
Where Should Compaction Run?

Compaction Worker Selection on Disaggregated LSM-KVS

- Suhwan Shin, Jaedong Lee, Seehwan Yoo, Jongmoo Choi (Dankook University)

Presentation by Suhwan Shin

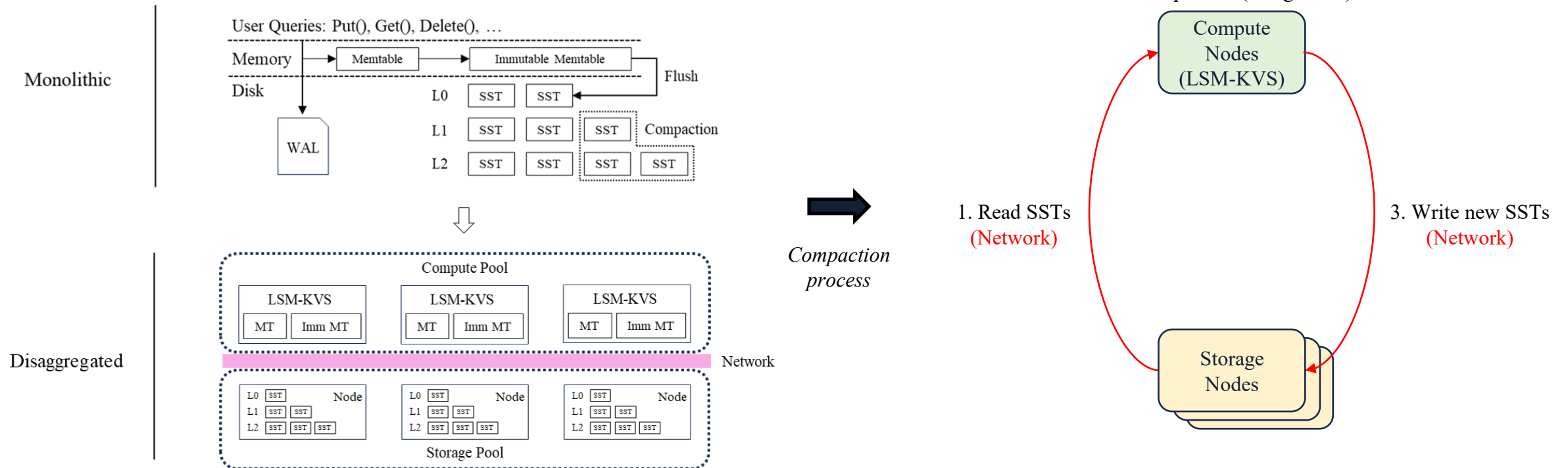
sshshin@dankook.ac.kr

2026.09.29

BACKGROUND

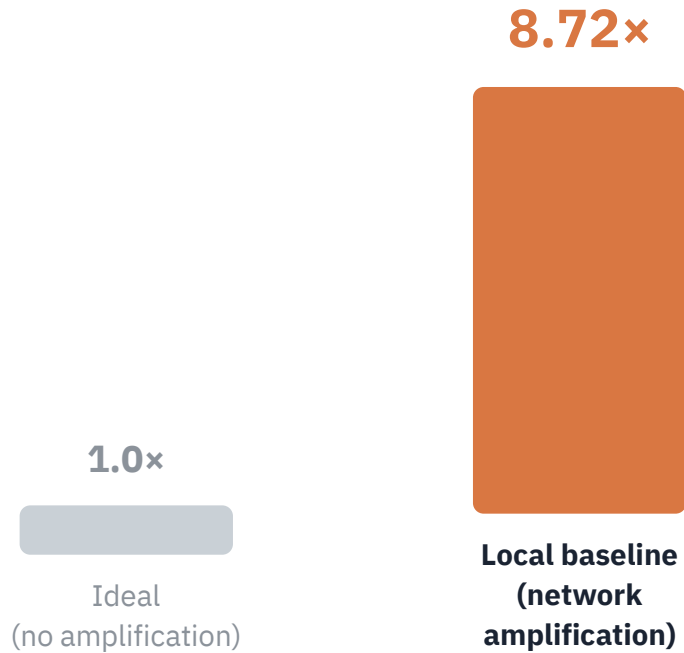
Disaggregated Storage & Compaction

- Compute and storage separated (e.g., RocksDB on Tectonic)
- Compaction merges and reorganizes LSM-tree SST files
- Every input SST already sits on remote storage



PROBLEM

How large is the network cost?



