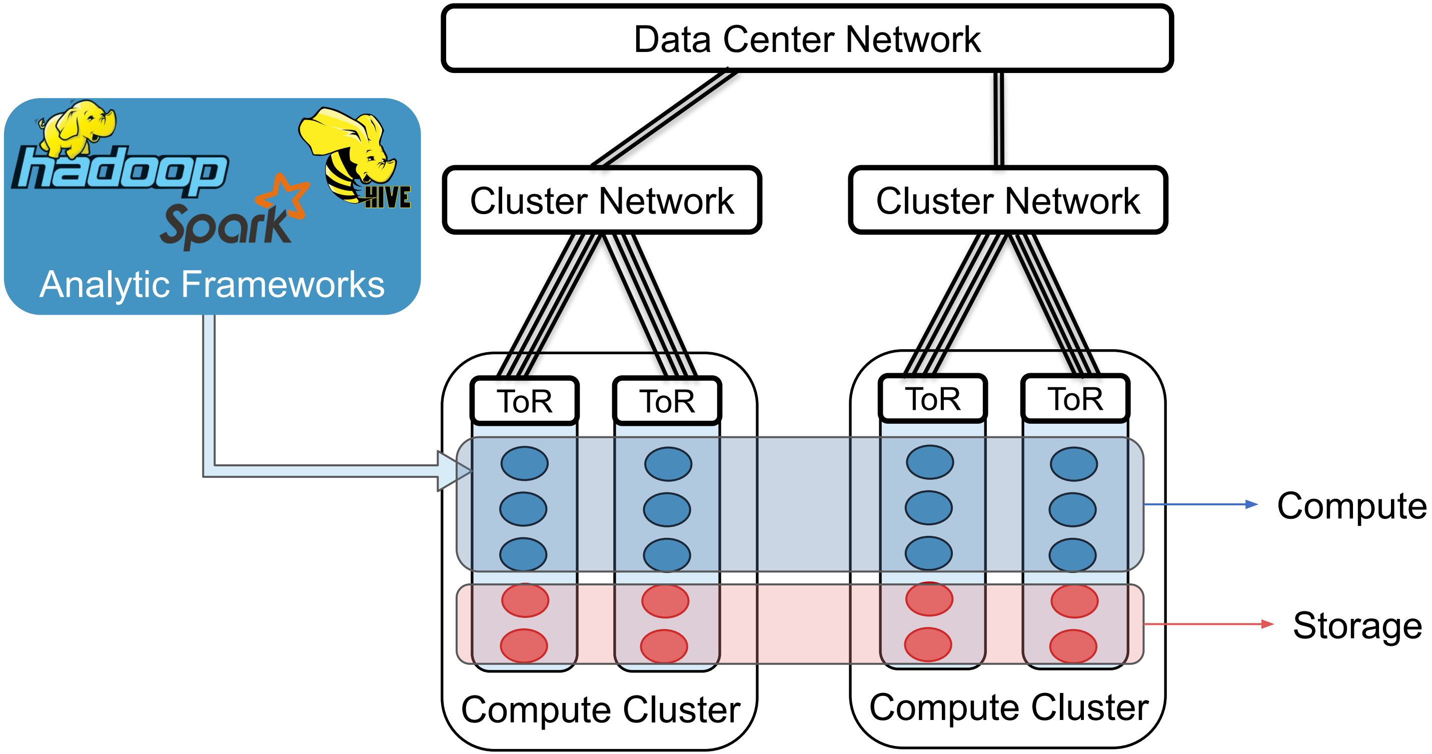
D3N
A multi-layer cache for the rest of us

