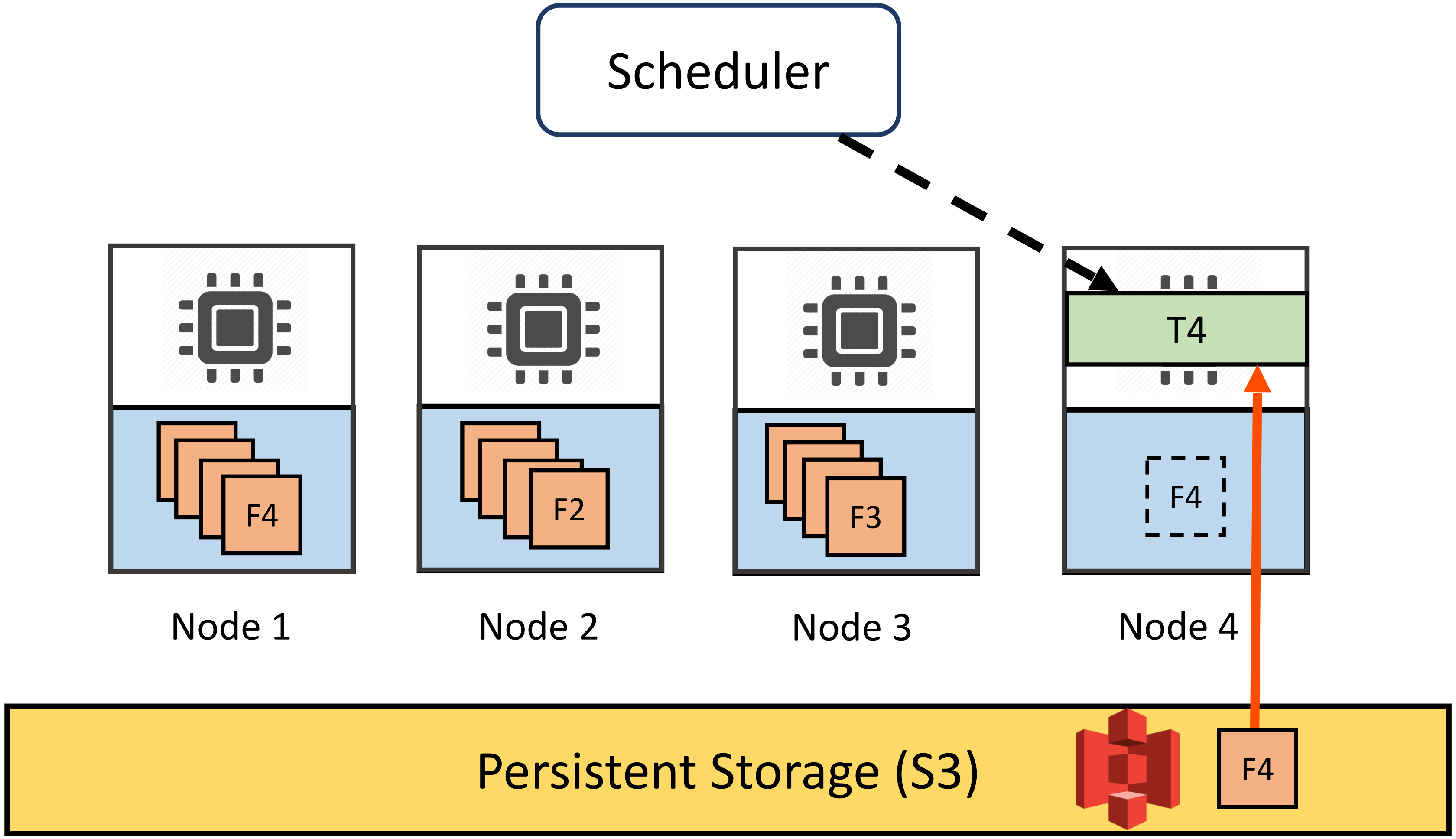
Lazy Consistent Hashing

Elasticity without data re-shuffle



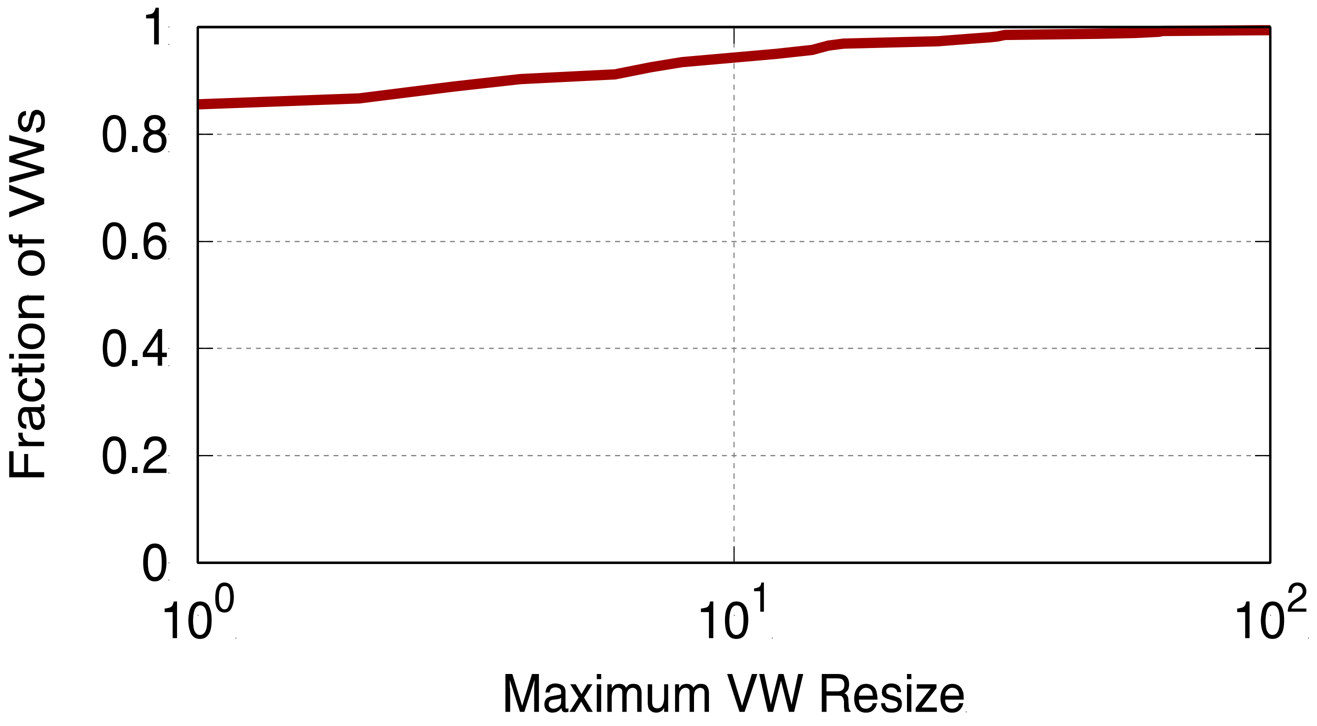
Lazy Consistent Hashing

Elasticity without data re-shuffle

