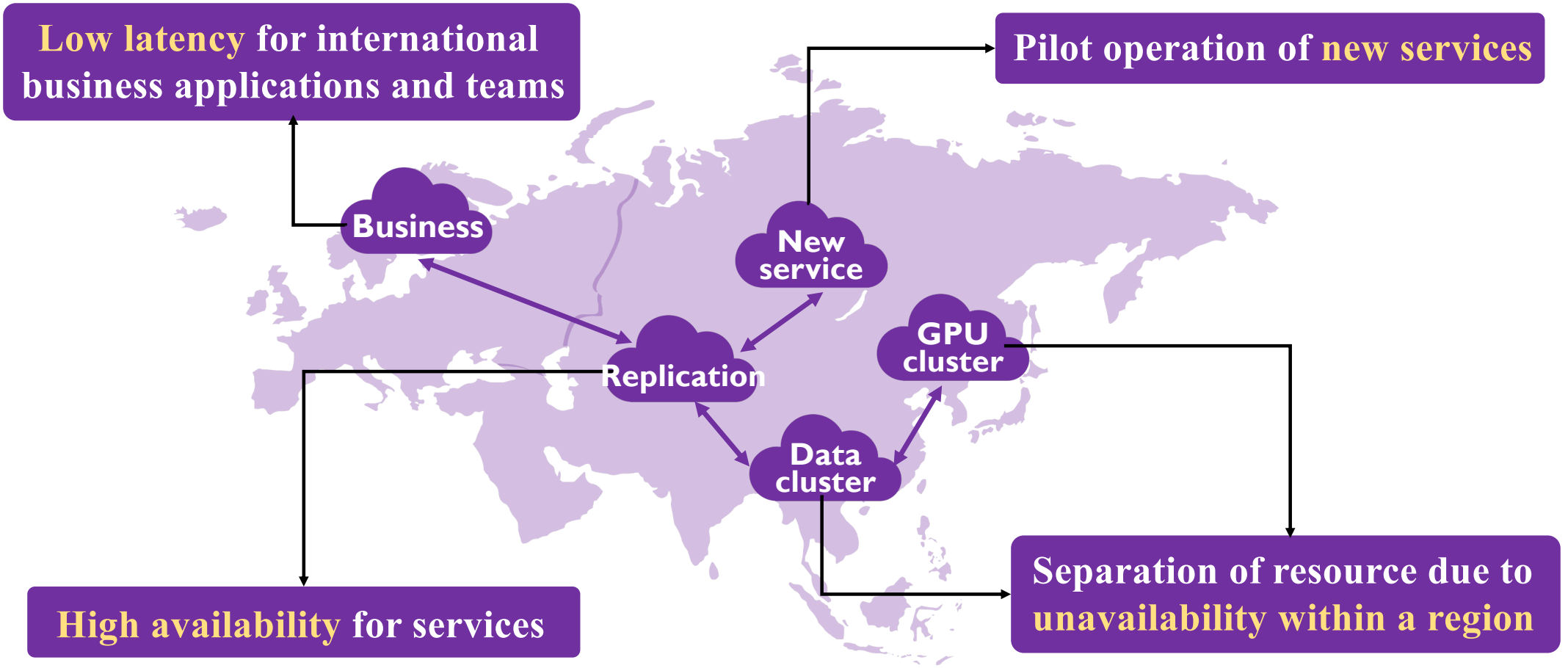
Reducing Cross-Cloud/Region Costs with the Auto-Configuring MACARON Cache

Hojin Park, Ziyue Qiu, Gregory R. Ganger, George Amvrosiadis
Carnegie Mellon University, Uber

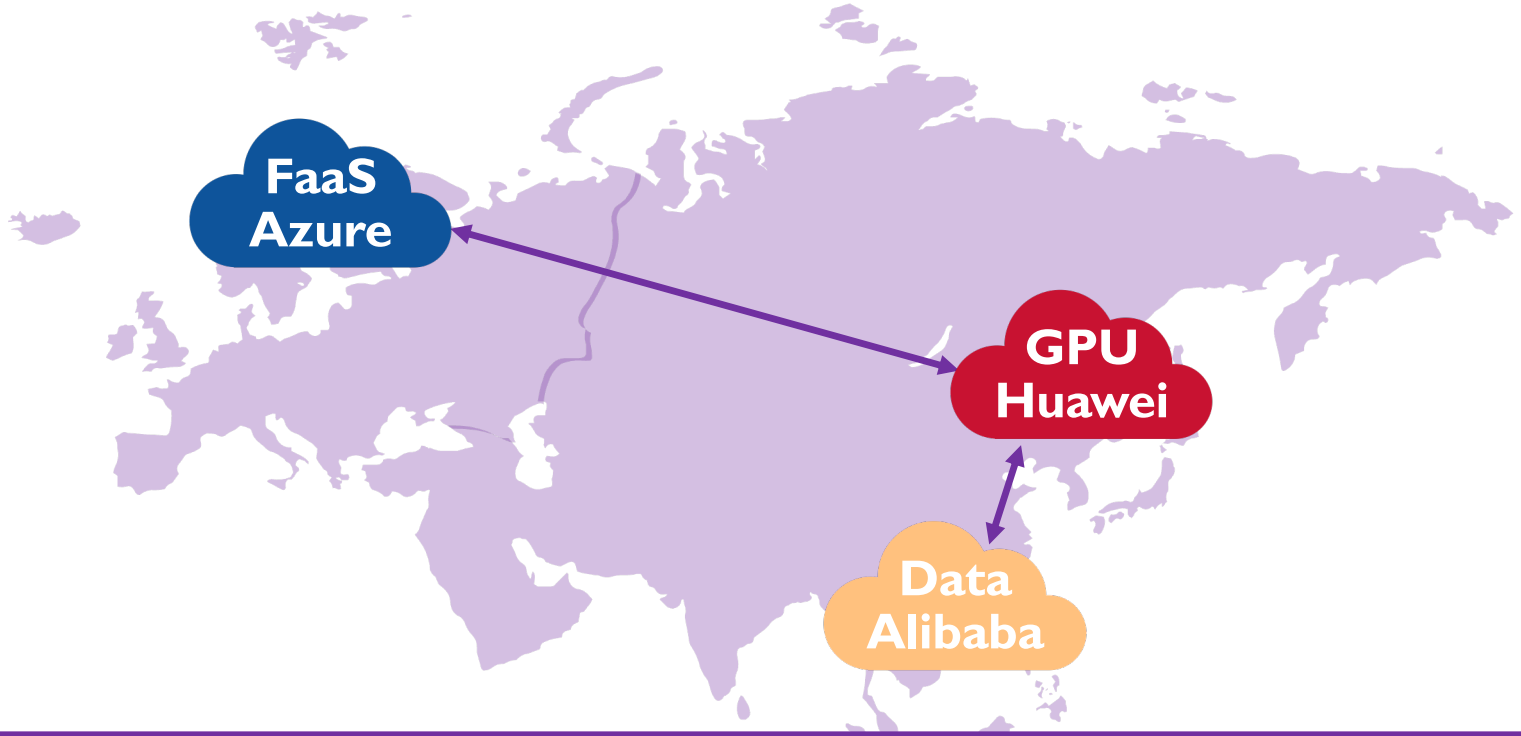
Speaker: Chao Bi

SOSP' 24

Public cloud deployment – Cross Region



Public cloud deployment – Cross Cloud



Multiple clouds saves cost due to price differences between providers

55% of multi-cloud users deploy a single workload across multiple clouds

Challenge

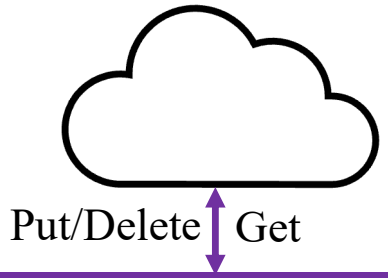
- Prohibitive data egress cost
  - In IBM trace (access 694TB/week), $64K/week for cross-cloud data transfer and $14K/week for cross-region data transfer