On an 8-node Ceph cluster running the standard Local baseline, every byte of user writes generates 8.72× that much network traffic

$$NA \approx WA(\rho + 1) + 1$$

(NA = network amplification, WA = write amplification, ρ = replication factor)

PRIOR WORK

Nobody looks at where the data is

System	Execution	Worker Selection
RocksDB-Cloud	Cloud	Round-robin
CaaS-LSM	Stateless Service	Least-loaded
Disagg. RocksDB	Shared store	Load-based
Hailstorm	Idle compute	Least-loaded
LARC (ours)	Storage	Locality-aware

None of them ask whether a worker already has the data

What We Asked?

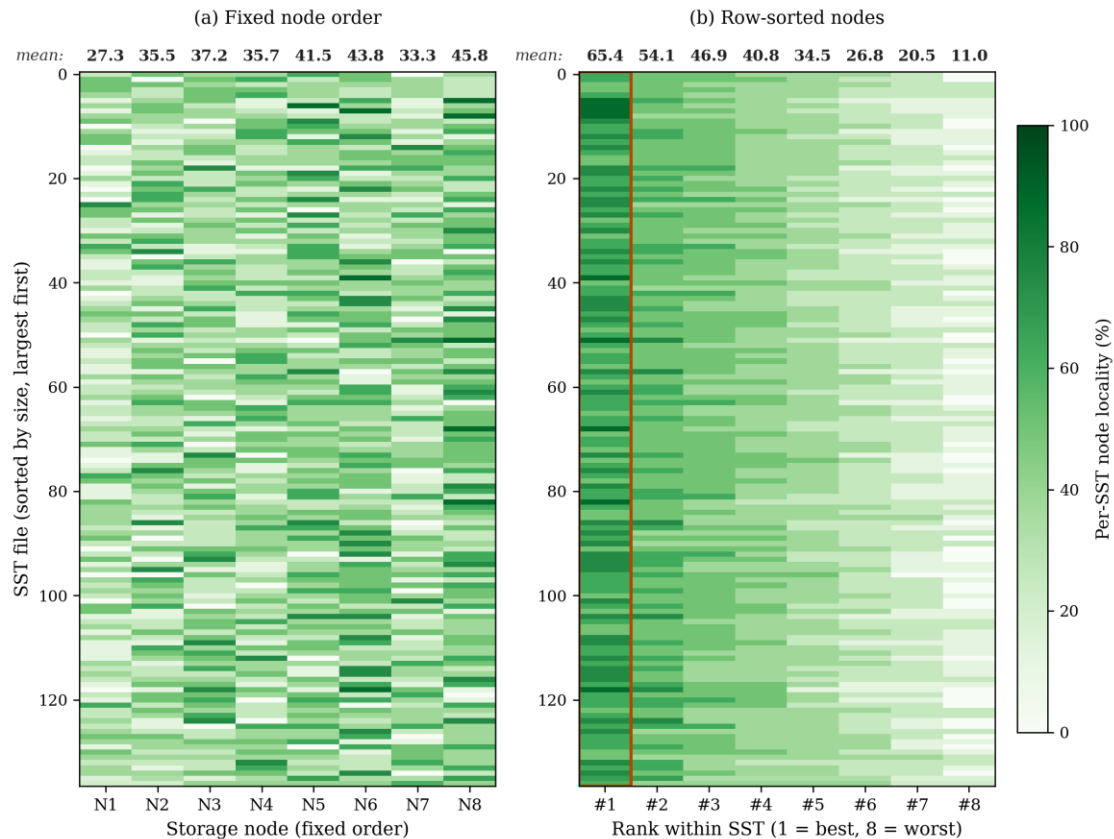
"Does physical data placement create a locality opportunity at the per-compaction level and can it be exploited for worker selection?"