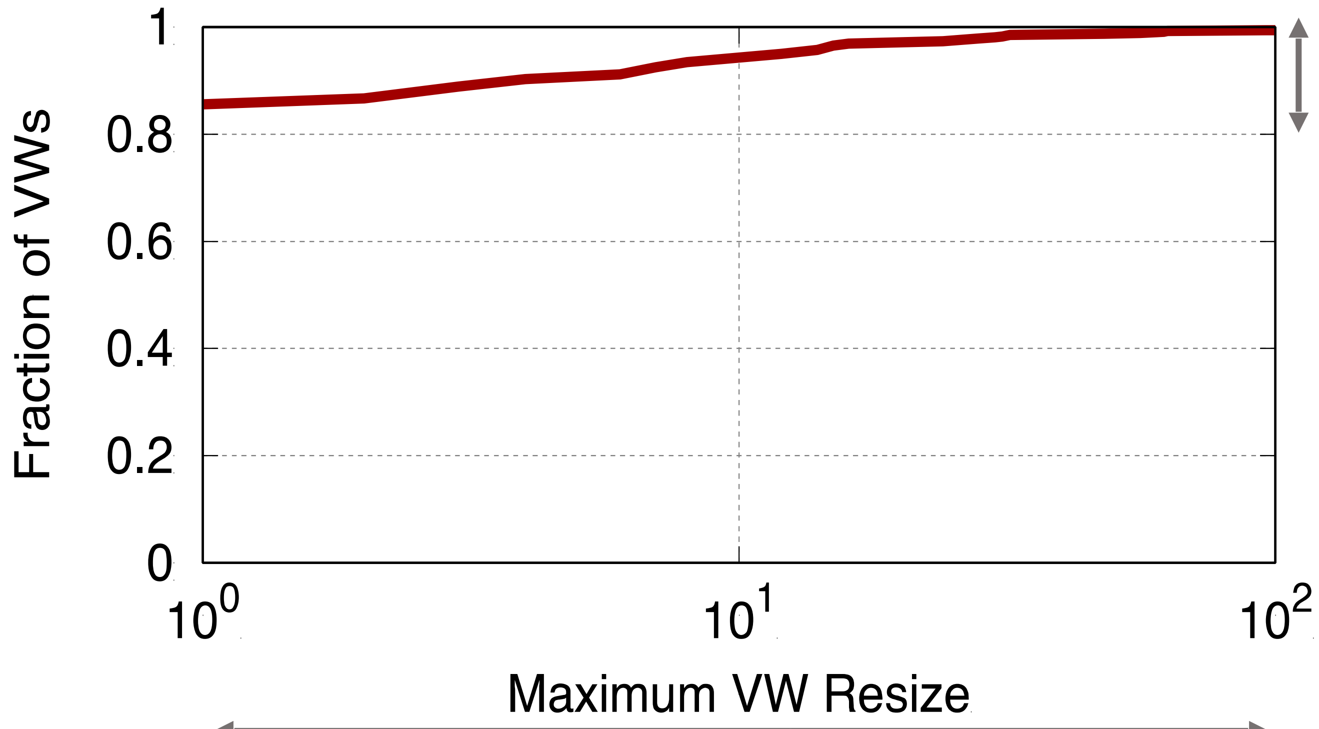


Do customers exploit elasticity in the wild?

Do customers exploit elasticity in the wild?



Do customers exploit elasticity in the wild?