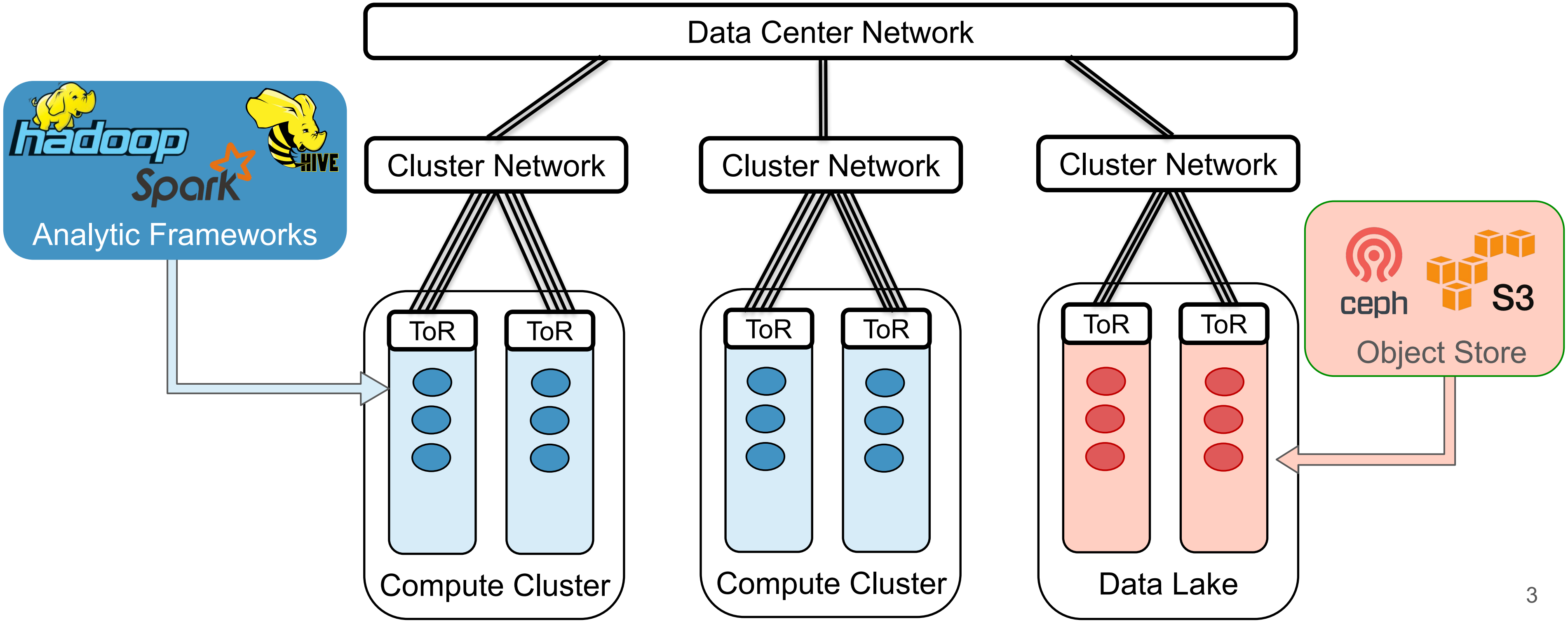
E. Ugur Kaynar, Mania Abdi, Mohammad Hossein Hajkazemi,
Ata Turk, Raja Sambasivan, David Cohen,
Larry Rudolph, Peter Desnoyers, Orran Krieger