MOTIVATION

SST-Level Locality

8-node, 32-OSD Ceph · 100M-key fillrandom → Results 137 SSTs, 1,091 unique objects

SST Locality Heatmap (137 SSTs × 8 nodes, 1,091 unique objects)



65.4%

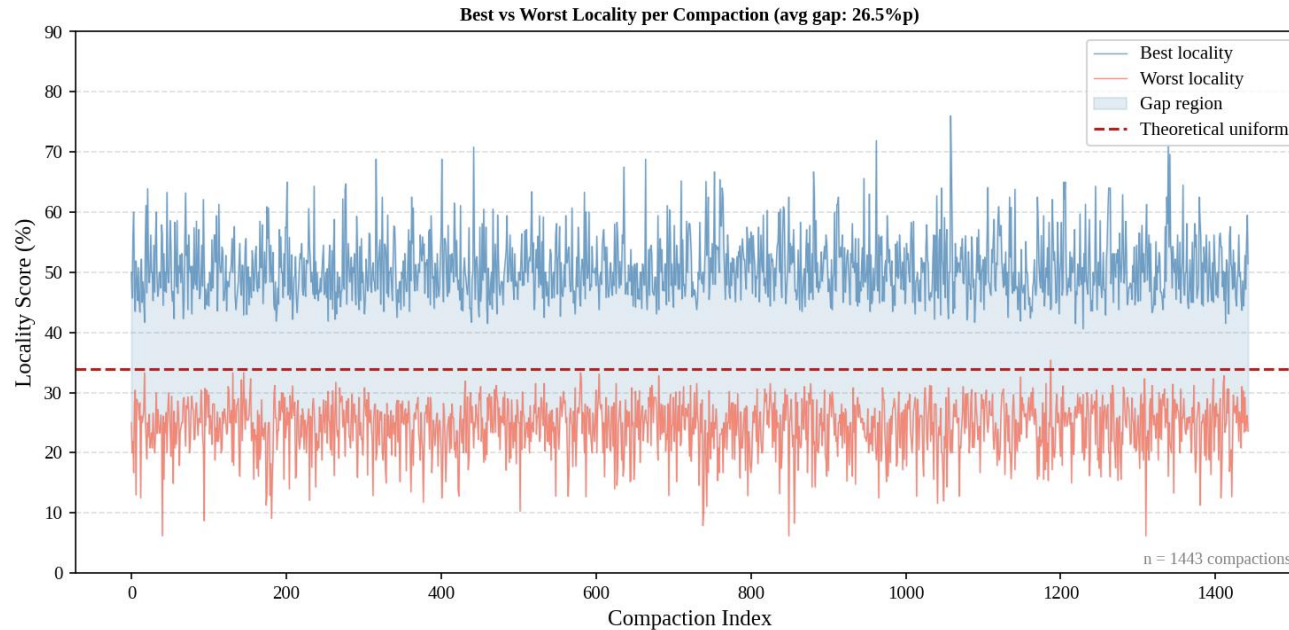
avg. share held by the top-ranked node

11.0%

avg. share held by the bottom-ranked node

MOTIVATION

Compaction-Level Locality



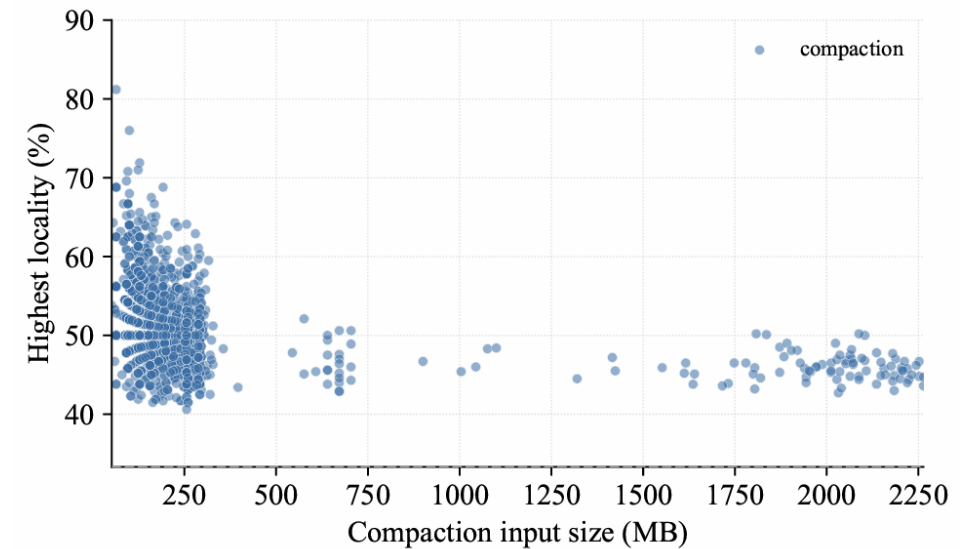
51.1%

best node

24.0%

worst node

This gap is structural, not incidental — it recurs at every compaction



The gap holds regardless of compaction size

The Locality Score

Locality Score = the share of a compaction's input that a worker already stores locally

