Application-Centric SSD Cache Allocation for Hadoop Applications

Zhen Tang, Institute of Software, Chinese Academy of Sciences

Wei Wang, Institute of Software, Chinese Academy of Sciences

Yu Huang, Nanjing University

Heng Wu, Institute of Software, Chinese Academy of Sciences

Jun Wei, Institute of Software, Chinese Academy of Sciences

Tao Huang, Institute of Software, Chinese Academy of Sciences

September 23, 2017

ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences

NANJING UNIVERSITY 1902

SSD vs HDD

Higher Performance, especially in random access

Expensive in Per GB Capacity

Large Capacity

Lower random IOPS (I/O Operations Per Second)

SSD Caching System is a balance between **Cost** and **Performance**

SSD Caching System

IO Requests

Data Mapping Module

Admission Policy

SSD 240 GB
20000 IOPS
600 MB/s

HDD 4TB
200 IOPS
150 MB/s

The Data Mapping Module is the Fundamental Part

SSD Cache is Widely Used in the Virtualization Environment

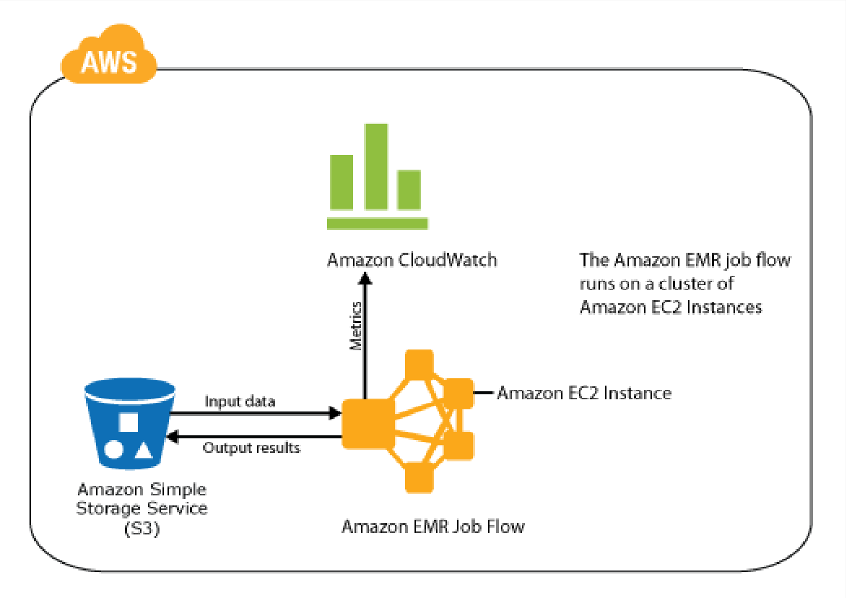
Amazon Elastic Block Store (EBS)
Supporting Amazon EC2

	Solid State Drives (SSD)		Hard Disk Drives (HDD)	
Volume Type	EBS Provisioned IOPS SSD (io1)	EBS General Purpose SSD (gp2)*	Throughput Optimized HDD (st1)	Cold HDD (sc1)
Short Description	Highest performance SSD volume designed for latency-sensitive transactional workloads	General Purpose SSD volume that balances price performance for a wide variety of transactional workloads	Low cost HDD volume designed for frequently accessed, throughput intensive workloads	Lowest cost HDD volume designed for less frequently accessed workloads
Use Cases	I/O-intensive NoSQL and relational databases	Boot volumes, low-latency interactive apps, dev & test	Big data, data warehouses, log processing	Colder data requiring fewer scans per day
API Name	io1	gp2	st1	sc1
Volume Size	4 GB - 16 TB	1 GB - 16 TB	500 GB - 16 TB	500 GB - 16 TB
Max IOPS**/Volume	20,000	10,000	500	250
Max Throughput/Volume	320 MB/s	160 MB/s	500 MB/s	250 MB/s
Max IOPS/Instance	75,000	75,000	75,000	75,000
Max Throughput/Instance	1,750 MB/s	1,750 MB/s	1,750 MB/s	1,750 MB/s
Price	$0.125/GB-month $0.065/provisioned IOPS	$0.10/GB-month	$0.045/GB-month	$0.025/GB-month
Dominant Performance Attribute	IOPS	IOPS	MB/s	MB/s

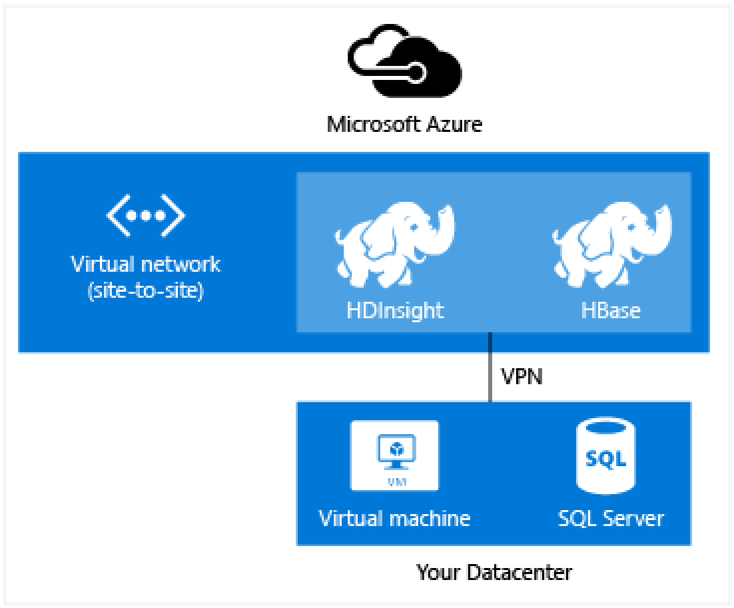
	EBS Magnetic
Volume Type	EBS Magnetic
Use Case	Infrequent Data Access
API Name	standard
Volume Size	1 GB - 1 TB
Max IOPS/Volume	40-200
Max IOPS Burst Performance	-
Max Throughput/Volume	40-90 MB/s
Max Throughput Burst Performance	-
Max IOPS/Instance	48,000
Max Throughput/Instance	800 MB/s
Price	$0.05/GB-month $0.05/million I/O