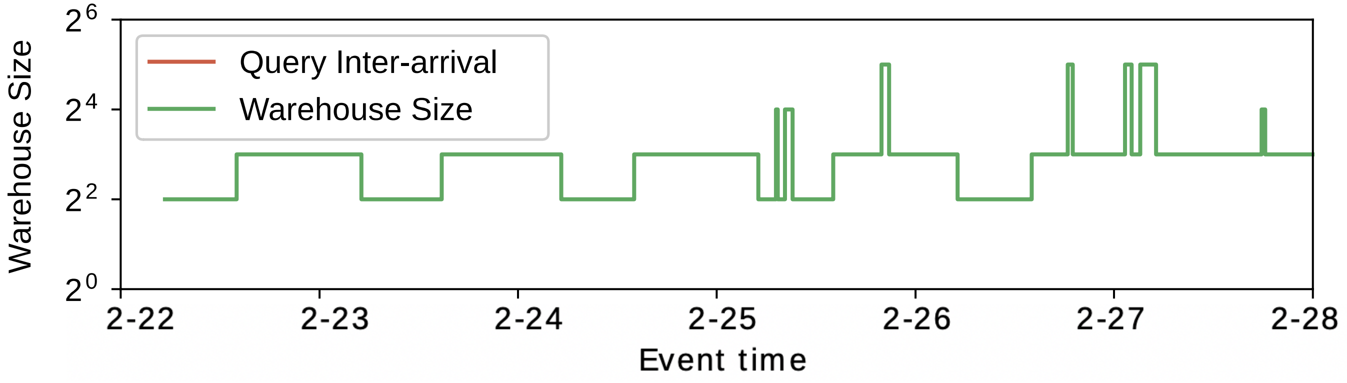
20% warehouses exploit elasticity

Sometimes by up to 100x

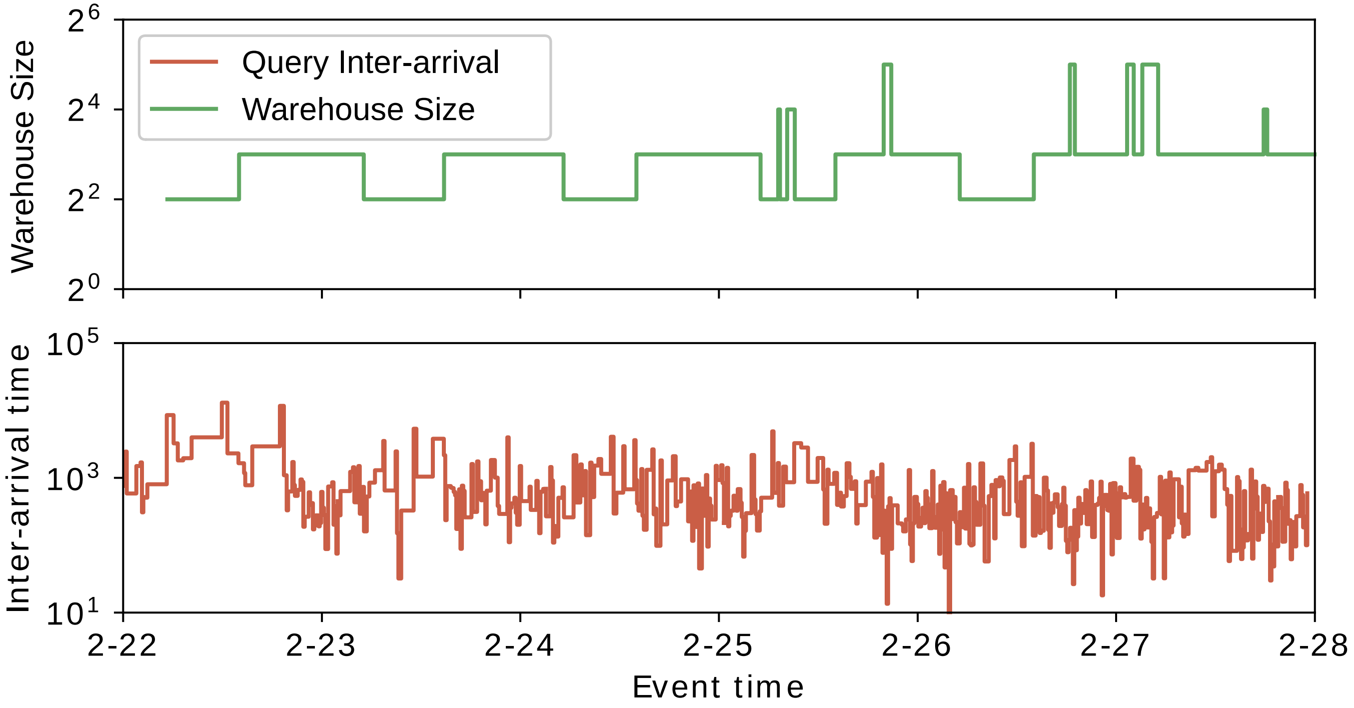
Resource scaling by up to 100x needed at times

At what time-scales are warehouses resized?

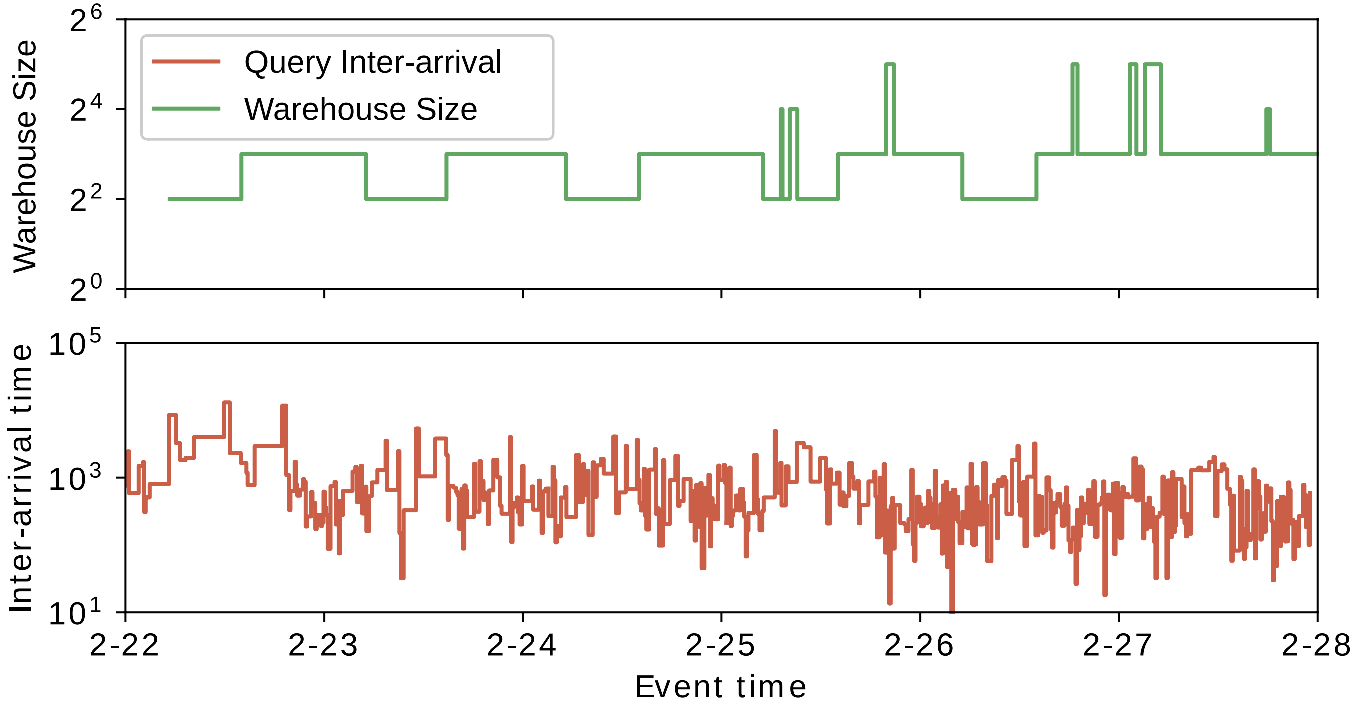
At what time-scales are warehouses resized?



