Curator: Self-Managing Storage for Enterprise Clusters

I. Cano*, S. Aiyar, V. Arora, M. Bhattacharyya, A. Chaganti, C. Cheah, B. Chun, K. Gupta, V. Khot and A. Krishnamurthy*

Nutanix Inc.

***University of Washington**

NSDI '17

Enterprises have been increasingly relying on cluster storage systems

Enterprises have been increasingly relying on cluster storage systems

- Transparent access to storage
- Scale up storage by buying more resources

Cluster storage systems embody significant functionality to support the needs of enterprise clusters

- Automatic replication and recovery
- Seamless integration of SSDs and HDDs
- Snapshotting and reclamation of unnecessary data
- Space-saving transformations
- …

Much of their functionality can be performed in the background to keep the I/O path as simple as possible

What is the appropriate systems support for engineering these background tasks?

Curator

Framework and systems support for building background tasks

- Extensible, flexible and scalable framework
  - Borrow ideas from big data analytics but use them ***inside*** of a storage system
- Synchronization between background tasks and foreground I/O
  - ***Co-design*** background and foreground components
- Heterogeneity across and within clusters over time
  - Use ***Reinforcement Learning*** to deal with heterogeneity

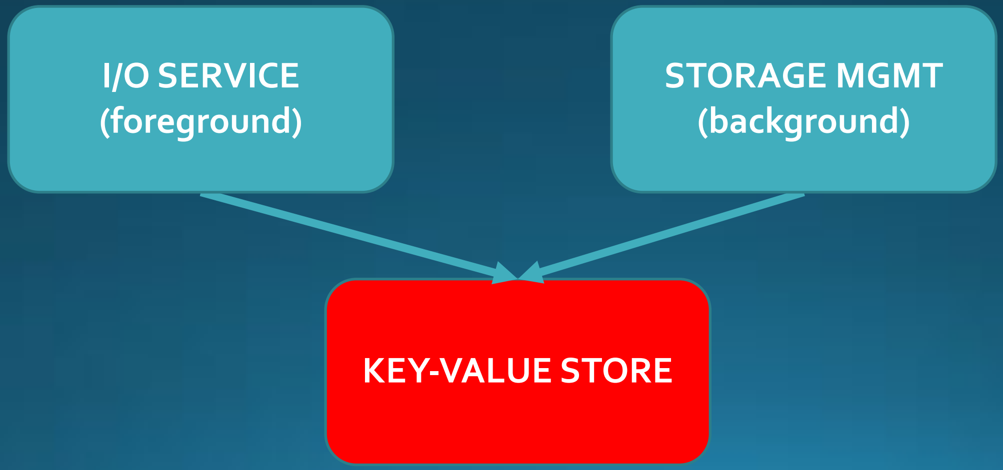
Curator

Framework and systems support for building background tasks

- **Extensible, flexible and scalable framework**
  - Borrow ideas from big data analytics but use them ***inside*** of a storage system
- Synchronization between background tasks and foreground I/O
  - ***Co-design*** background and foreground components
- Heterogeneity across and within clusters over time
  - Use ***Reinforcement Learning*** to deal with heterogeneity

Distributed Key-Value Store

- Metadata for the entire storage system stored in k-v store
- Foreground I/O and background tasks coordinate using the k-v store
- Key-value store supports replication and consistency using Paxos



Data Structures and Metadata Maps

- Data stored in units called ***extents***
- Extents are grouped together and stored as ***extent groups*** on physical devices