And Supports the Elastic Hadoop Clusters

Amazon EMR



Microsoft HDInsight



The VM View and the Application View

VM view: IO Latency
Application view: **Job Completion Time**

Node-1
HDFS DataNode

Node-2
HDFS NameNode

Node-3
HDFS DataNode

Node-4
HDFS DataNode

Node-5
HDFS DataNode

5 Nodes
Total Working Set = 20GB
Total SSD Cache Space = 5GB

VM-centric Approach

Allocating the SSD cache according to the working set , aiming to minimize per-VM IO latency

Node-1
HDFS DataNode

Working Set 5GB
Cache Size 1.25GB (75%)

Node-2
HDFS NameNode

Working Set 1GB
Cache Size 0.25GB (75%)

Node-3
HDFS DataNode

Working Set 4GB
Cache Size 1GB (75%)

Node-4
HDFS DataNode

Working Set 3GB
Cache Size 0.75GB (75%)

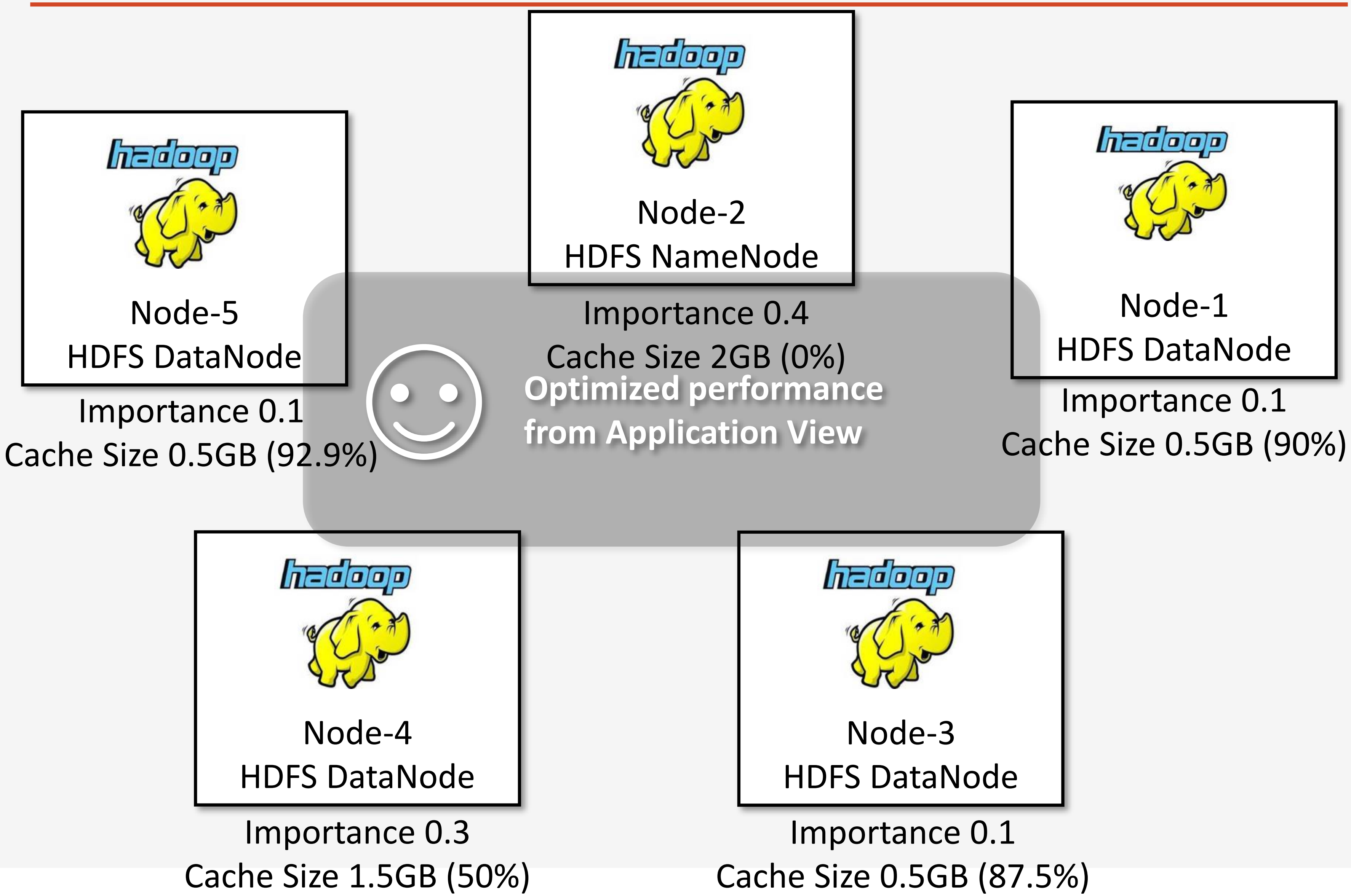
Node-5
HDFS DataNode

Working Set 7GB
Cache Size 1.75GB (75%)