$$L(W) = |\{d \in D : OSD(d) \cap OSD(W) \neq \emptyset\}| / |D|$$

$$W^* = \operatorname{argmax}_W L(W)$$

D = the set of input objects for this compaction

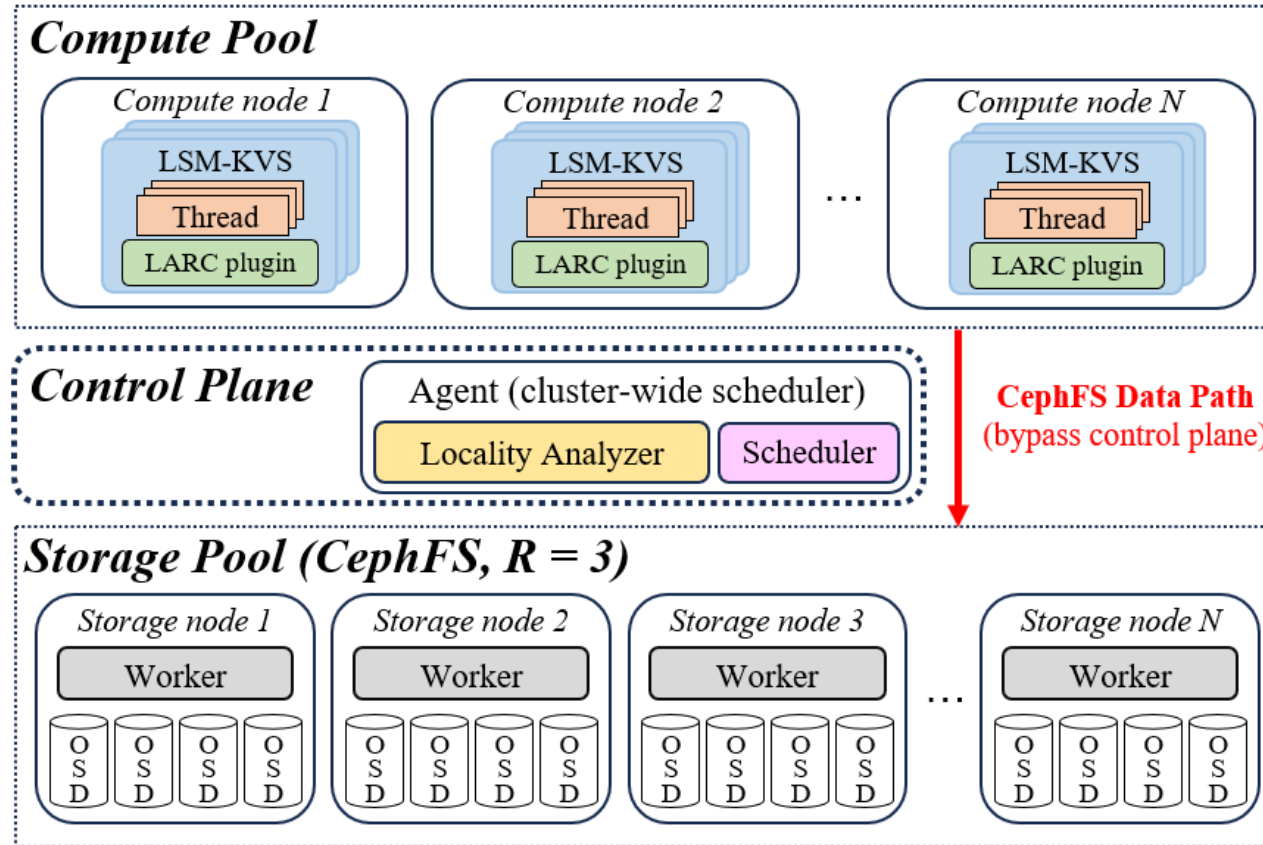
d = an individual object in D

W = a candidate worker (storage node)