Motivation



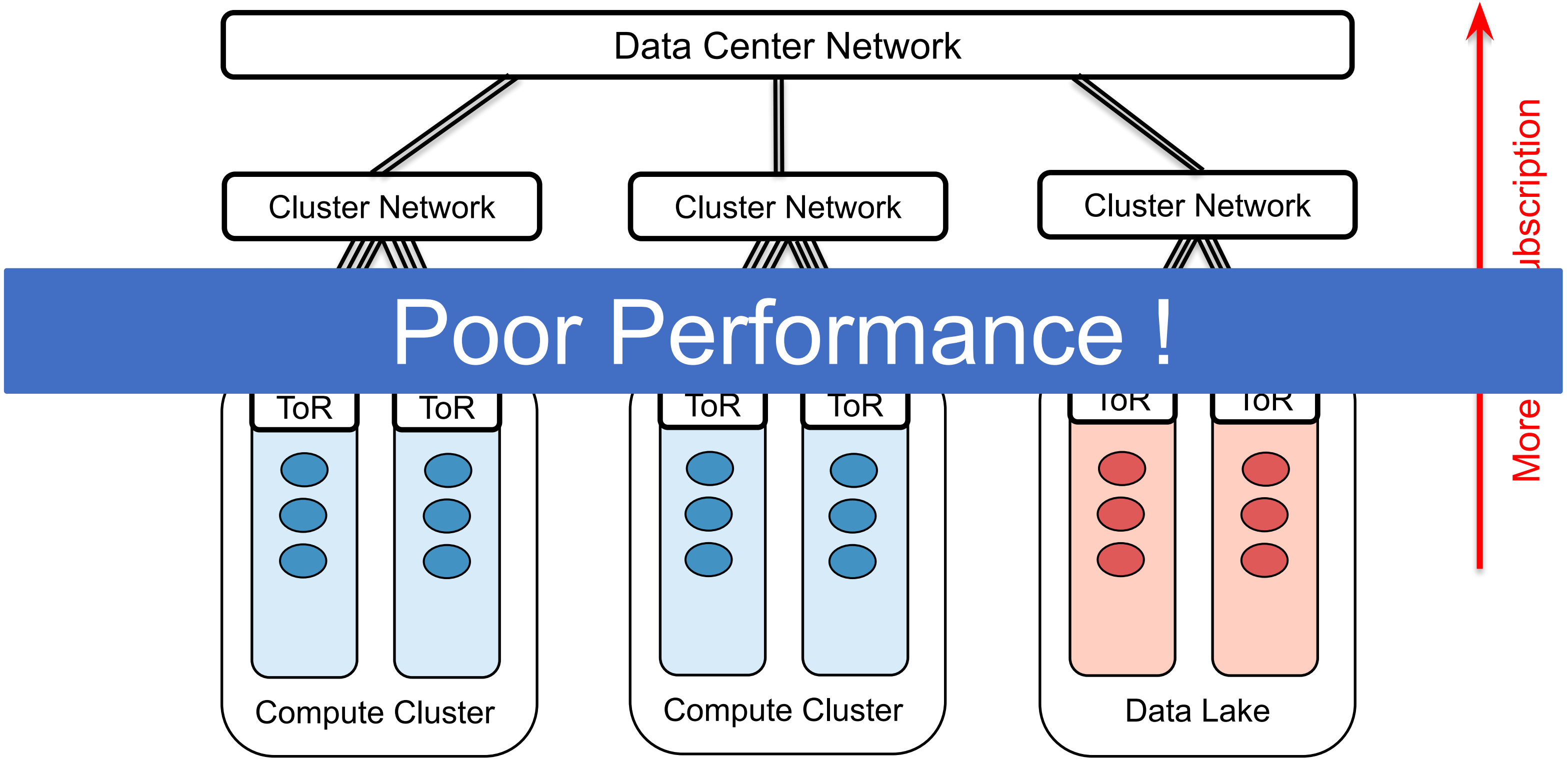
Data Center Network
Cluster Network
Cluster Network
hadoop
HIVE
Spark
Analytic Frameworks
ToR
ToR
ToR
ToR
Compute Cluster
Compute Cluster
Compute
Storage

Motivation



Data Center Network
Cluster Network
Cluster Network
Cluster Network
hadoop
Spark
HIVE
Analytic Frameworks
ToR
ToR
ToR
ToR
ToR
ToR
Compute Cluster
Compute Cluster
Data Lake
ceph
S3
Object Store

Network Limitations in Data Center



Caching for Big Data Analytics

Two Sigma [2018], Facebook [VLDB 2012], and **Yahoo [2010]** analytic cluster traces show that;