At what time-scales are warehouses resized?



At what time-scales are warehouses resized?

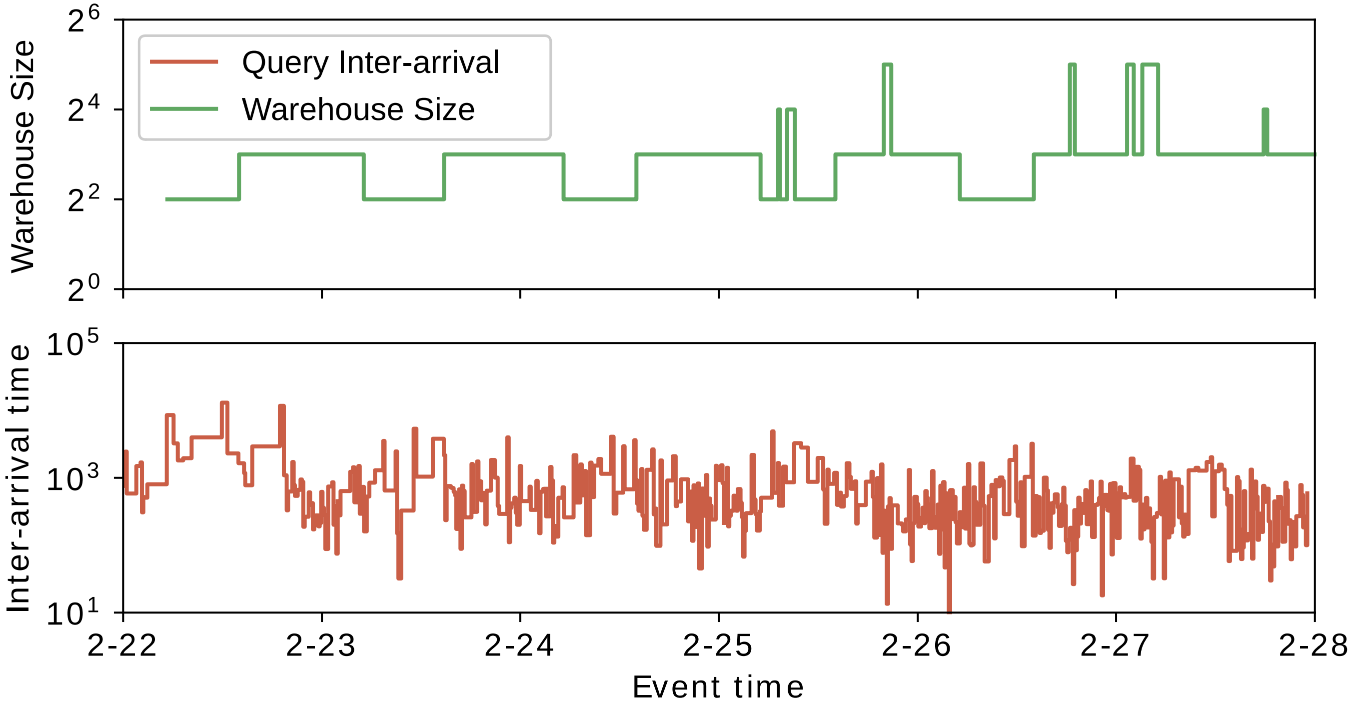


Granularity of warehouse elasticity

>>

Changes in query load

At what time-scales are warehouses resized?