OSD(x) = the set of OSDs storing x

L(W) = worker W's locality score

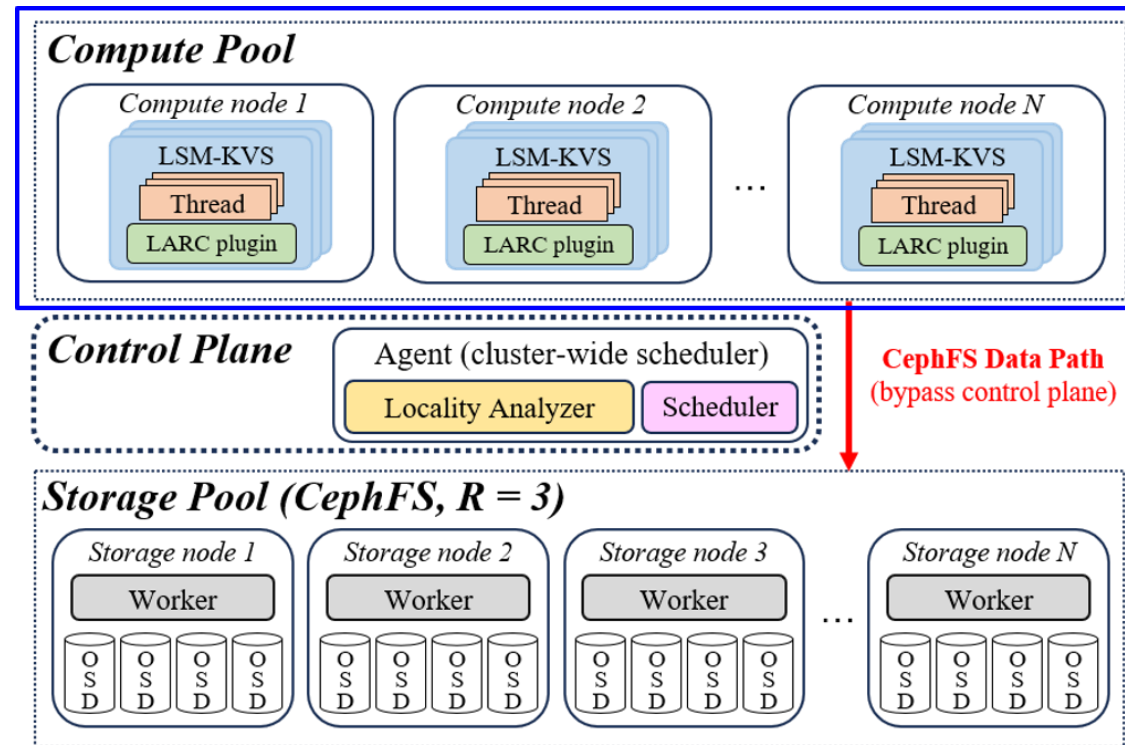
Architecture



1. Plugin → 2. Agent → 3. Worker

Architecture

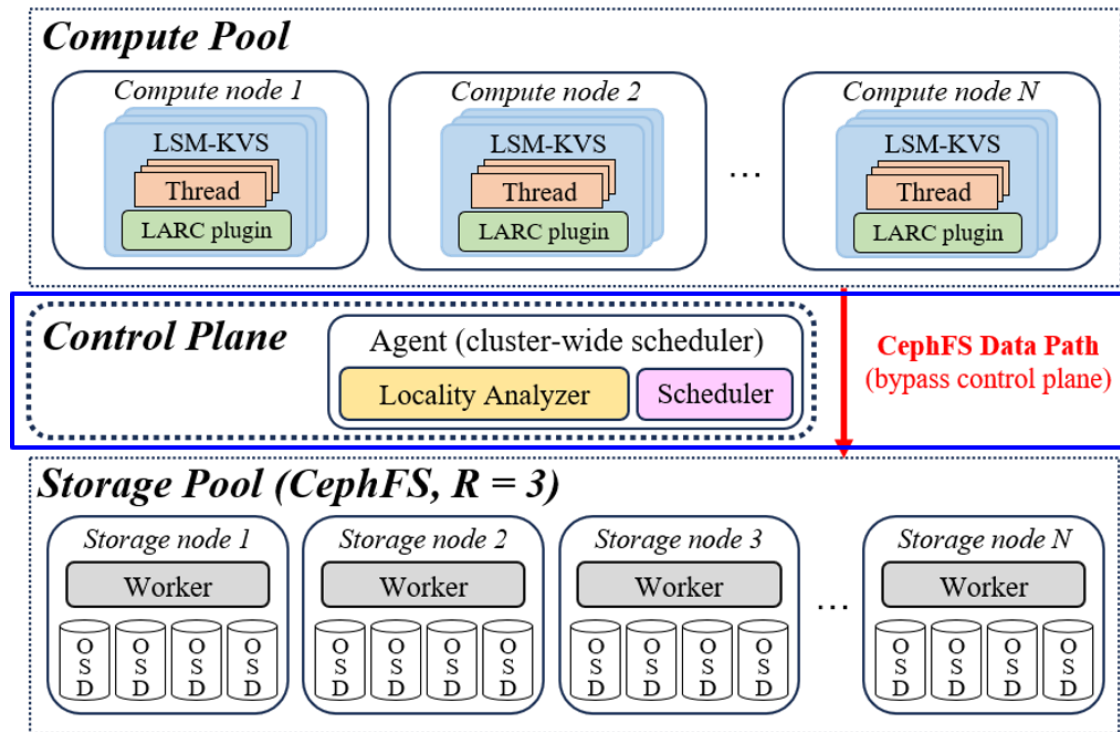
- On the compute node, a lightweight plugin hooks into CompactionService