Guarantee the fairness from VM view

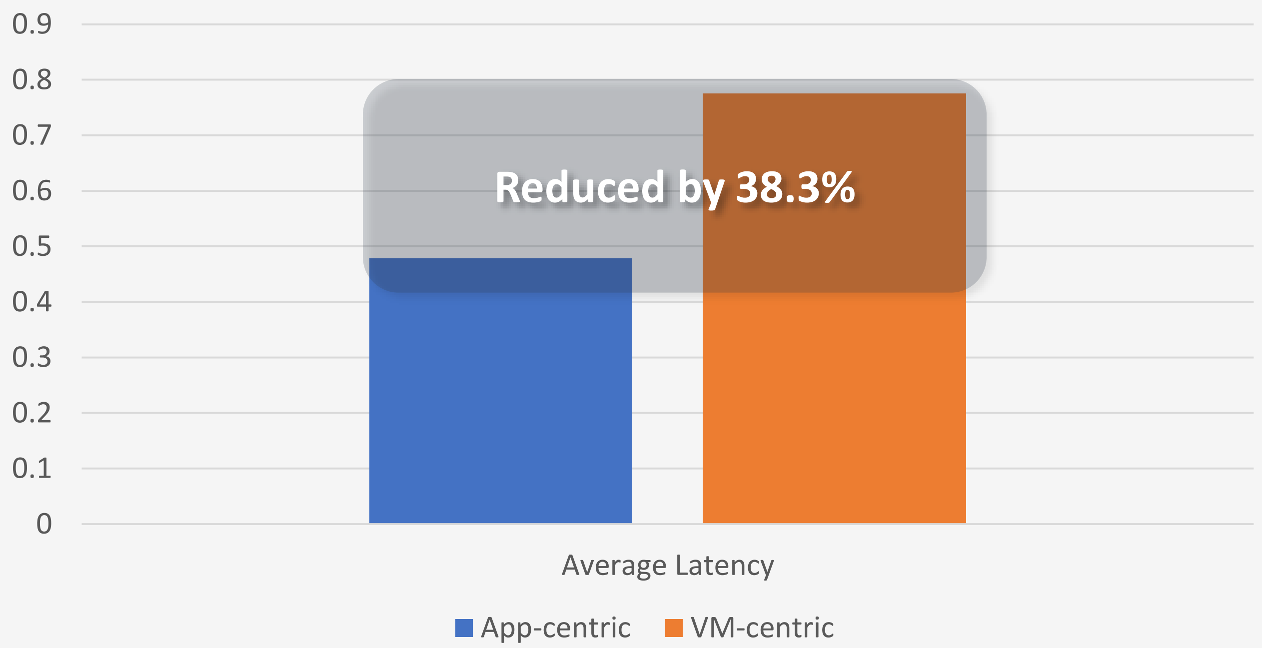
Optimized performance for individual VMs