Operation / Price (¢)	AWS	Azure	GCP
Egress to Internet (per GB)	9	8.7	11
Egress btw. regions (per GB)	2	2	2
Object storage (per GB-month)	2.3	2.1	2.3
DRAM (per GB-month)	700-1200		
Object GET (per 1k requests)	0.04	0.05	0.04
Object PUT (per 1k requests)	0.5	0.65	0.5

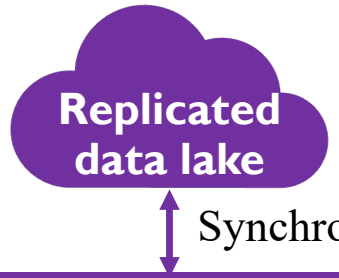
- High access latency
  - Time is money in cloud services
  - ~100ms for fetching 1KB object from U.S. to Europe

Current Approaches

Remote data access



Full replication



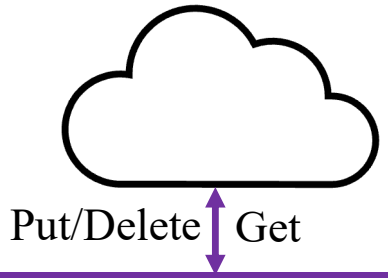
Remote data lake

Egress cost
Access latency

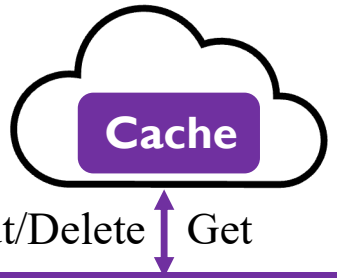
Synchronization cost
Excess capacity cost

Current Approaches

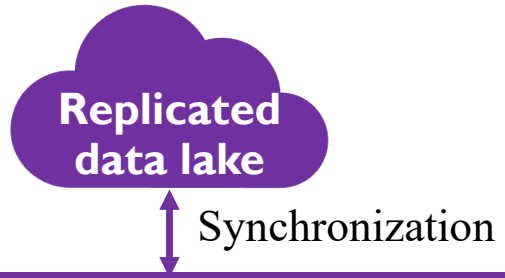
Remote data access



ECPC
(Elastic Cloud Provider Caching)



Full replication



Remote data lake

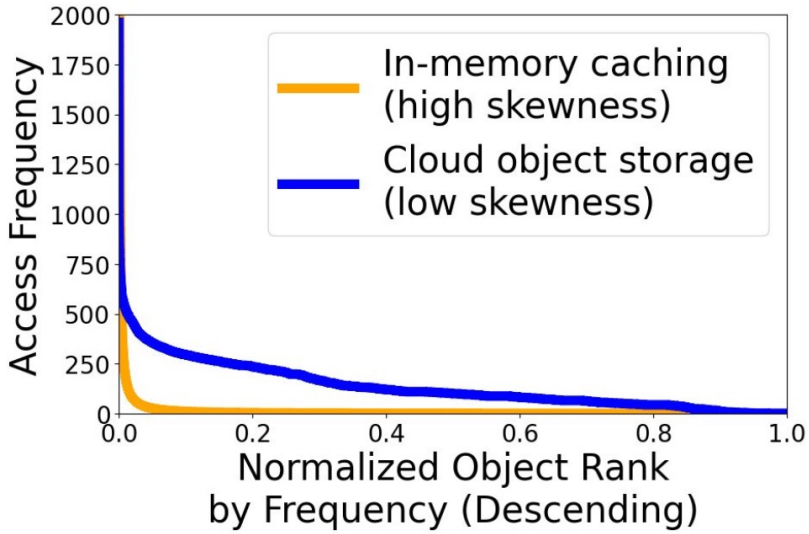
Egress cost	DRAM capacity cost	Synchronization cost
Access latency	Manual configuration	Excess capacity cost

Design objective1: We need auto-configuring cache which optimize both cost and performance

Workload Characteristics

- Cloud object storage workloads are skewed towards low access per object
- Many object in cloud object storage workloads are large
  - Median object is 10-100KB in IBM trace, while for Twitter is 20-30B

Trace	Operation %		Skewness (Zipf factor)	Total data size
	Put	Get		
IBM 9	N/A	100	0.22	6 TB
IBM 12	1	99	0.97	5 TB
IBM 18	2	98	0.64	4 TB
IBM 55	55	45	0.42	13 TB
IBM 83	40	60	0.72	64 TB
IBM 96	58	42	0.20	78 TB
Uber	N/A	100	0.52	324 TB
VMware	N/A	100	0.47	215 GB



Design objective2: To reduce high data egress cost we need large cache capacities, which are feasible by leveraging cheap storage type

Workload Characteristics

- Diverse and dynamic data access patterns
  - For IBM 80 (dynamicity), dynamically adjust cache size results in 85% cost reduction
  - For Uber (stable, periodic data access patterns), result in a 15% cost reduction

Trace	Operation %		Skewness	Total	Data accessed		Remarks
	Put	Get	(Zipf factor)	data size	Put	Get	
IBM 9	N/A	100	0.22	6 TB	0	34 TB	Short lifetime: last access - first access < 10min
IBM 12	1	99	0.97	5 TB	4 TB	603 TB	High data access repetitiveness
IBM 18	2	98	0.64	4 TB	231 GB	14 TB	High request rate, small object sizes
IBM 55	55	45	0.42	13 TB	12 TB	10 TB	Strong diurnal access pattern
IBM 83	40	60	0.72	64 TB	37 TB	94 TB	Low compulsory miss ratio (=0.12)
IBM 96	58	42	0.20	78 TB	46 TB	36 TB	High compulsory miss ratio (=0.87)
Uber	N/A	100	0.52	324 TB	0	941 TB	Stable data access pattern
VMware	N/A	100	0.47	215 GB	0	71 TB	Small total data size, high request rate for testing

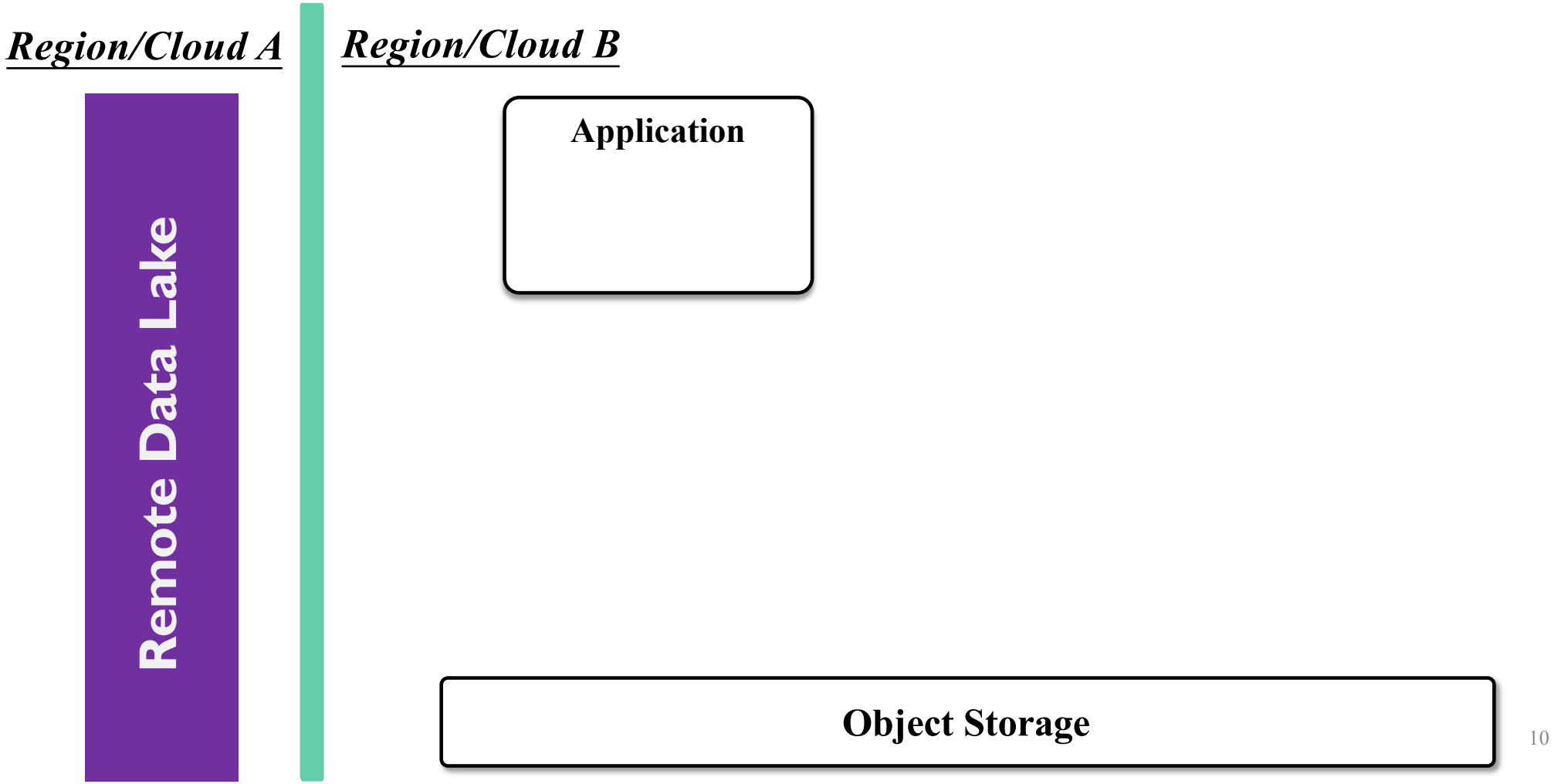
Design objective3: Re-configuration based on workload monitoring is necessary to accommodate diverse and evolving data access patterns.