interface, detects a compaction, and forwards it to the Agent on the control plane over gRPC



1. Plugin → 2. Agent → 3. Worker

Architecture

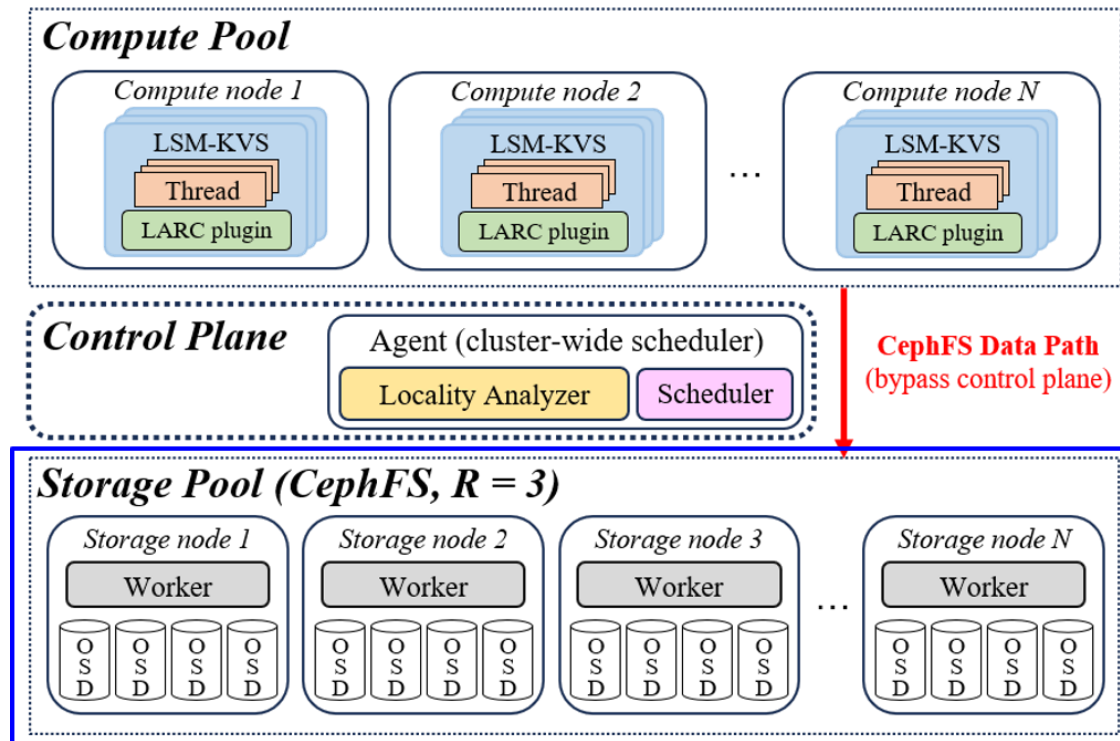
- Inside the Agent, a Locality Analyzer computes the placement info and a Scheduler picks the best worker