- High data input reuse
- Uneven Data Popularity
- File popularity changes over time
- Datasets accessed repeatedly by the same analytic clusters and between different analytic clusters

CACHING

Alluxio (formerly known as Tachyon[SOCC'14]), **HDFS-Cache**, **Pacman**[NSDI'12], **Adaptive Caching** [SOCC'16], **Scarlett**[Eurosys'11] , **Netco**[SOCC'19]

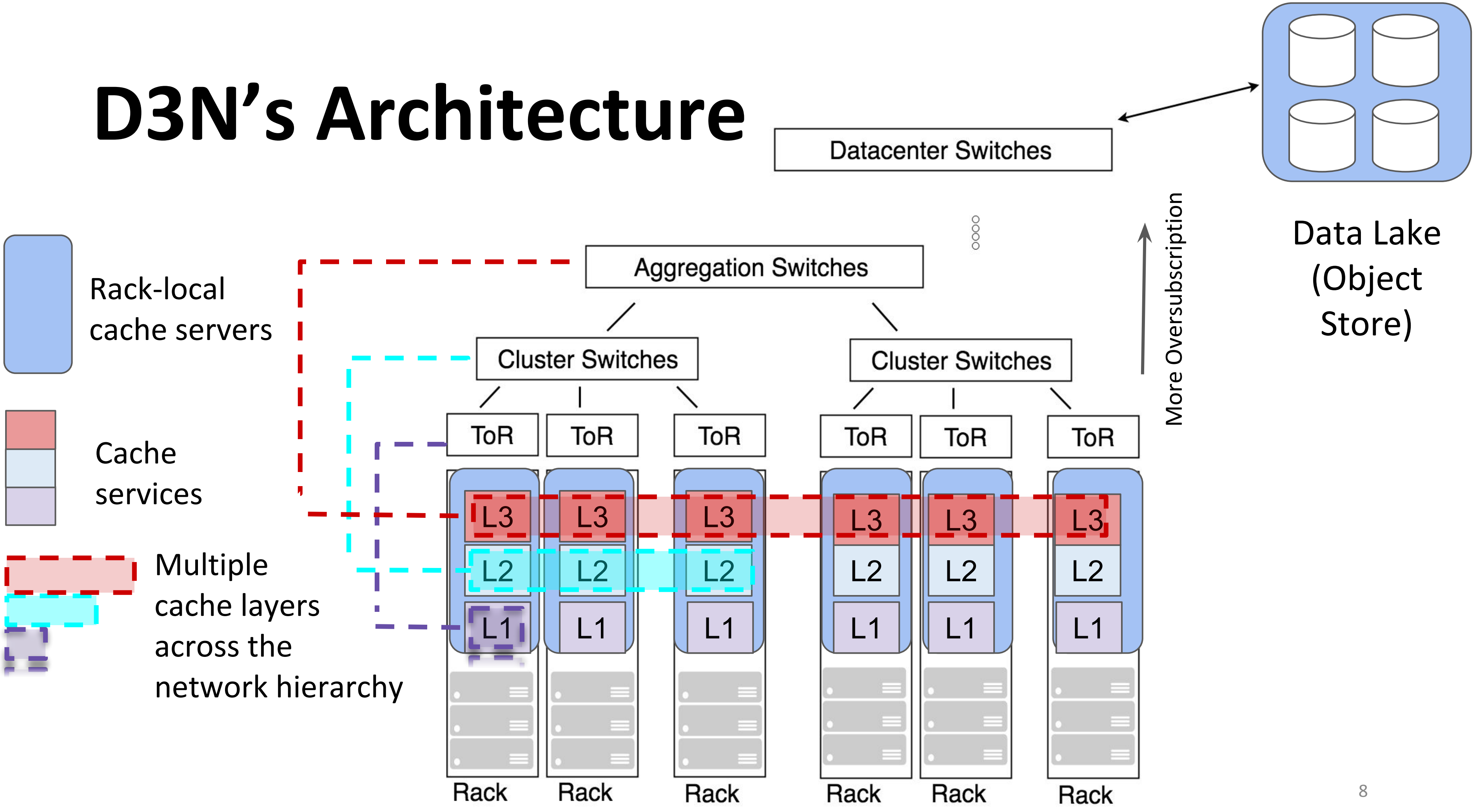
Fundamental Goals of D3N

- Extension of the data lake
- Reduce demand on network
- Automatically adjust to:
  - access pattern
  - network contention