Design Characteristics

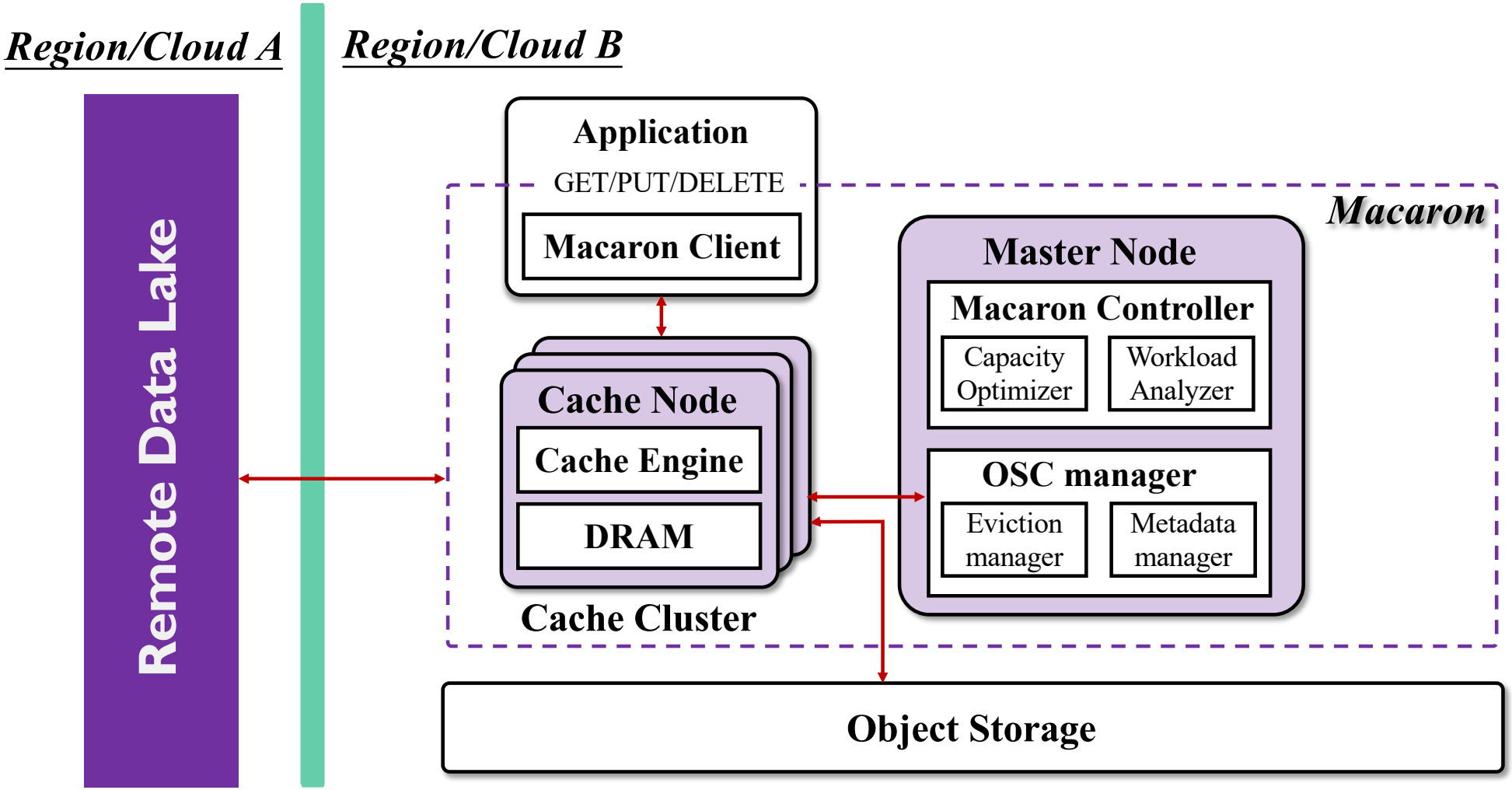
- Two-level caching
  - **For latency:** DRAM (expensive), use the smallest size that meets requirements
  - **For cost:** object storage (300x cheaper than DRAM)

- Adaptive cache reconfiguration
  - Periodically analyze data access patterns and adjust capacity of each cache level
  - Assuming patterns will repeat in the future