Granularity of warehouse elasticity

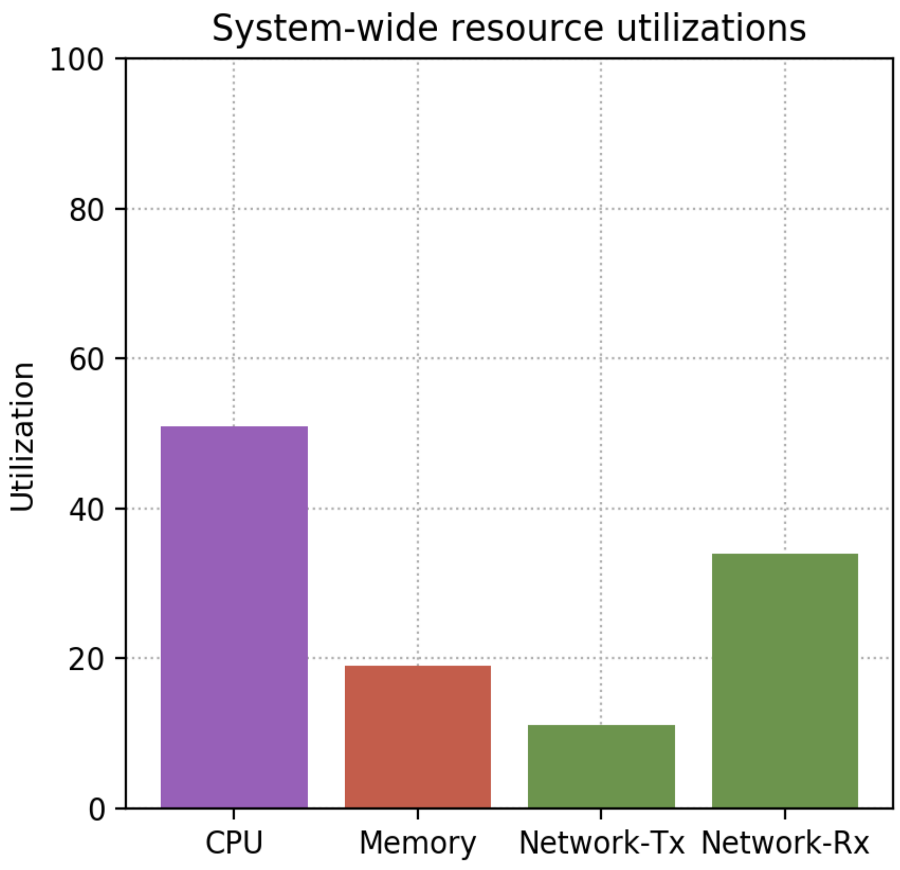
>>

Changes in query load

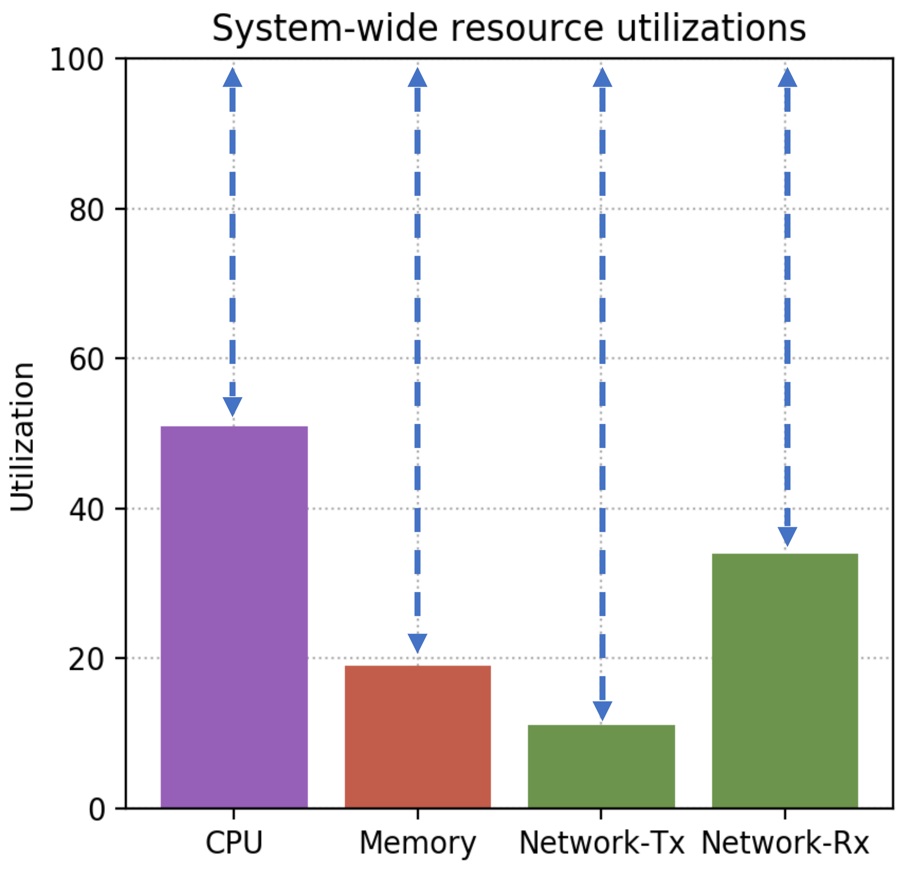
Need finer-grained elasticity in order to better match demand