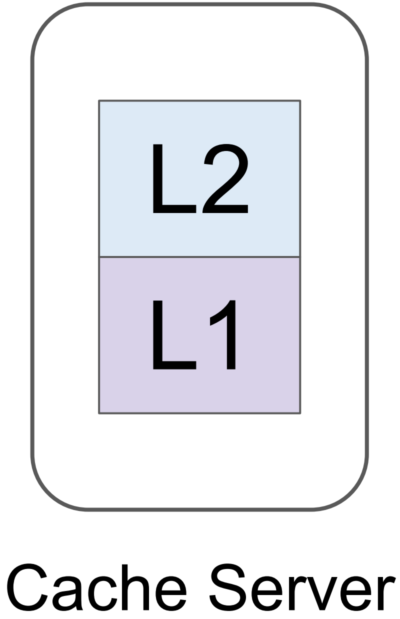
Design Principles

- Transparent to user
- Naturally scalable with the clusters that access it
- Cache policies based purely on local information
- Hierarchical multi-level cache

D3N's Architecture



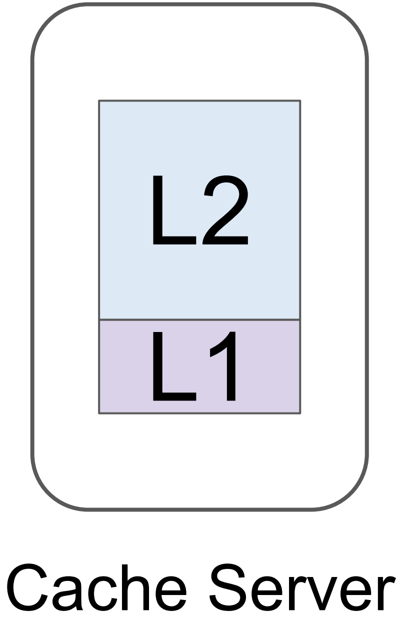
Dynamic Cache Size Management



The algorithm partitions the cache space based on:

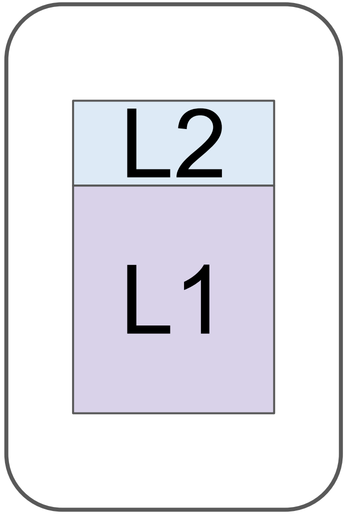
- Access Pattern
- Network Congestion

Dynamic Cache Size Management



- High rack locality with small working set size
- Congestion to storage network

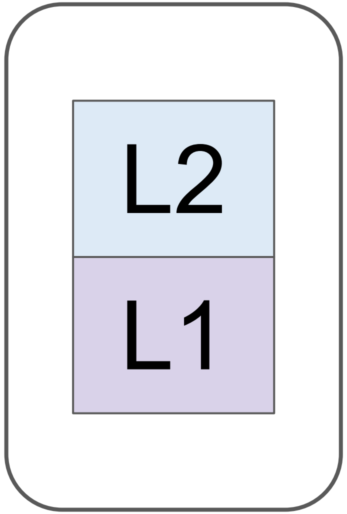
Dynamic Cache Size Management



Cache Server

- High rack locality
- Congestion within the cluster network

Dynamic Cache Size Management



Cache Server

The algorithm partitions the cache space based on:

- Access Pattern
- Network Congestion

Dynamic Cache Size Management

The algorithm measures

- the reuse distance histogram
- mean miss latency

Find the optimal cache size split.