Extent Id Map

eid	egid
33	1984
44	1984
56	1984

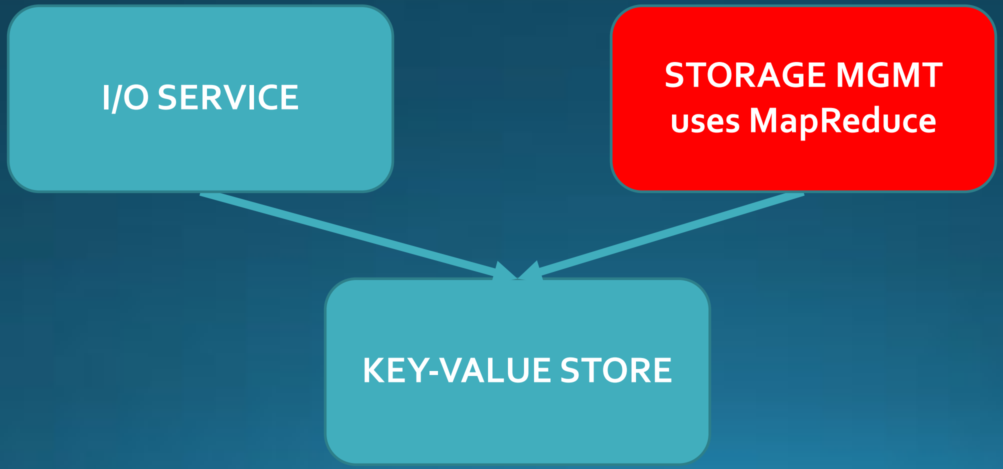
Extent Group Id Map

egid	physical location
1984	disk1,disk2
...	...

- Multiple levels of redirection simplifies data sharing across files and helps with minimizing map updates

MapReduce Framework

- Globally distributed maps processed using MapReduce
- System-wide knowledge of metadata used to perform various self-managing tasks

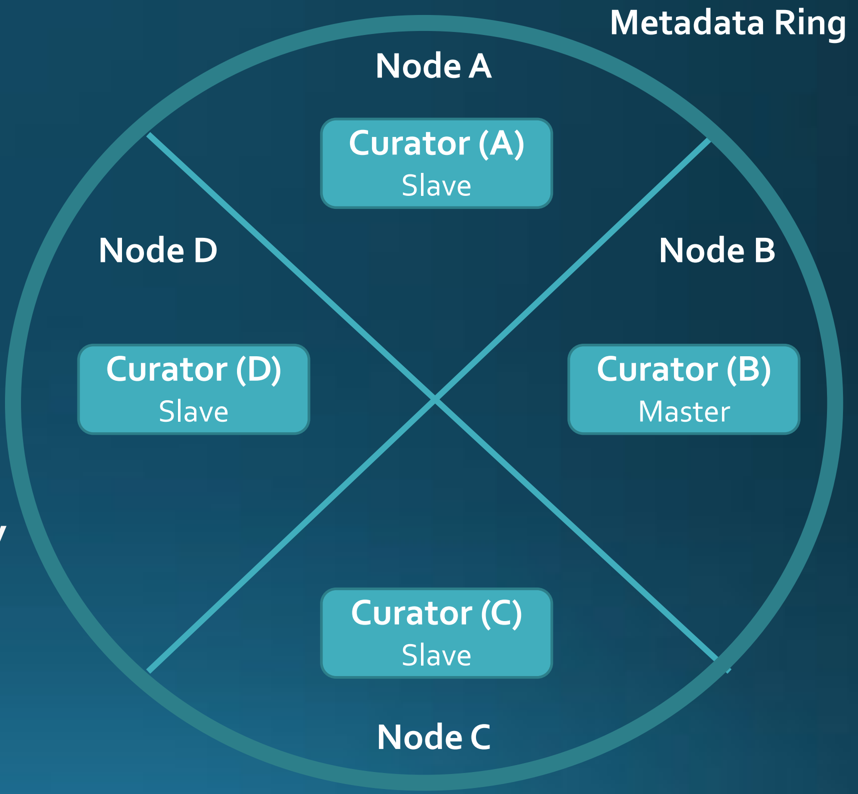


Example: Tiering

- Move cold data from fast (SSD) to slow storage (HDD, Cloud)
- Identify cold data using a MapReduce job
  - Modified Time (mtime): Extent Group Id map
  - Access Time (atime): Extent Group Id Access Map