Resource Utilization

Resource Utilization

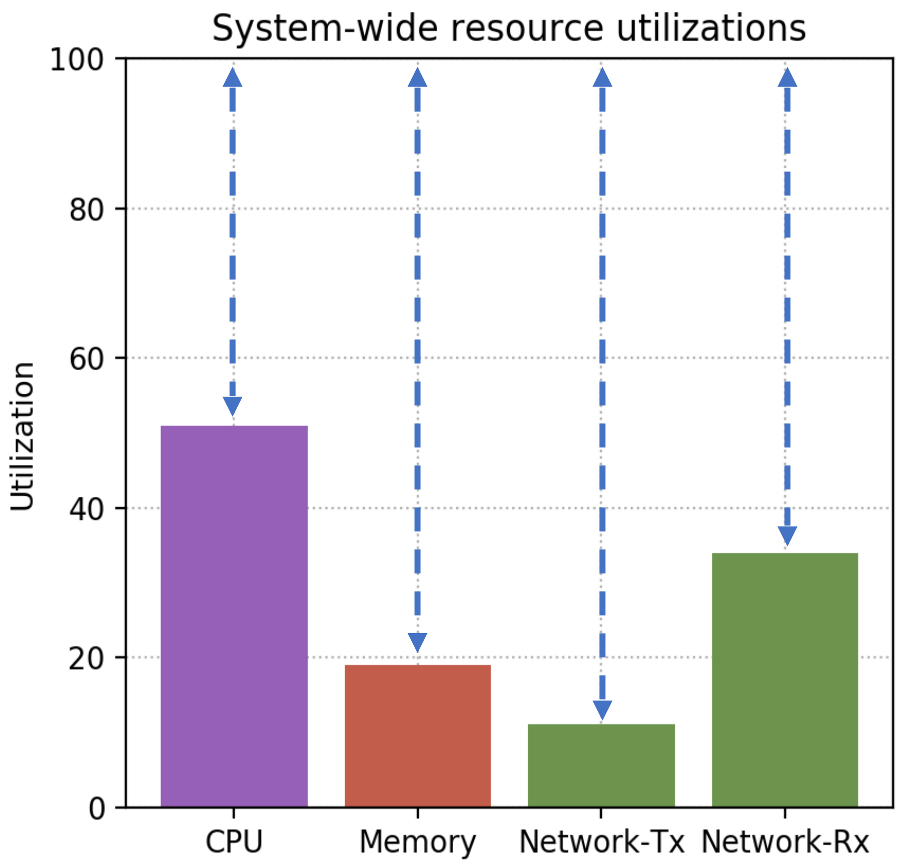


Resource Utilization



Significant room for improvement in resource utilizations

Resource Utilization

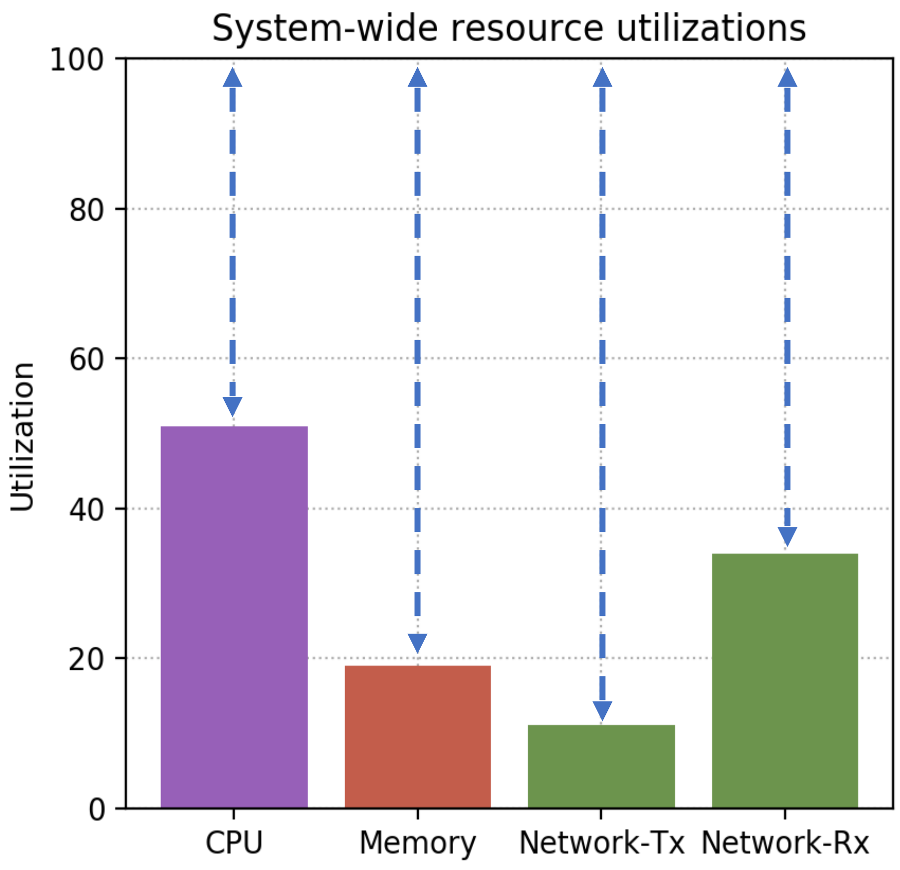


Virtual Warehouse abstraction

- Good performance isolation
- Trade-off: Low resource utilization

Significant room for improvement in resource utilizations

Resource Utilization



Virtual Warehouse abstraction

- Good performance isolation
- Trade-off: Low resource utilization

Solution #1

Finer-grained elasticity with current design

Solution #2

Move to resource shared model

Significant room for improvement in resource utilizations

Alternate design: Resource Sharing

Alternate design: Resource Sharing

Alternate design: Resource Sharing

Alternate design: Resource Sharing

Move to per-second pricing -> pre-warmed pool not cost effective

Alternate design: Resource Sharing

Customer ⟷ snowflake ⟷ aws

per-second | per-second

Move to per-second pricing -> pre-warmed pool not cost effective

~~**Solution #1**~~

~~Finer-grained elasticity with current design~~

Alternate design: Resource Sharing

Customer ⟷ snowflake ⟷ aws

per-second

per-second

Move to per-second pricing -> pre-warmed pool not cost effective

~~**Solution #1**~~

~~Finer-grained elasticity with current design~~

Solution #2

Move to resource shared model