Architecture

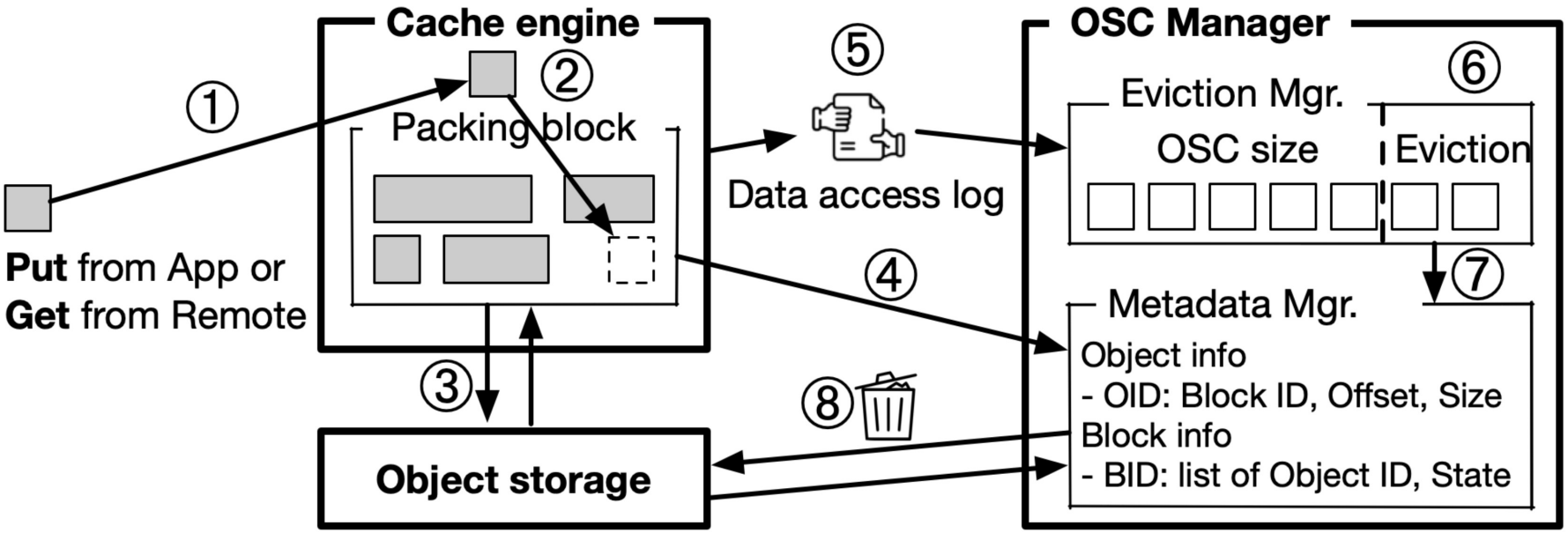


Architecture – Data Transfer

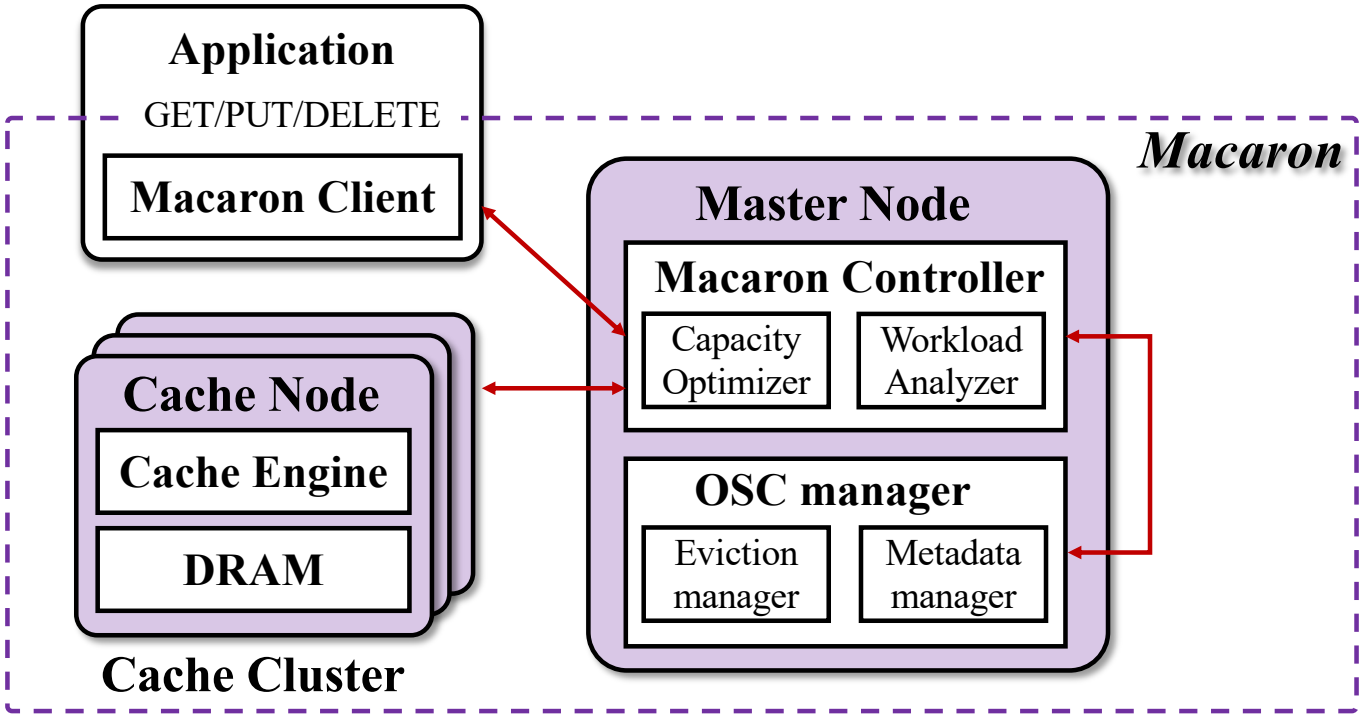


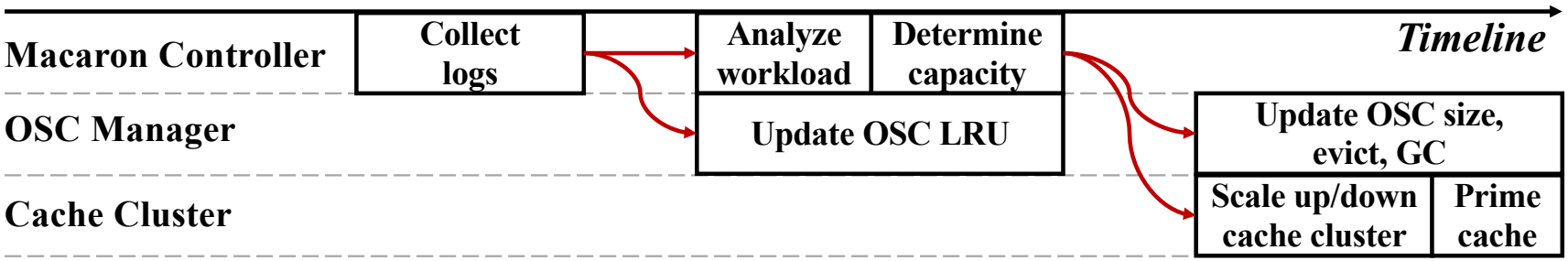
OSC manager

- Object packing
  - Object storage write operations are ~13x more expensive than reads
- Lazy eviction and garbage collection
  - Blocks with over 50% evicted/deleted objects will be reorganized



Architecture – Control Messages

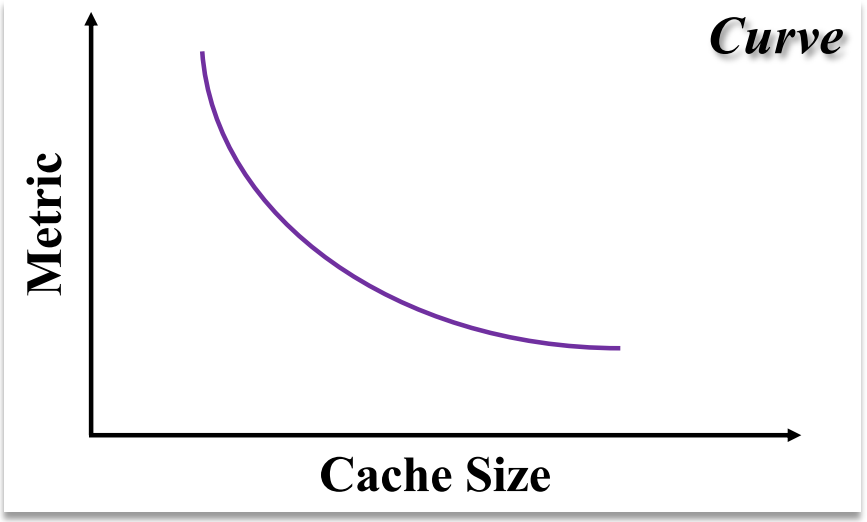




Workload Analyzer – Miniature Simulation

- Key metrics:
  - MRC (Miss Ratio Curve), BMC (Byte Miss Curve), ALC (Average Latency Curve)

- Miniature simulation
  - Generating curves that emulates caches of any specified size by proportionally scaling down the actual cache size and using spatial sampling to sample data accesses



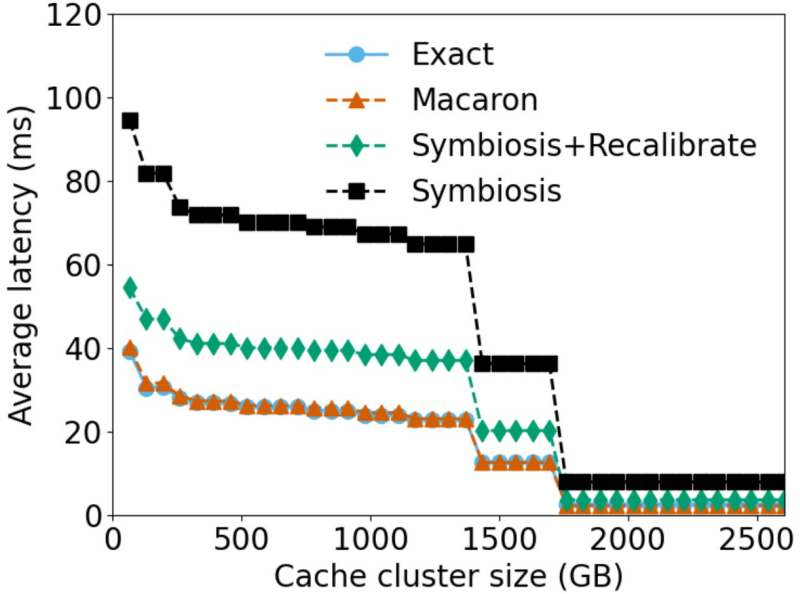
Workload Analyzer – Miniature Simulation