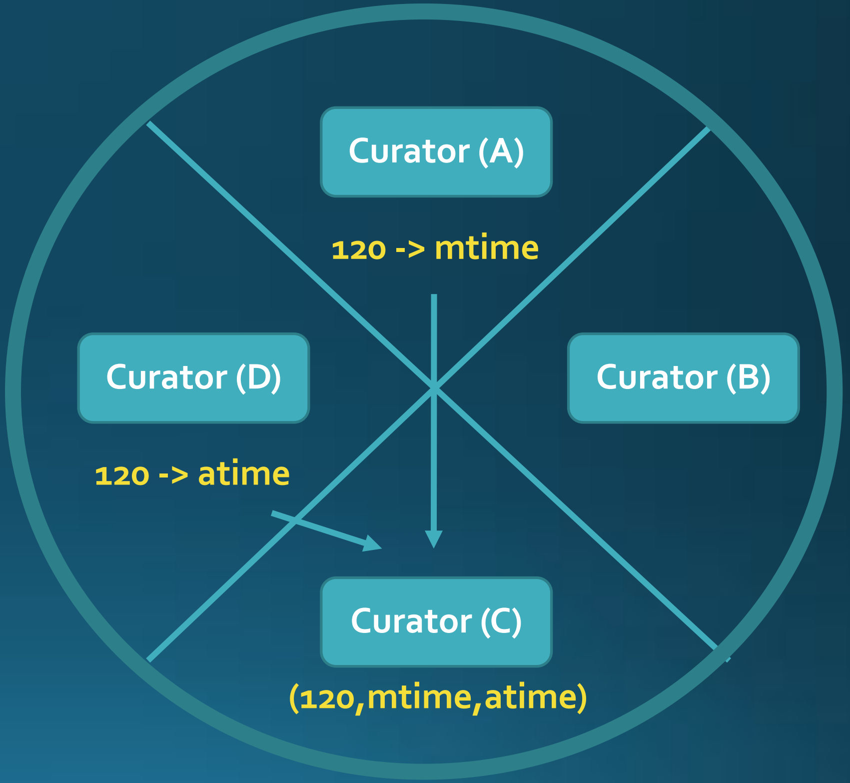
Example: Tiering

- egid 120
  - mtime owned by Node A
  - atime owned by Node D
- egid 120 == “cold” ?
  - Maps globally distributed
    → not a local decision
- Use MapReduce to perform a “join”



Example: Tiering

- Map phase
  - Scan both metadata maps
  - Emit egid -> mtime or atime
  - Partition using egid
- Reduce phase
  - Reduce based on egid
    - Generate tuples (egid,mtime,atime)
  - Sort locally and identify the cold egroups



Other tasks implemented using Curator

- Disk Failure
- Disk Balancing
- Fault Tolerance
- Garbage Collection
- Data Removal
- Compression
- Erasure Coding
- Deduplication
- Snapshot Tree Reduction

Challenges

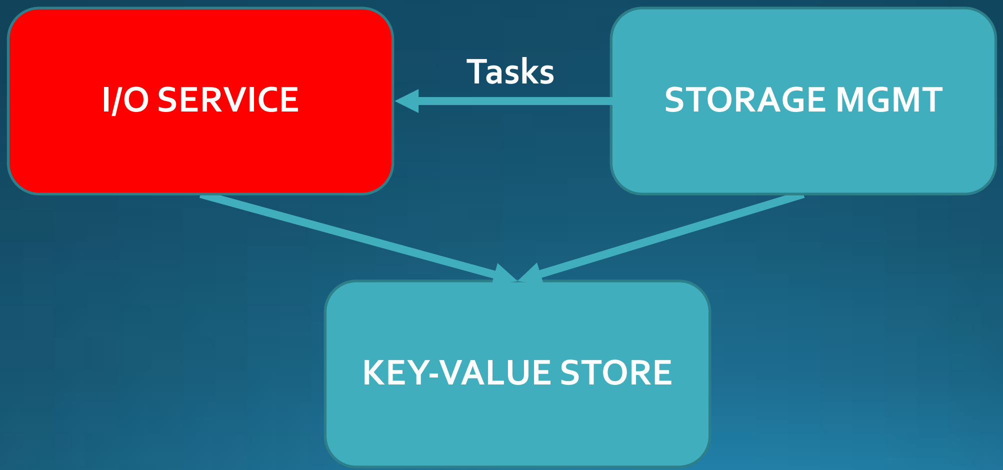
- Extensible, flexible and scalable framework ✓
- Synchronization between background tasks and foreground I/O
- Heterogeneity across and within clusters over time