1. Plugin → 2. Agent → 3. Worker

Architecture

- That worker then reads objects it already has locally straight from disk and fetches only what's missing over the network



1. Plugin → 2. Agent → 3. Worker

Experimental Setup

Cluster	8-node Ceph, 32 OSDs, replication factor $p=3$
Implementation	RocksDB v9.10.0 + LARC (Plugin / Agent / Worker)
Workloads	fillrandom (write-only) · mixgraph (read ratio 0–100%)
Dataset	100M-key
Baseline	Local vs LARC
Metrics	Throughput, Network Amplification (NA), Total Network Traffic, Locality Score

Throughput & Network Amplification

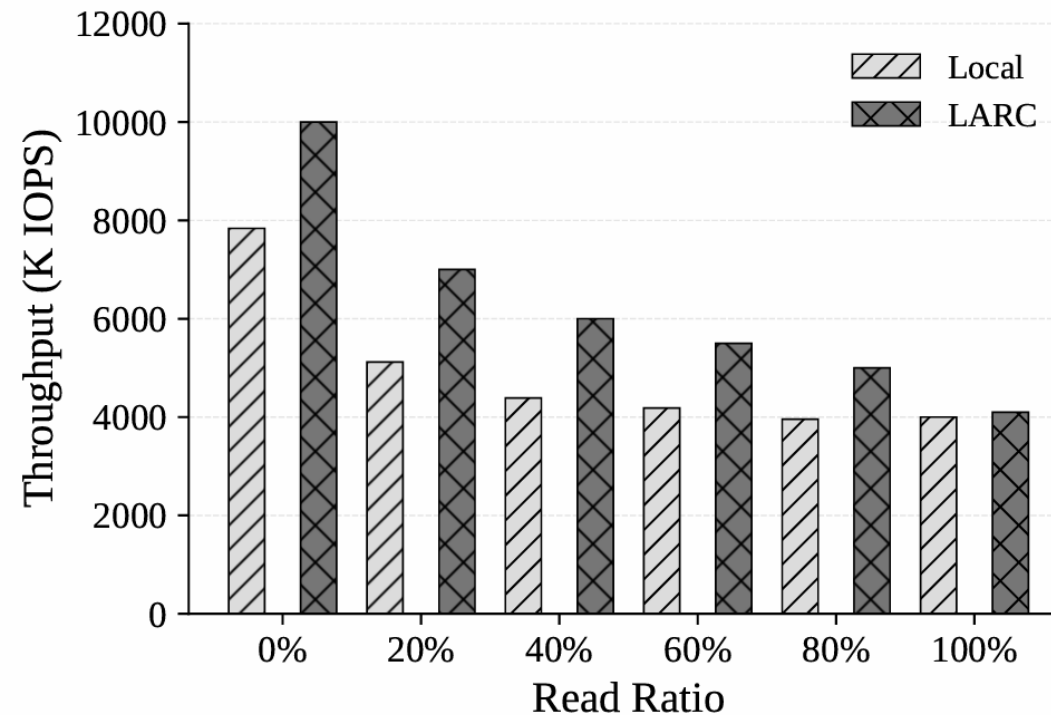
8-node Ceph · RocksDB v9.10.0 · 100M-key fillrandom

Metric	Local	LARC
Avg. locality	—	51.1%
Network amplification	8.72×	5.86×
Throughput	baseline	+16.1%

30.9 GB of cluster-wide traffic saved in a single 100M-key run

Realistic Workload (mixgraph)

Same 100M-key dataset, production-representative mixgraph workload · read ratio varied 0-100%



+27.6% throughput at 100% write (largest gain)

LARC (Locality-Aware Remote Compaction)

Locality is a structural signal — LARC puts it to work

- Physical data placement creates a persistent, exploitable locality signal per compaction
- LARC exploits it via **client-side scoring** from a cached CRUSH map - no extra metadata lookups
- Locality 51.1% · NA $8.72\times \rightarrow 5.86\times$ · Throughput +27.6%, with zero changes to the storage layer

Future Work

- *Extending locality-awareness to output placement as clusters scale, and toward multi-parameter scheduling that jointly considers load and bandwidth*

Thank you

Any Question?