However, as the importance of nodes are different



Improve Application-Level Performance

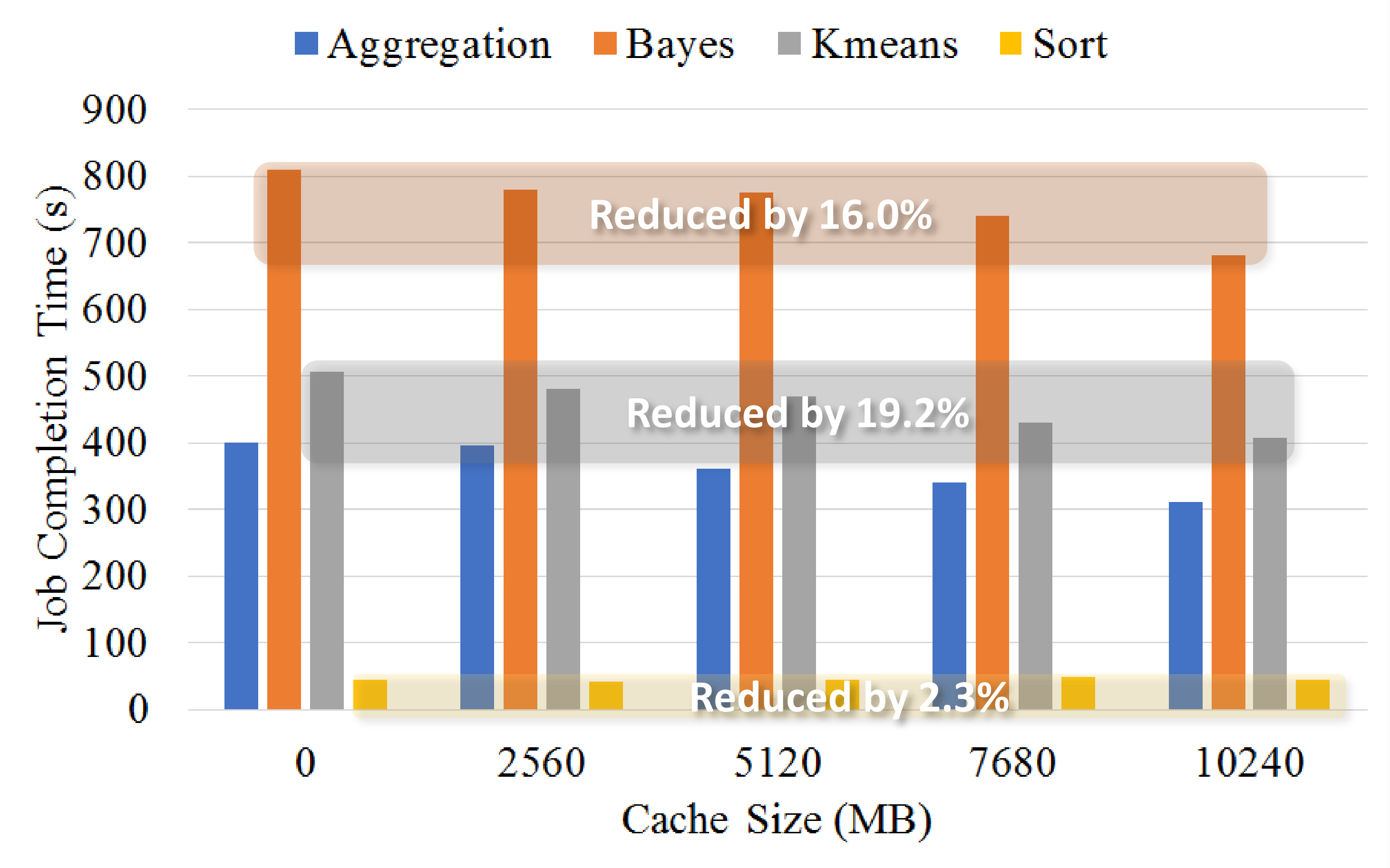
- Latency related to job execution = Importance(VM) * Latency(VM)
  - VM-centric: $0.775l_{HDD}$
  - App-centric: $0.47836l_{HDD}$
- Average latency related to job execution **reduced by 38.3%**