Challenges

- Extensible, flexible and scalable framework ✓
- **Synchronization between background tasks and foreground I/O**
- Heterogeneity across and within clusters over time

Co-Design Background Tasks and Foreground I/O

- I/O Service provides an extended *low-level* API
- Storage Mgmt. only gives hints to I/O Service to act on the data
- Background tasks are batched and throttled
- Soft updates on metadata to achieve lock-free execution

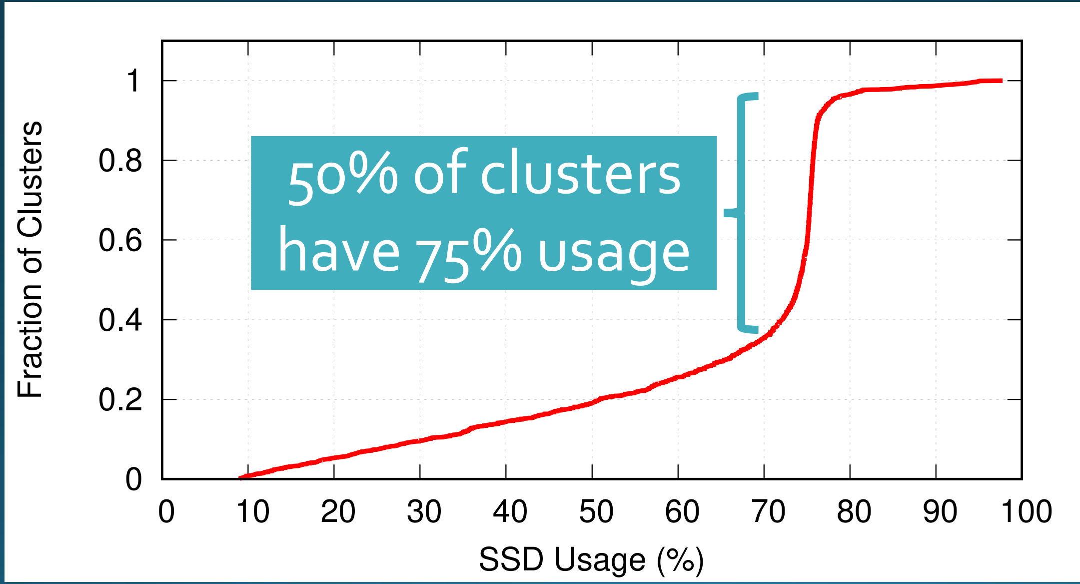


Evaluation

- Clusters in-the-wild
  - ~ 50 clusters over a period of 2.5 months
- Key results
  - Recovery
    - 95% of clusters have at most 0.1% of under replicated data
  - Garbage
    - 90% of clusters have at most 2% of garbage
  - More results in paper

Tiering

- Goal: maximize SSD effectiveness for both reads and writes
- Curator uses a threshold to achieve some "desired" SSD occupancy