Algorithm 1 Re-use distance measurement

b: requested block
ℓ: layer (1 or 2)
S_t: total cache size (in blocks)
SL_ℓ: shadow LRU list (length S_t)
HC_ℓ: re-use distance histogram
$\overrightarrow{s} = (s_1, s_2)$: cache distribution, $s_1 + s_2 = S_t$

▷ Measure re-use distance for access to block b, layer ℓ
procedure MEASURE(b,ℓ)
  if $b \in SL_\ell$ **then**
    find i s.t. $SL_\ell(i) = b$ ▷ LRU position
    $HC_\ell(i)$++
  reorder SL_ℓ LRU due to access to b

Algorithm 2 Cache distribution adaptation

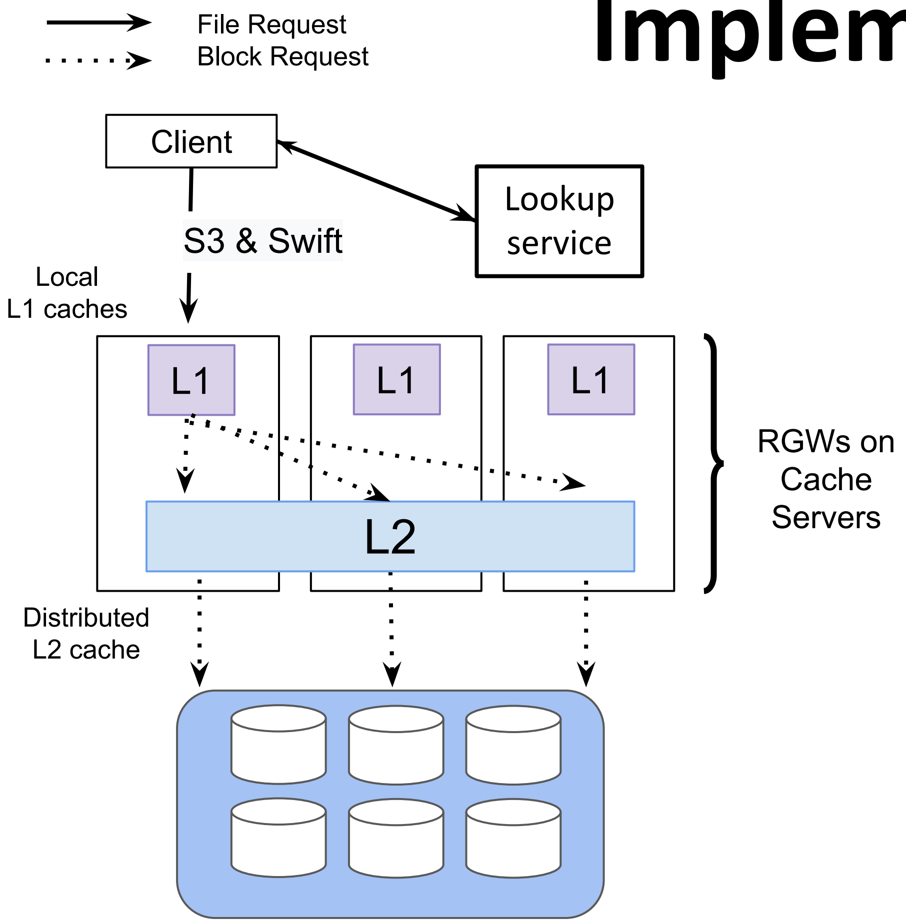
$b, \ell, S_t, \overrightarrow{s}, HC_\ell$: As in Algorithm 1
MR_ℓ: miss rate (i.e. miss ratio Curve)
L_ℓ: measured miss latency
q: adaptation limit (maximum assignment change in blocks)