Alternate design: Resource Sharing

Move to per-second pricing -> pre-warmed pool not cost effective

~~Solution #1~~

~~Finer-grained elasticity with current design~~

Solution #2

Move to resource shared model

Statistical Multiplexing

- Better resource utilization
- Helps support elasticity

Alternate design: Resource Sharing

Customer ⟷ per-second ⟷ snowflake ⟷ per-second ⟷ aws

Move to per-second pricing -> pre-warmed pool not cost effective

~~**Solution #1**~~

~~Finer-grained elasticity with current design~~

Solution #2

Move to resource shared model

Statistical Multiplexing

- Better resource utilization
- Helps support elasticity

For 30% of warehouses

Standard deviation >= Mean

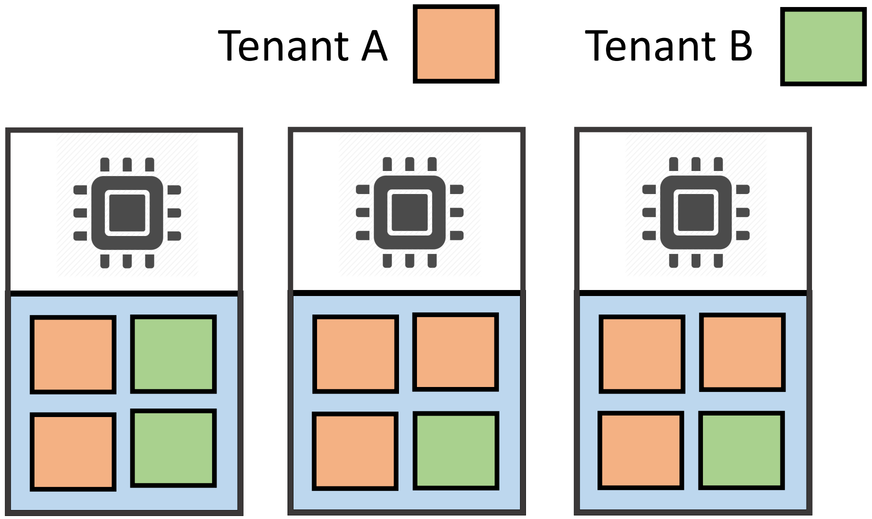
Resource sharing challenges

Resource sharing challenges

- Challenges of moving to a resource shared architecture
  - Maintaining **isolation guarantees**
  - **Shared Ephemeral Storage System**

Resource sharing challenges

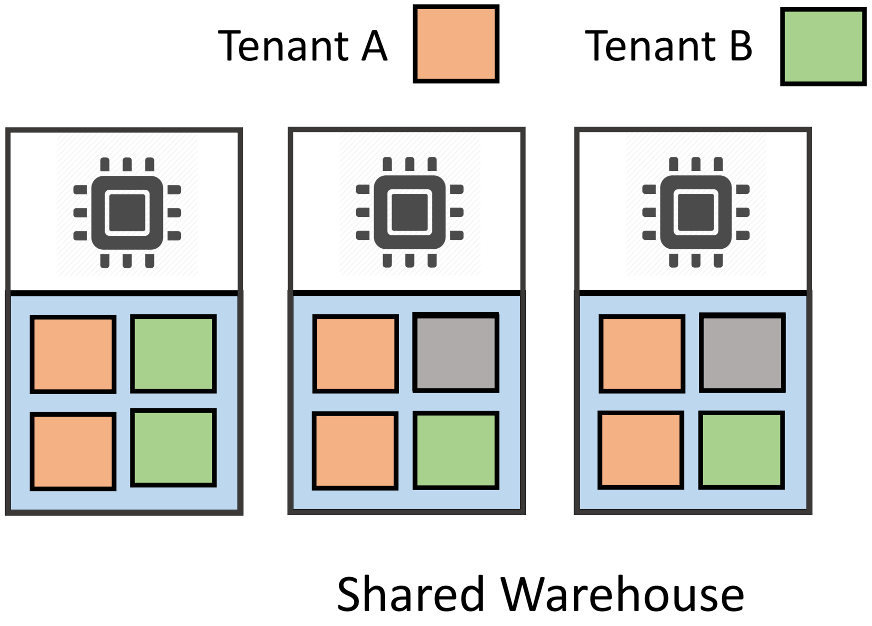
- Challenges of moving to a resource shared architecture
  - Maintaining **isolation guarantees**
  - **Shared Ephemeral Storage System**



Shared Warehouse

Resource sharing challenges

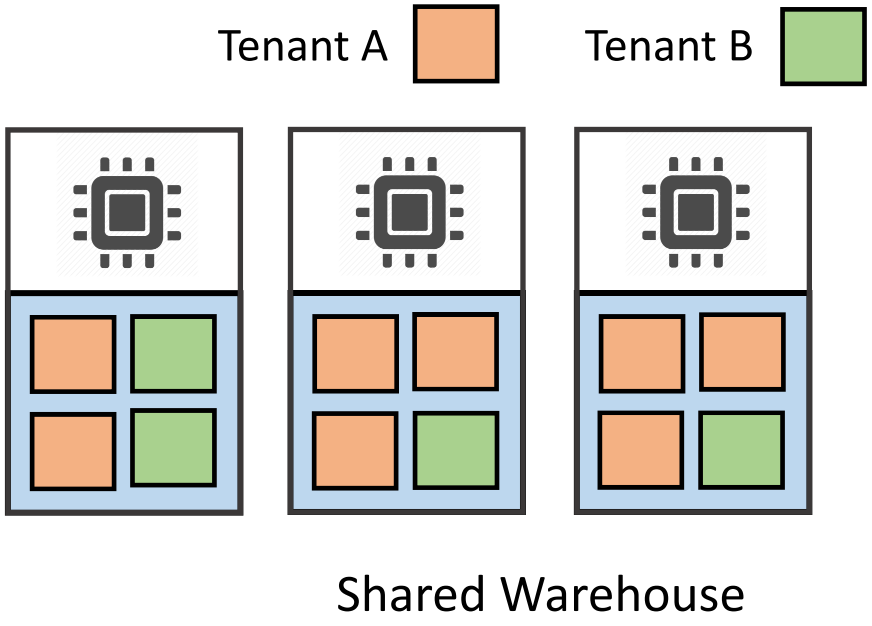
- Challenges of moving to a resource shared architecture
  - Maintaining **isolation guarantees**
  - **Shared Ephemeral Storage System**