Challenges

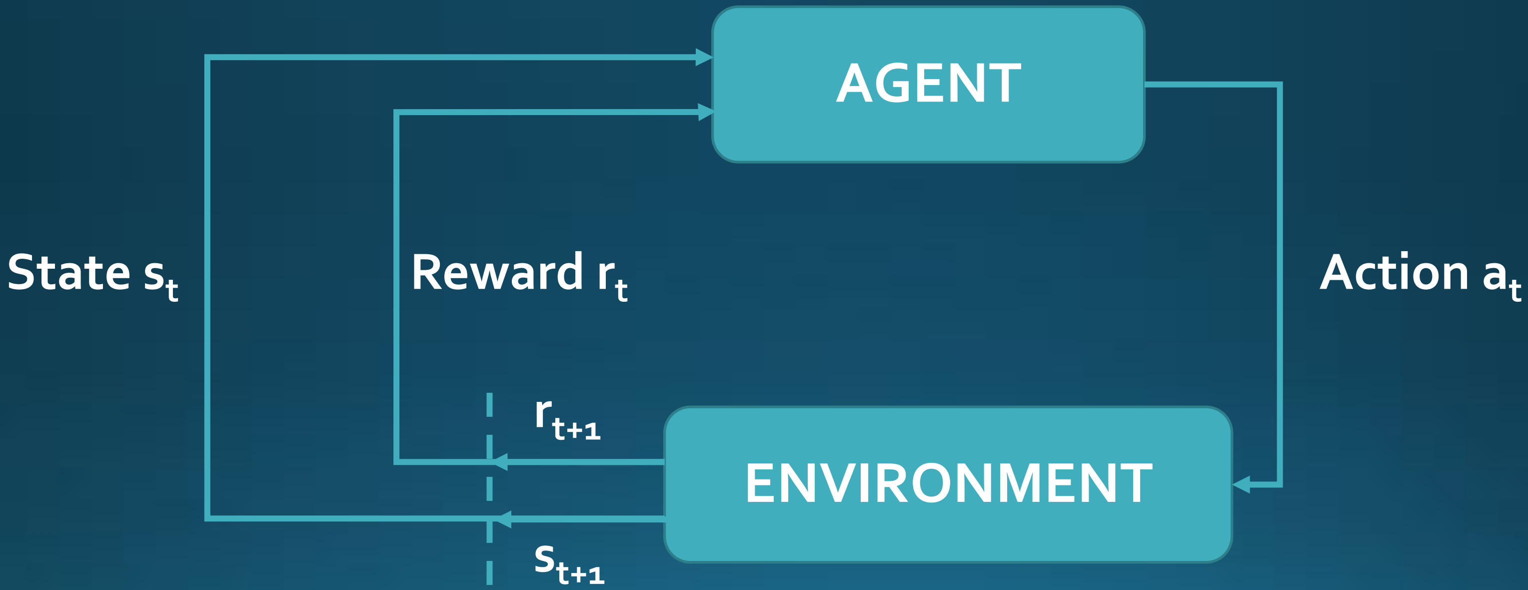
- Extensible, flexible and scalable framework ✓
- Synchronization between background tasks and foreground I/O ✓
- **Heterogeneity across and within clusters over time**

Machine Learning - Reinforcement Learning

- Heterogeneity across cluster deployments
  - Resources
  - Workloads
- Dynamic changes within individual clusters
  - Changing workloads
  - Different loads over time

- Threshold-based heuristics sub-optimal
- ML to the rescue!
  - Reinforcement Learning