- Key metrics:
  - MRC (Miss Ratio Curve), BMC (Byte Miss Curve), ALC (Average Latency Curve)
- Miniature simulation
  - Generating curves that emulates caches of any specified size by proportionally scaling down the actual cache size and using spatial sampling to sample data accesses
- Issues of ALC in Symbiosis[1]
  - Assuming cache and disk latency don't change
  - Assuming subsequent accesses after the first one are classified as hits

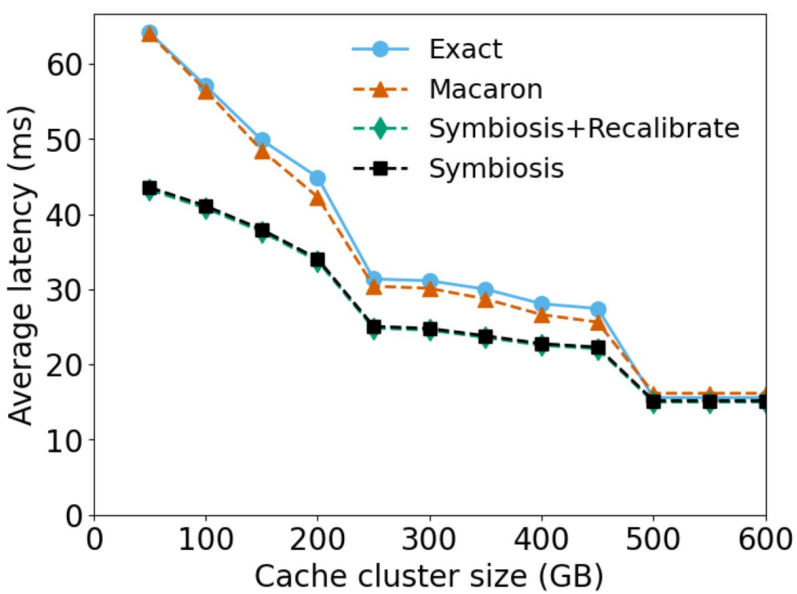
[1] Symbiosis: The art of application and kernel cache cooperation. (FAST'24)

Workload Analyzer – Miniature Simulation

- Macaron
  - Directly compute average latency for each access during miniature simulation
  - Add the request delay in the simulation
  - Use serverless function to deploy (~30s)



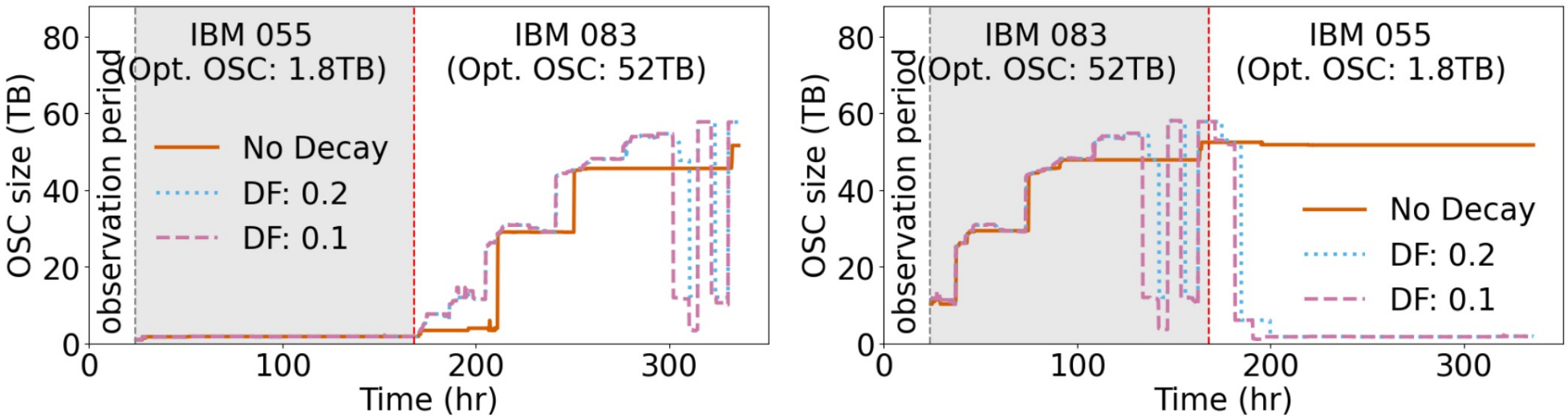
ALC of IBM 55



ALC of IBM 12

Workload Analyzer – Metric Aggregation

- For performance (ALC), only the latest access pattern is important
- For cost (MRC, BMC), store the results for future reference
  - Cloud data access exhibits periodicity
  - Use an exponential decay mechanism by multiplying metrics by a factor $\gamma^{elapsed\_time}$



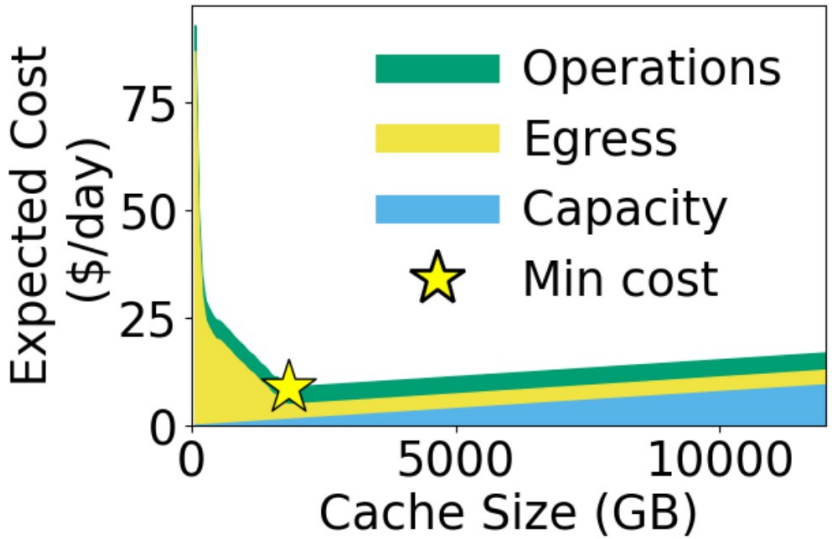
Testing macaron's adaptivity to a workload change

Capacity Optimizer

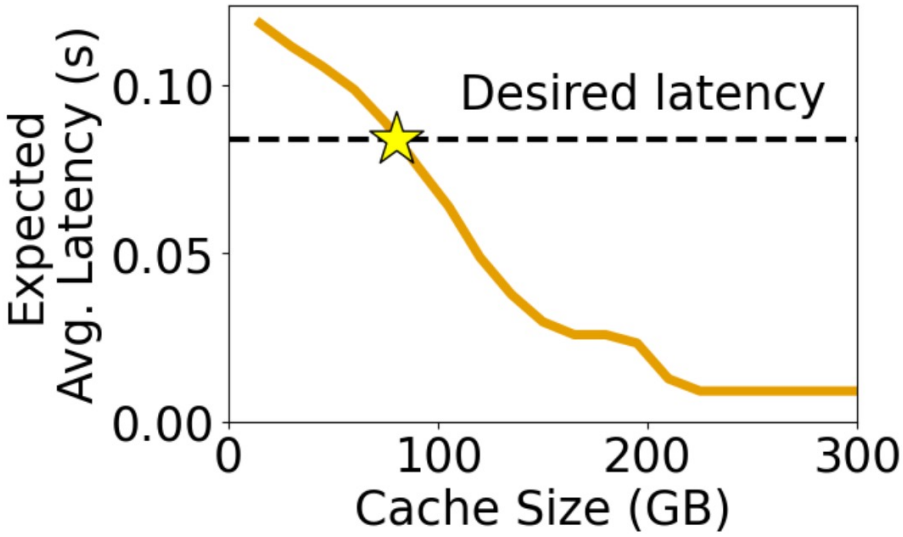
$$TotalCost(C) = EgressCost(C) + OSCCapacityCost(C + GarbageSize) + OpCost(C)$$

$$EgressPrice * BMC(C)$$

$$PutPrice * \frac{\#Writes + \#Reads * MRC(C)}{\#Objects\ per\ Packing\ Block}$$