The relationships among VMs inside the application cannot be ignored

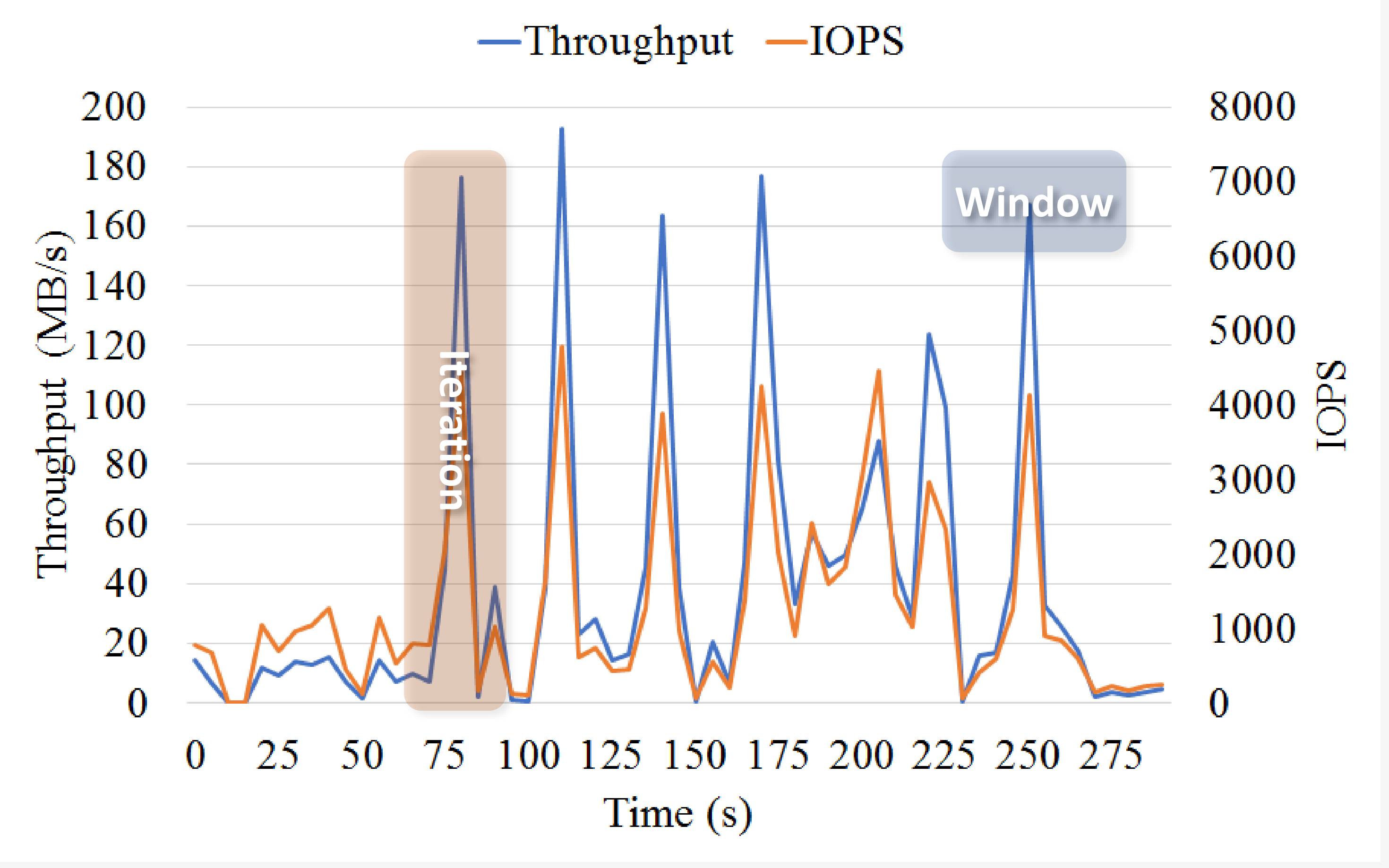
Two Principals

Allocating too much SSD Cache is unnecessary due to different importance



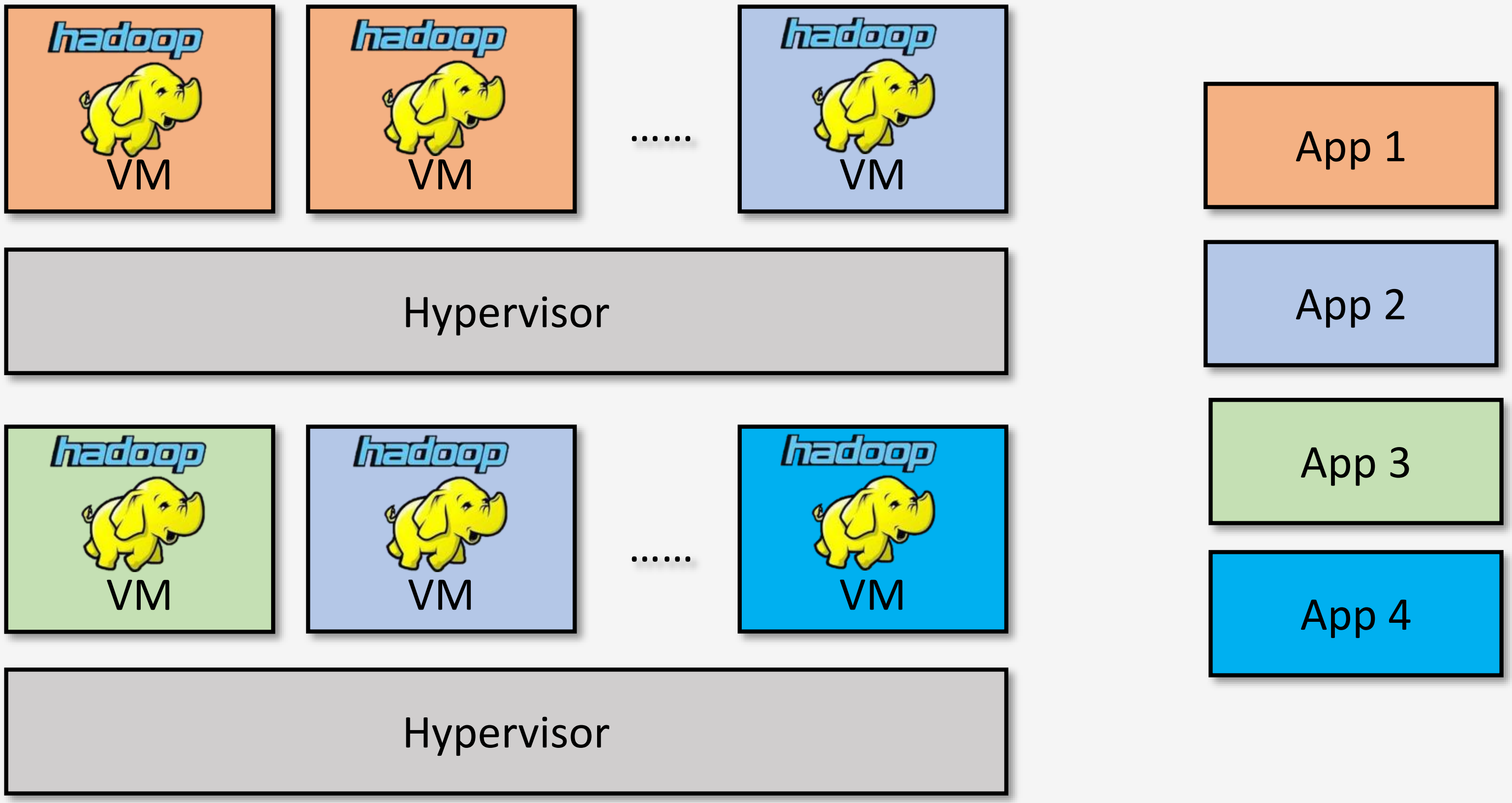
Two Principals

For different stages of the workload, the requirements may be different



We Also Need Global View

One application may be deployed on multiple hypervisors
Multiple applications may be deployed on one hypervisor
We need SSD cache allocation **from the global view**



How to allocate SSD cache from application view?

- Problem: How to allocation per-VM SSD cache for elastic Hadoop clusters, to **reduce the job completion time**?
- Solution: Application-centric SSD cache allocation (AC-SSD)
- Two challenges
  - How to **allocate** appropriate SSD cache resources (space and IOPS) for virtual machines inside the Hadoop cluster from the application view?
  - How to **dynamically change** the plan to adapt to continuously and dynamically changing workloads?