- Sharing cache
  - No pre-determined lifetime
  - Co-existence with int. data

Resource sharing challenges

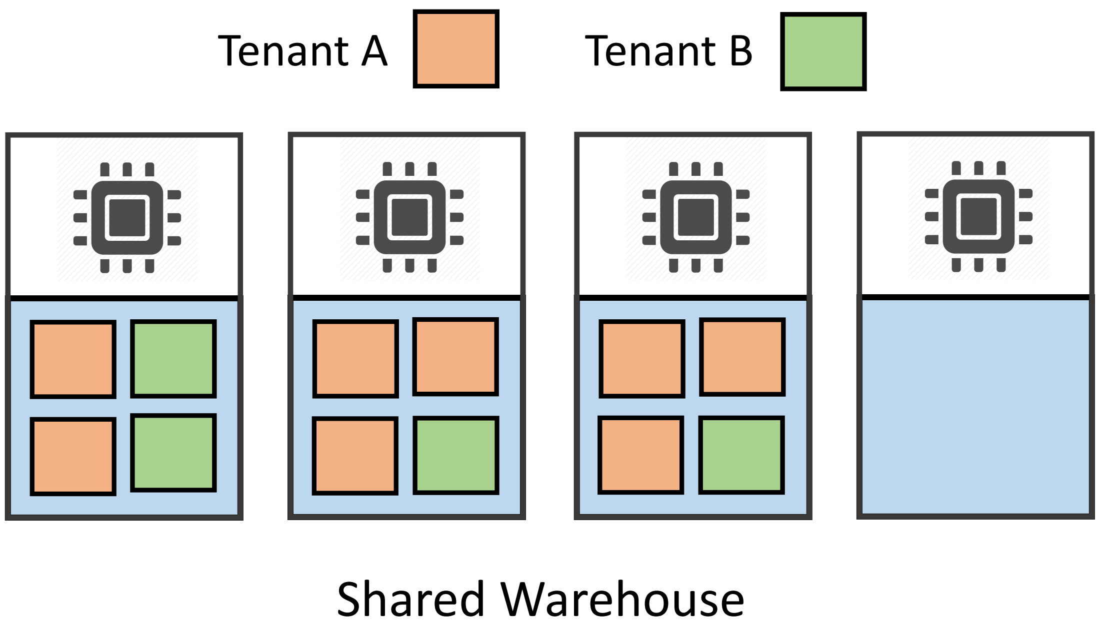
- Challenges of moving to a resource shared architecture
  - Maintaining **isolation guarantees**
  - **Shared Ephemeral Storage System**



- Sharing cache
  - No pre-determined lifetime
  - Co-existence with int. data
- Elasticity without violating isolation
  - Possible cross-tenant interference
  - Need private address-spaces for tenants

Resource sharing challenges

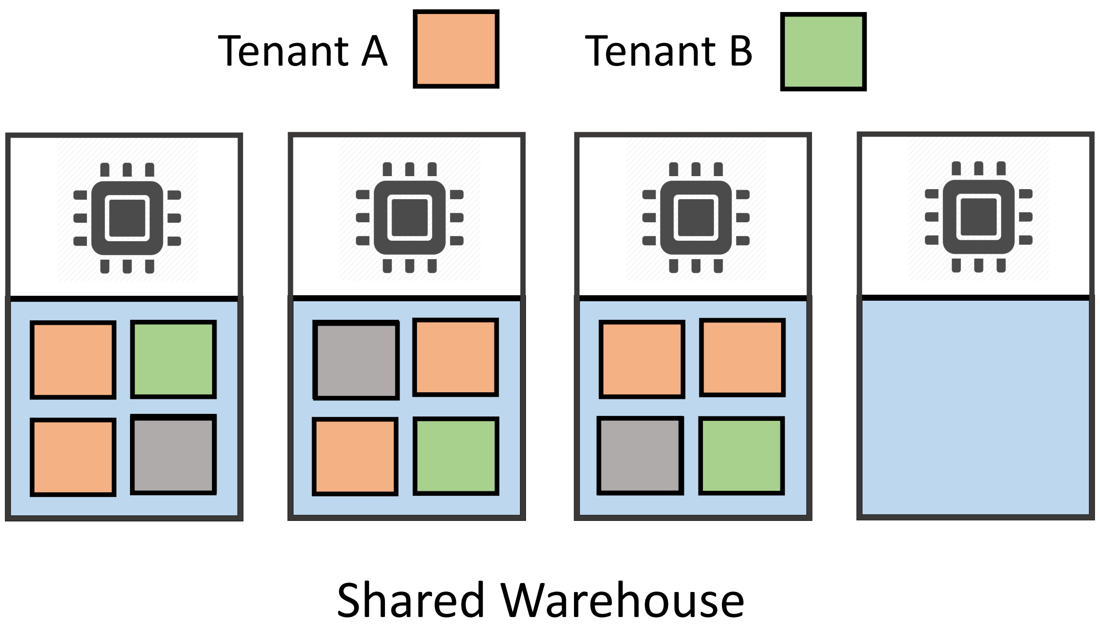
- Challenges of moving to a resource shared architecture
  - Maintaining **isolation guarantees**
  - **Shared Ephemeral Storage System**



- Sharing cache
  - No pre-determined lifetime
  - Co-existence with int. data
- Elasticity without violating isolation
  - Possible cross-tenant interference
  - Need private address-spaces for tenants

Resource sharing challenges

- Challenges of moving to a resource shared architecture
  - Maintaining **isolation guarantees**
  - **Shared Ephemeral Storage System**



- Sharing cache
  - No pre-determined lifetime
  - Co-existence with int. data
- Elasticity without violating isolation
  - Possible cross-tenant interference
  - Need private address-spaces for tenants

Conclusion

Design aspects

Data-driven insights

Future Directions

Conclusion

Design aspects

Data-driven insights

Future Directions

Conclusion

Design aspects

Data-driven insights

Future Directions

- **Dataset publicly released**
  - https://github.com/resource-disaggregation/snowset

Thank You

Questions?