Expected cost curve



Expected avg. latency curve

Other Details

- Policy during observation period
  - Using the first day as the observation period
  - Macaron cache all accessed data in the observation period, 37% cost reduction compared to not storing any data

- DRAM cache priming
  - When launching new cache nodes, the OSC manager scans the LRU order of cache items and preloads data into new cache nodes until they are full

Evaluation – Setup

- Trace
  - 15 IBM traces
  - 3 Uber traces
  - 1 VMware trace
- Configurations
  - Workloads are located in the N.Virginia region, remote data lake in N.California
  - Use AWS's pricing model
- Macaron simulator [1]
  - 1.5 million dollars for replaying all traces
  - Simulate latencies of remote access and object storage access

[1] https://github.com/hojinp/macaron_simulator/

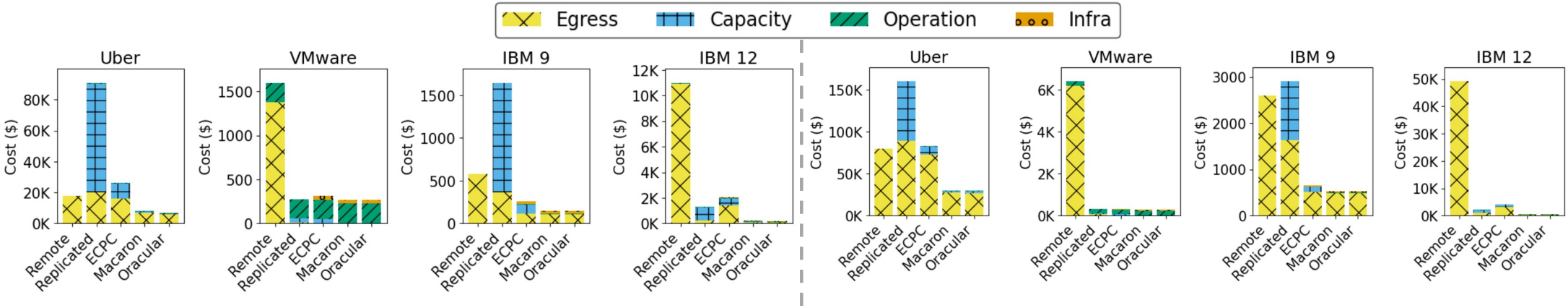
Evaluation – Setup

- Baseline
  - Remote: accessing all data from a remote cloud
  - Replicated: having all data replicated locally
  - ECPC (AWS ElastiCache): use Macaron's optimizer to efficiently auto-scale the cache
  - Oracular: ideal, unreachable

	Egress	Capacity		Operation			Infra	
		DRAM Capacity	OSC Capacity	Remote Put/Get	OSC Put	OSC Get	Serverless	OSC Manager
Remote	√	✗	✗	√	✗	✗	✗	✗
Replicated	√	✗	√	√	√	√	✗	√
ECPC	√	√	✗	√	✗	✗	√	✗
Oracular	√	√	√	√	✗	√	✗	√
Macaron	√	√	√	√	√	√	√	√

Evaluation – Cost-efficiency Analysis

- Comparison with Remote and Replicated
  - In cross-cloud, across 19 traces Macaron achieves a cost reduction of 2.4% to 99.3% (avg. 65%) and 24.9% to 82.9% (avg. 75%) compared to Remote and Replicated
  - In cross-region, for the 16 traces Macaron provides cost reductions of 28.2% to 98.5% (avg.67.4%) and 18.1% to 91.1% (avg. 78.4%) compared to Remote and Replicated

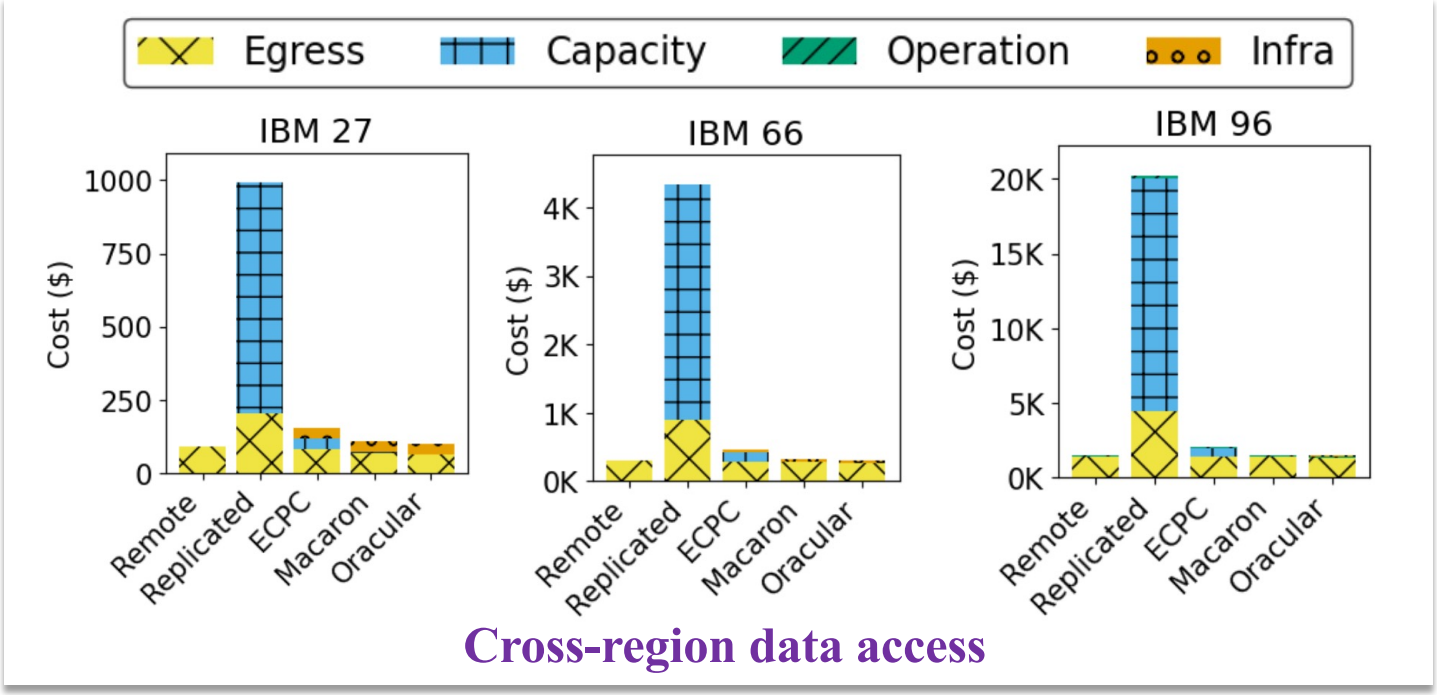


Cross-region data access

Cross-cloud data access

Evaluation – Cost-efficiency Analysis

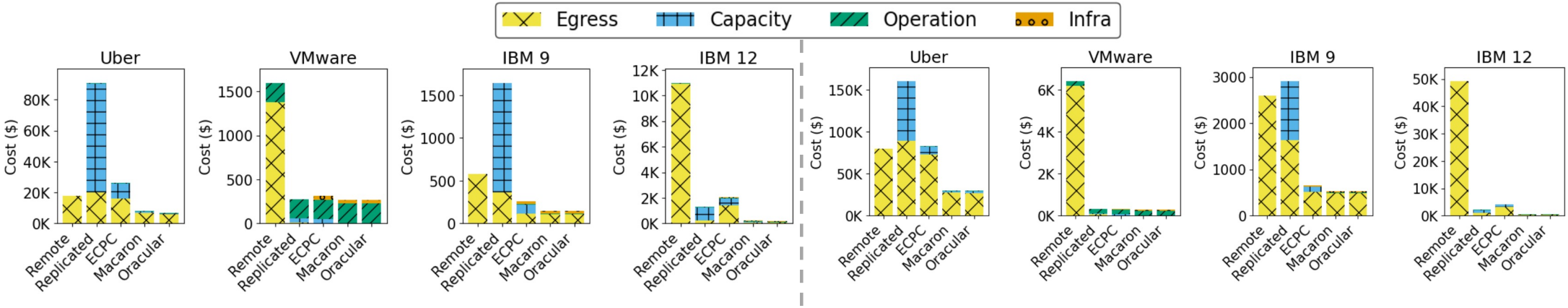
- Comparison with Remote
  - IBM 27, 66, and 96 traces have high compulsory miss ratios (57%, 79%, 87%)
  - Incur 24%, 5.8%, and 1.5% higher costs compared to Remote