Reinforcement Learning 101



Agent Deployment

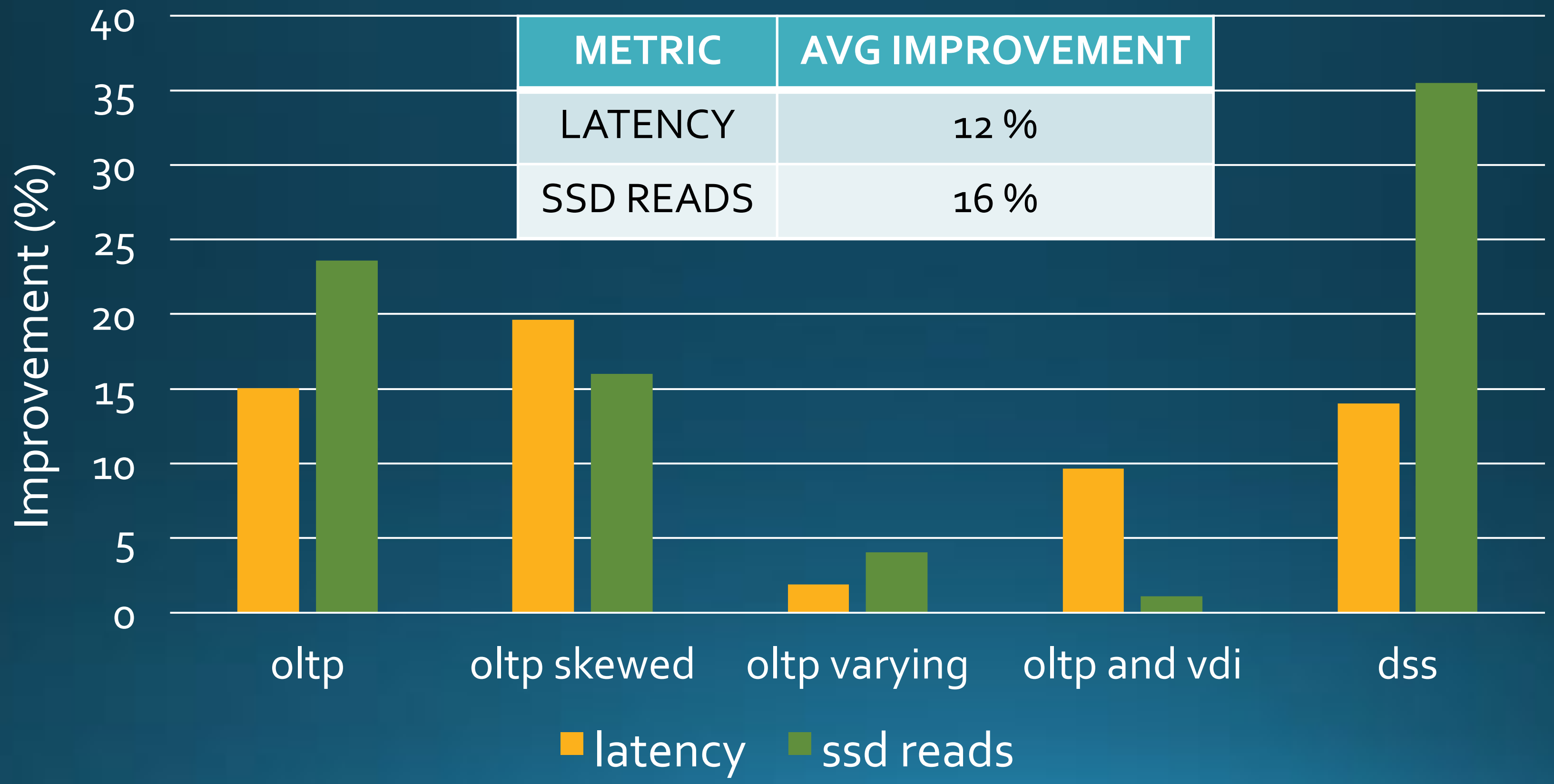
- How the agent is deployed?
  - No prior knowledge and needs to learn from scratch
  - Some prior knowledge and then keeps on learning while deployed

Use Case: Tiering

- State = (cpu, memory, ssd, iops, riops, wiops)
- Actions = { run, not run }
- Reward = - latency

- Leverage threshold-based heuristics to “bootstrap” our agent
- Build a dataset from real traces
  - ~ 40 clusters in-the-wild
  - ~ 32K examples
  - State-Action-Reward tuples
- Pre-train two models, one for each action

Evaluation



METRIC	AVG IMPROVEMENT
LATENCY	12 %
SSD READS	16 %

Conclusions

***Curator*: framework and systems support for building background tasks**

- Borrowed ideas from big data analytics but used them ***inside*** of a storage system
  - Distributed key-value store for metadata
  - MapReduce framework to process metadata
- ***Co-designed*** background tasks and foreground I/O
  - I/O Service provides an extended low-level API
  - Storage Mgmt. only gives hints to I/O Service to act on actual data
- Used ***Reinforcement Learning*** to deal with heterogeneity across and within clusters over time
  - Results on Tiering showed up to ~20% latency improvements