▷ Calculate updated $L_1 L_2$ cache distribution $\overrightarrow{s}_{new}$
procedure ADAPT
  for ℓ in 1, 2 **do**
    $MR_\ell(i) = _{k=i}^{S_t} HC_\ell(k)$ ▷ Calculate miss ratio curve
  $C_{min} = \inf$
  $s_{new} = \emptyset$
  for $\overrightarrow{s}'$ in $(s_1 - q, s_2 + q) \ldots (s_1 + q, s_2 - q)$ **do**
    $\overrightarrow{m} = (MR_1(s_1) + MR_2(s_2))$ ▷ Predicted miss rate
    $C = m_1 L_1 + m_2 L_2$ ▷ Expected latency
    if $C < C_{min}$ **then**
      $C_{min} = C$
      $s_{new} = s'$

Edge Conditions and Failures

- VM Migration
  - Anycast to DNS lookup server.
  - TCP session keep active until a request is completed.
- Failure of cache server
  - Heartbeat service is used to keep track of active caches.
  - During a failure
    - lookup service will direct new requests to second nearest L1.
    - Consistent hashing algorithm remove the failed node from its map.

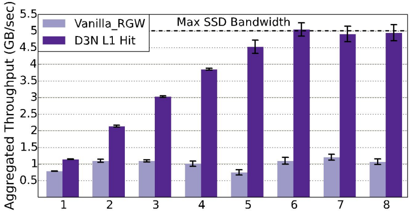
Implementation

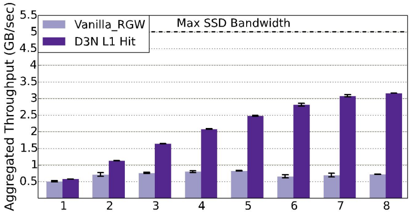


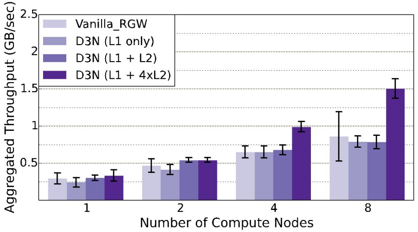
- Modification to **Ceph's RADOS gateway**. We add 2500 lines of code.
- Implements **two level cache,** L1 and L2.
- Read Cache
- Write Cache
  - Write-through
  - Write-back (today no redundancy)
- Stores cached data in 4 MB blocks as individual files on an SSD-backed file system.