Evaluation – Cost-efficiency Analysis

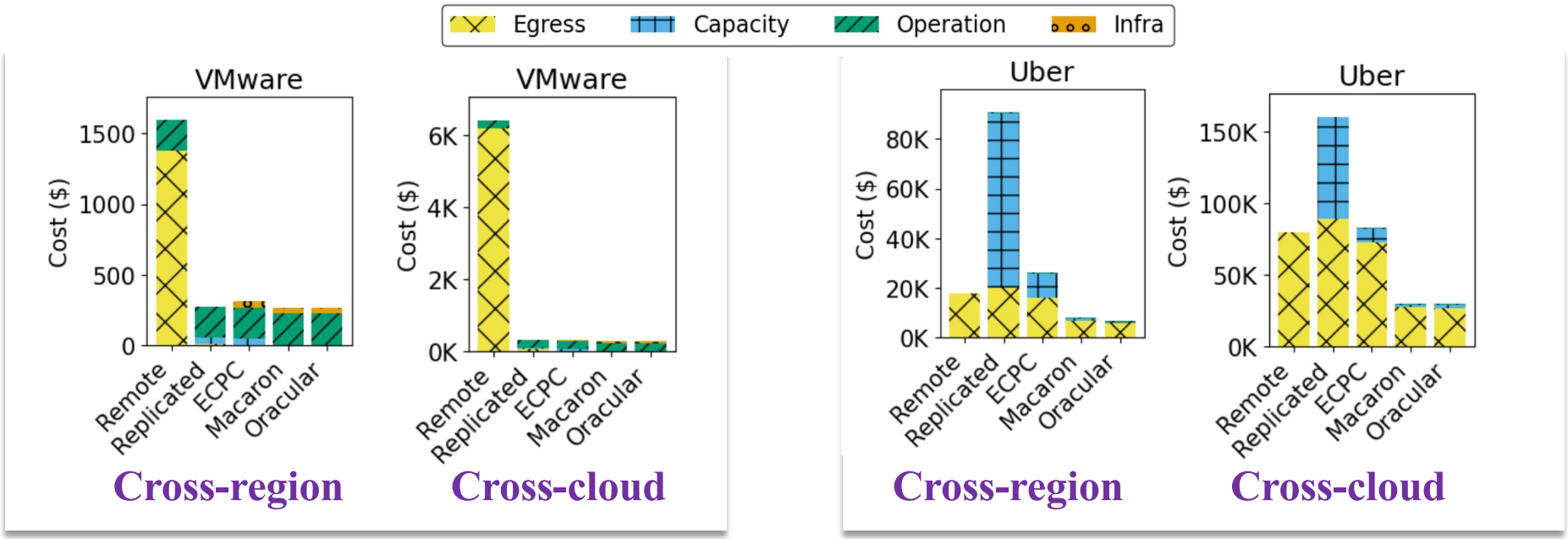
- Comparison with ECPC
  - In cross-cloud, across 19 traces Macaron reduces overall costs by 3.5–89.1% (avg. 46%)

- Comparison with Macaron
  - In cross-cloud, across 19 traces Oracular reduces overall costs by 0.4%-18.3% (avg. 6.8%)



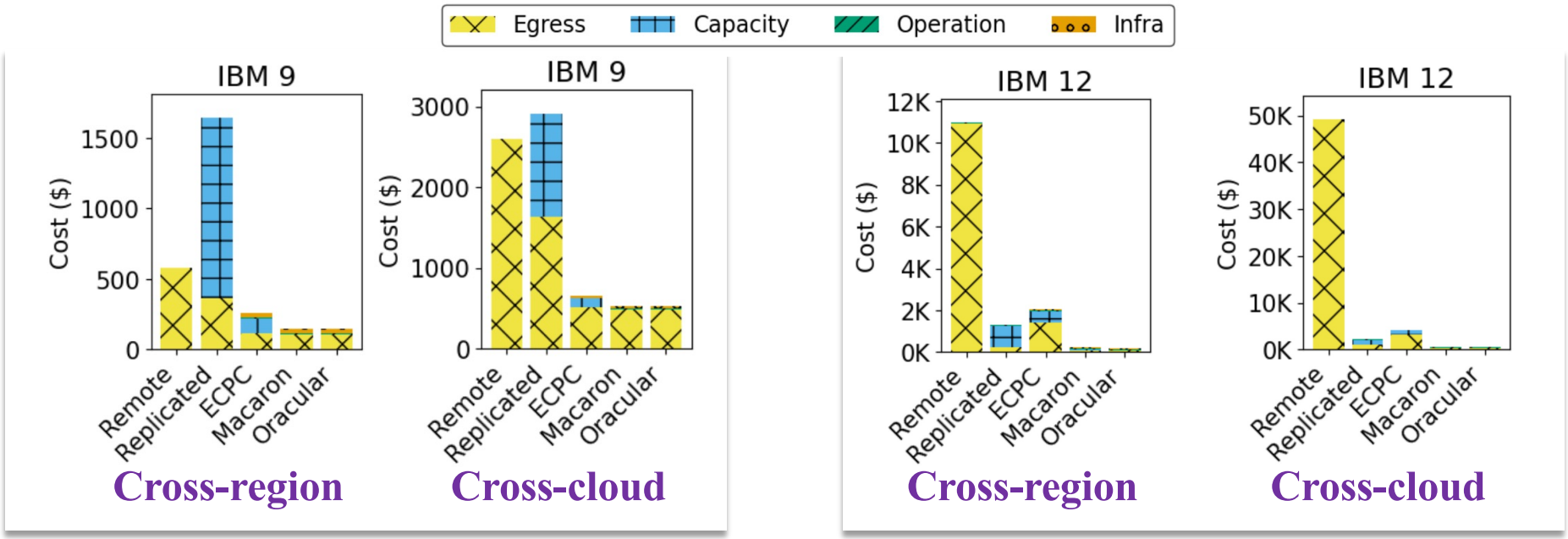
Evaluation – Cost-efficiency Analysis

- VMware trace (100% Get, small data size: 215GB)
  - Involves running numerous tests periodically, leading to a high frequency
  - 96% cost reduction compared to Remote
- Uber trace (100% Get, largest data size: 324TB)
  - Macaron achieves 81% cost reduction compared to Replicated



Evaluation – Cost-efficiency Analysis

- IBM9 trace (100% Get, 6TB)
  - A periodic burst for 15 minutes every hour
  - 79% cost reduction compared to Remote, and 82% reduction compared to Replicated
- IBM12 trace (99% Get/1%PUT, strong locality, 5 TB)
  - 98.9% reduction in data egress costs compared to Remote