Application-**C**entric **SSD** Cache Allocation (AC-SSD)

- To figure out the importance and allocate per VM SSD cache
  - Use **genetic algorithm** to calculate the nearly optimal weights for VMs based on importance
  - Weight based SSD cache allocation
- To react to rapidly changing workload
  - Use **closed loop adaptation**

Genetic Algorithm

- Proving that the per VM SSD cache allocation is NP complete (more details in paper)
- Definition
  - **Chromosome**: Tuple $\{w_{Storage}, w_{IOPS}\}$, indicates weights of SSD cache space and IOPS
  - **Genome**: A set of chromosomes, indicates the SSD cache allocation plan
  - **Selection**: Select genomes randomly by fitness
  - **Crossover**: Select random numbers of chromosome of two genomes and swap them
  - **Mutation**: Change the tuple of chromosome within a specific range

Fitness Calculation

- We calculate the fitness from 3 levels
  - **Importance Factor**: Indicates the contribution of VMs.
  - **IO time**: Indicates whether the workload is IO sensitive.
  - **Average request size**: indicates the access pattern, whether sequential or random.

Importance Factor

- For the execution of Hadoop application
  - The nodes require data **locally** and **from other nodes**
  - The **data dependency** indicates the contribution to the job completion time
- Based on the observation
  - We use the **ratio** of network throughput to IO throughput the importance factor

VM 1
IO throughput = 500 MB
Network throughput = 250 MB
Ratio=0.5

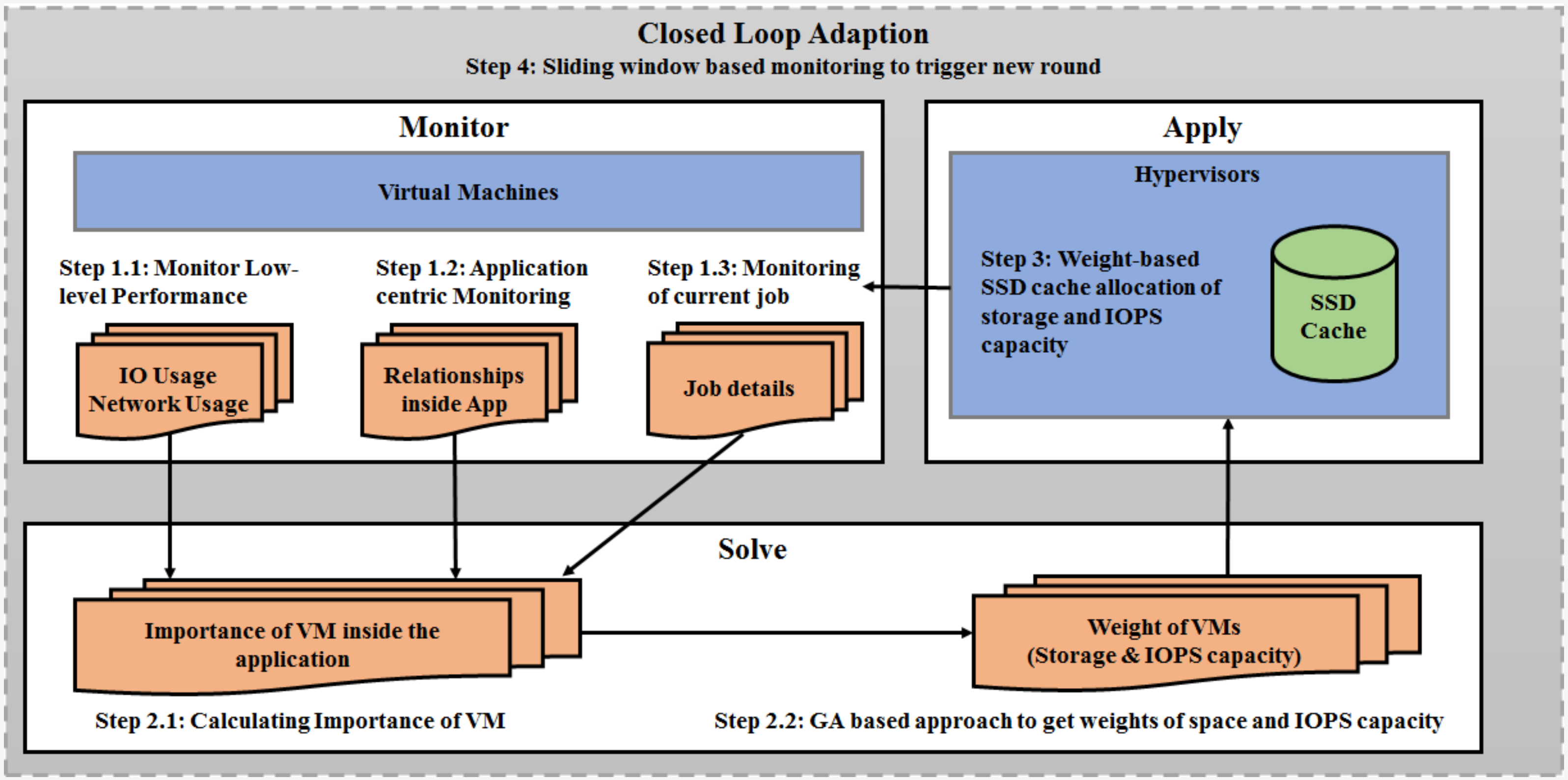
VM 2
IO throughput = 1000 MB
Network throughput = 200 MB
Ratio=0.2

VM 3
IO throughput = 800 MB
Network throughput = 640 MB
Ratio=0.8

We prefer this one!