Evaluation – Impact of Adaptivity Mechanisms

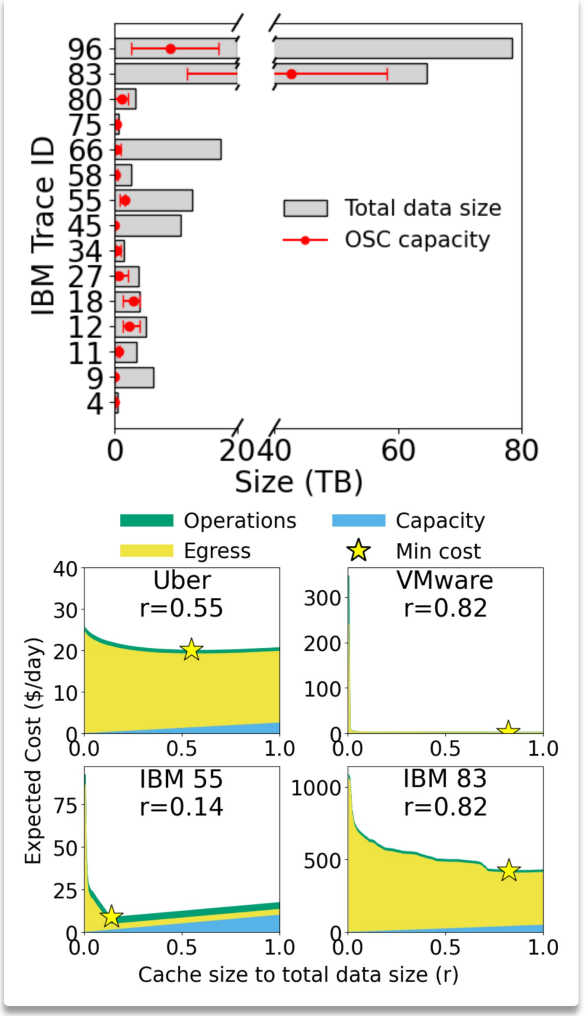
- Comparison with static Macaron
  - 0-85% (avg. 12%) cost reduction across 19traces in cross-cloud scenarios
  - 0-78% (avg. 8%) cost reduction across 19traces in cross-region scenarios

- Reconfiguration window changes (24 hours -> 15 minutes)
  - 0-41% (avg. 4%) cost reduction in cross-cloud scenarios
  - 0-25% (avg. 3%) cost reduction in cross-region scenarios

Evaluation – Effects of Macaron Optimization

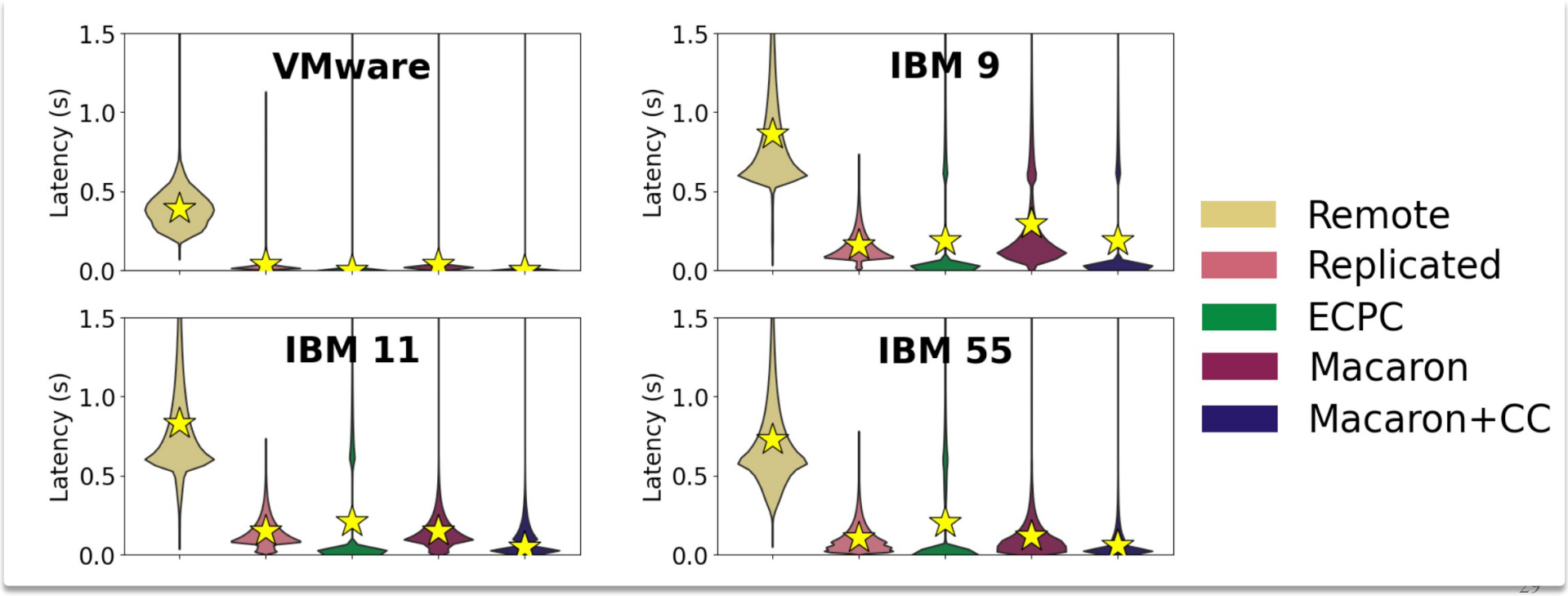
- Adaptive object storage capacity
  - The ratio of OSC capacity to data size ranges 1%~98%
  - Only one trace doesn't adjust the OSC capacity
  - Standard deviation of ratio ranges 0~0.28 (avg. 0.1)

- Object packing
  - 36% cost reduction in IBM 18 trace (small object)
  - 5% cost reduction in IBM 45 trace



Evaluation – Macaron with low latency

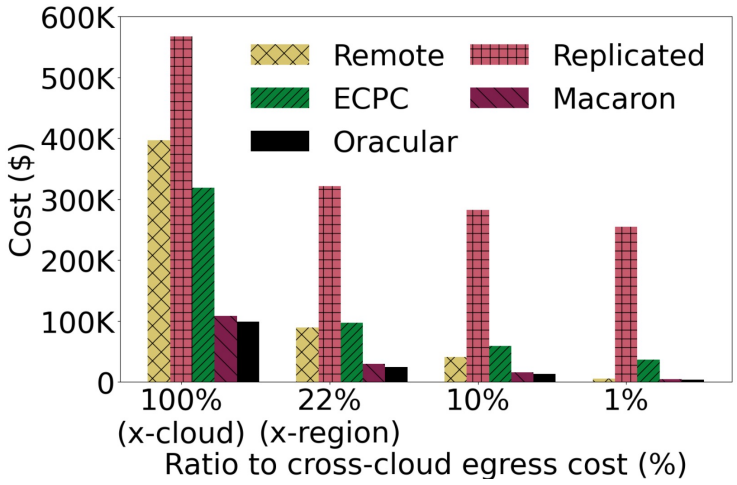
- In VMware, IBM 9, 11, 55 traces, compared to ECPC
  - Macaron+CC demonstrates cost savings of 0.3%, 9%, 37%, and 16%
  - Reduce latency by 3%, 1%, 76%, and 70%

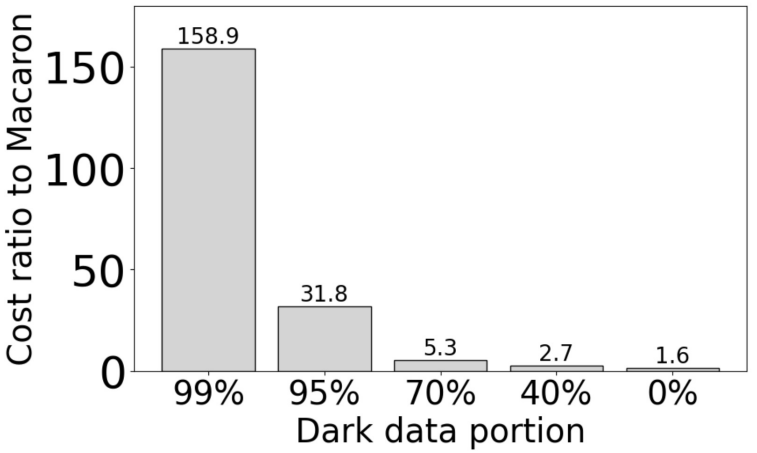


Evaluation – Sensitivity analysis

- Different egress cost
  - 100%: 9¢/GB

- Different dark data percentage
  - At 0% portion, Macaron is 37.5% cheaper than Replicated.
  - At 99% dark data, Replicated is 158.9× costlier than Macaron





Conclusion

- Macaron addresses the high data transfer and latency costs associated with cross-cloud or cross-region data access
- Macaron dynamically auto-configures the size and storage types of cache space for remote data, reducing cross-cloud dollar cost by 65% for a collection of real workload traces