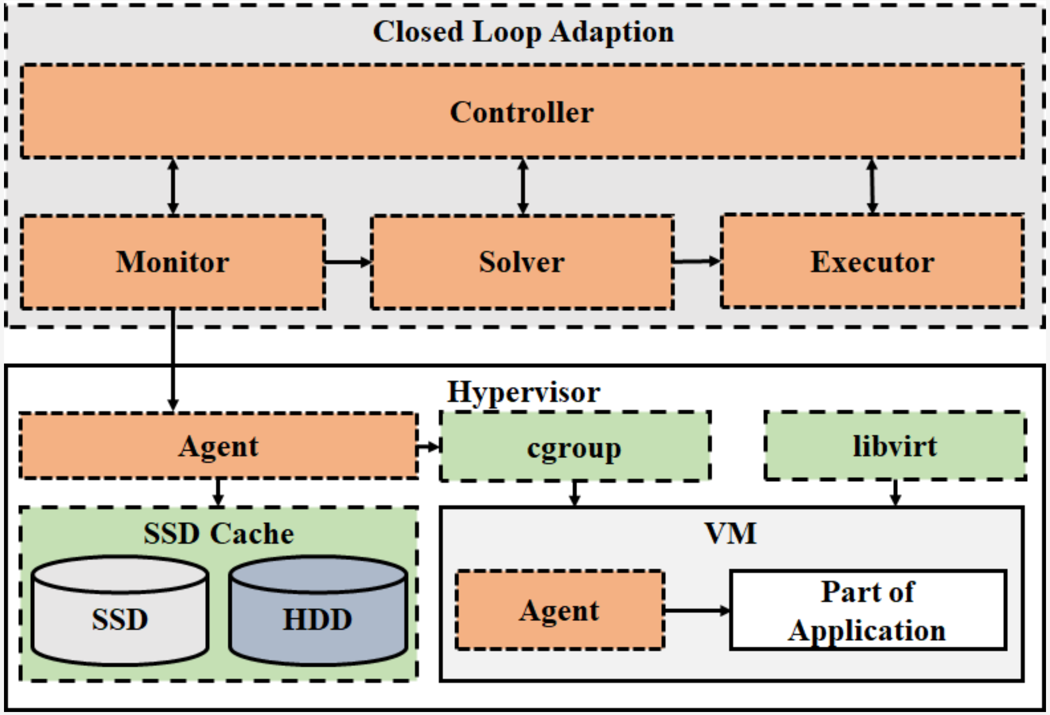
Closed Loop Adaptation

- Consists of 3 main steps: Monitor, Solve and Apply



Implementation

- We implement AC-SSD on Xen hypervisor
  - Supporting closed loop adaptation: Java based controller, monitor, solver, executor; agents on VMs and hypervisor
  - Use cgroup to control the weight of IOPS
  - LRU based SSD cache

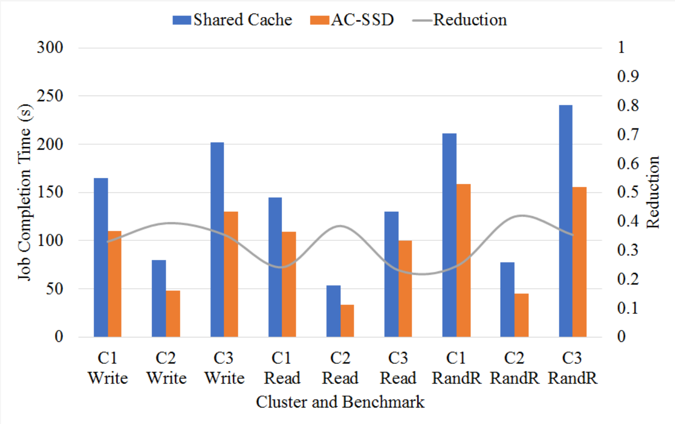


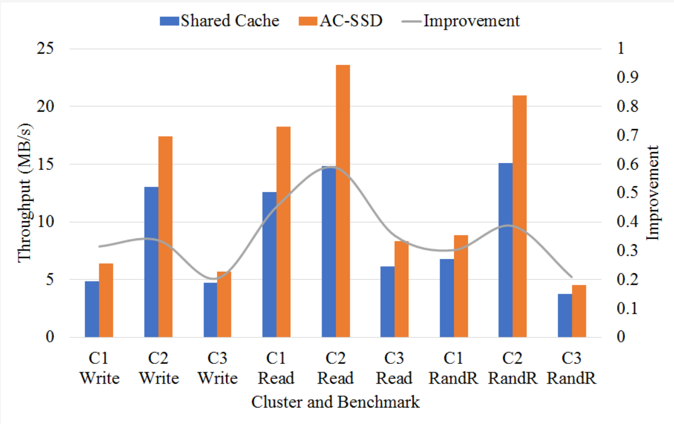
Experiment Setup

- Environment
  - 20 VMs hosted on 4 hypervisors
  - 3 Clusters of 5, 10 and 5 nodes, with different VM placements
  - 640MB SSD cache for each hypervisor
- Benchmarks
  - IO Sensitive (TestDFSIO)
  - Hadoop micro benchmarks (Sort, Terasort, Wordcount)
  - SQL micro benchmarks (Aggregation, Join, Scan)
  - Machine learning micro benchmarks (Bayes, KMeans, PageRank)

Result - IO sensitive

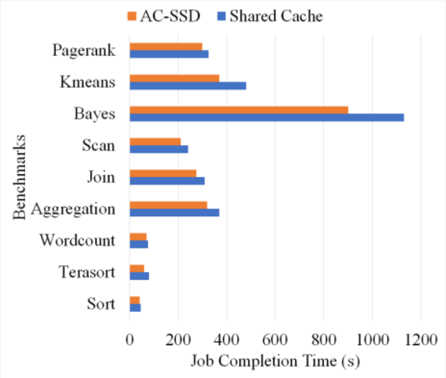
- Compared to shared cache
- For IO sensitive workloads (TestDFSIO)
  - Job completion time reduced by 31% in average
  - Throughput improved by 35% in average
  - Better for read workloads

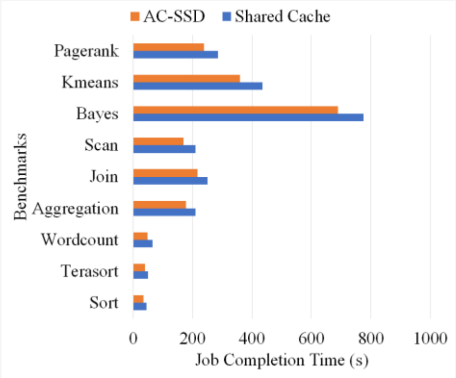


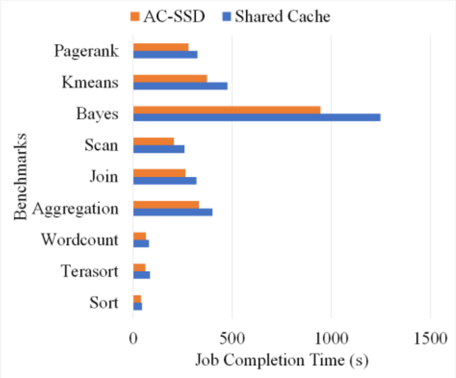


Result - Benchmarks

- Compared to shared cache
- For micro benchmarks
  - Job completion time reduced by 14.3%~17.8%
  - Works better for Kmeans and Bayes benchmark



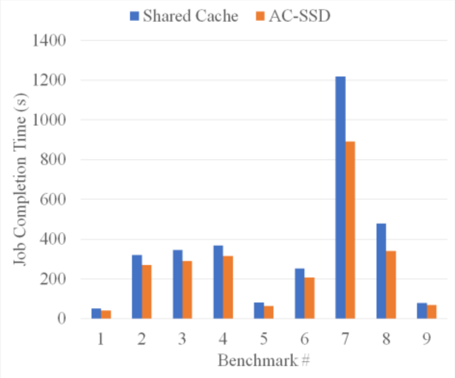


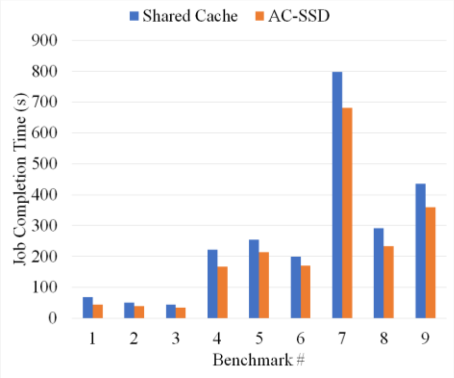


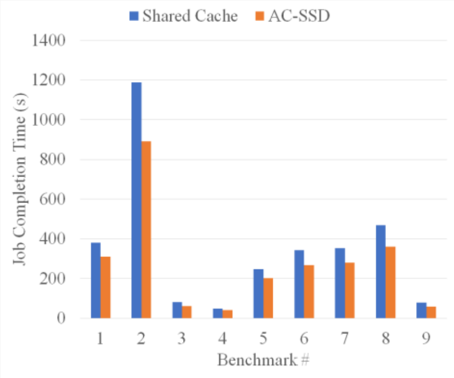
Result - Self Adaptation

- Compared to shared cache
- For rapid changing workloads
  - Reduced by ~20% for rapidly changing workloads, for 3 clusters

No.	Workload - C1	Workload - C2	Workload - C3
1	Sort	Wordcount	Aggregation
2	Join	Terasort	Bayes
3	Pagerank	Sort	Wordcount
4	Aggregation	Aggregation	Sort
5	Terasort	Join	Scan
6	Scan	Scan	Join
7	Bayes	Bayes	Pagerank
8	KMeans	Pagerank	KMeans
9	Wordcount	KMeans	Terasort







Conclusion

- We present AC-SSD, an Application-Centric SSD caching system
  - Present the importance factor of VMs inside the application, based on the network throughput and disk IO
  - Use genetic algorithm to calculate the nearly optimal weight of VMs
  - Use closed loop adaptation to react to rapidly changing workloads
  - The evaluation shows that it reduces the job completion time comparing to the shared cache

Thanks

Zhen Tang

tangzhen12@otcaix.iscas.ac.cn