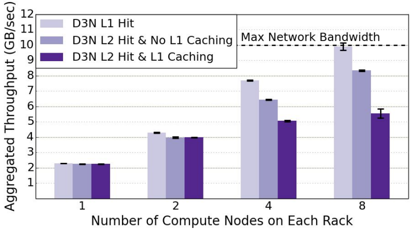
Evaluation of D3N

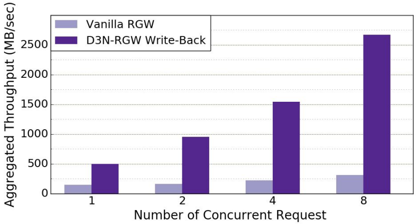
Micro Benchmarks

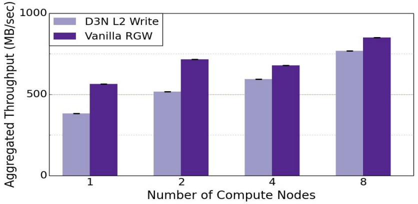




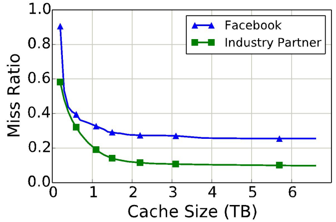


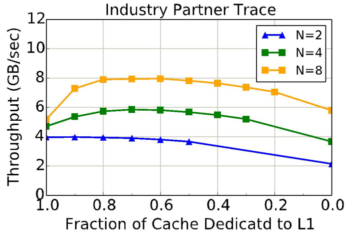


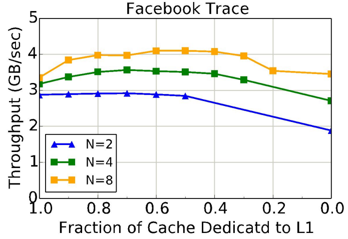




Value of Multi-level

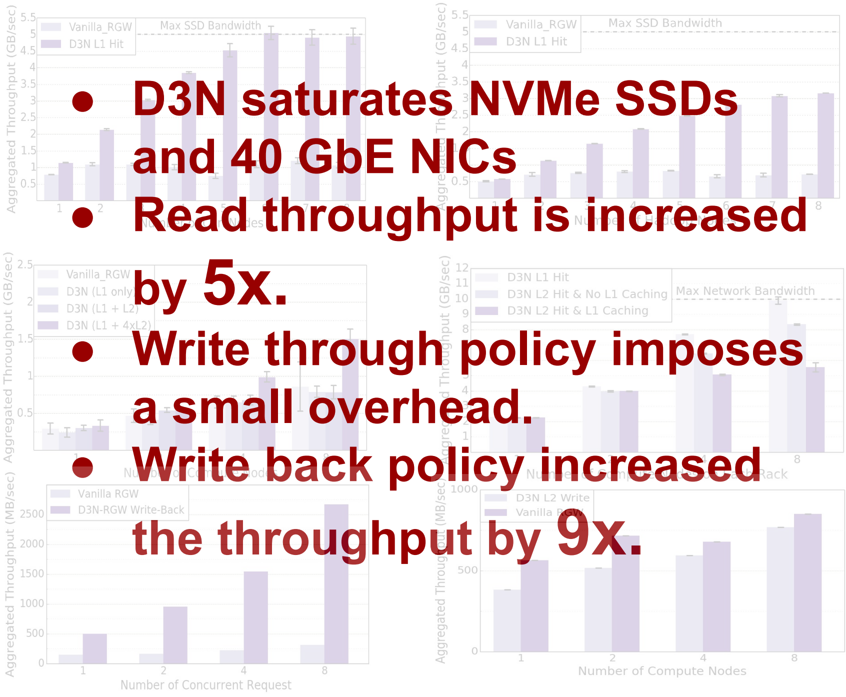






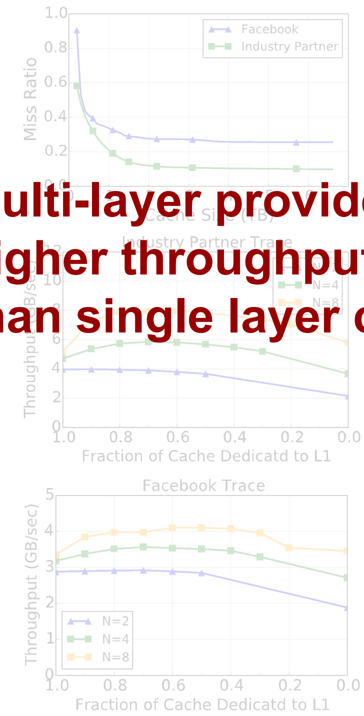
Evaluation of D3N

Micro Benchmarks



- **D3N saturates NVMe SSDs and 40 GbE NICs**
- **Read throughput is increased by 5x.**
- **Write through policy imposes a small overhead.**
- **Write back policy increased the throughput by 9x.**

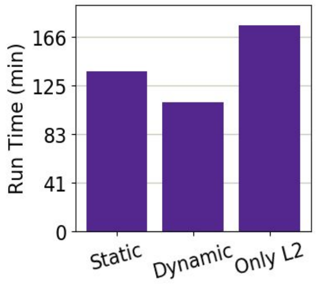
Value of Multi-level

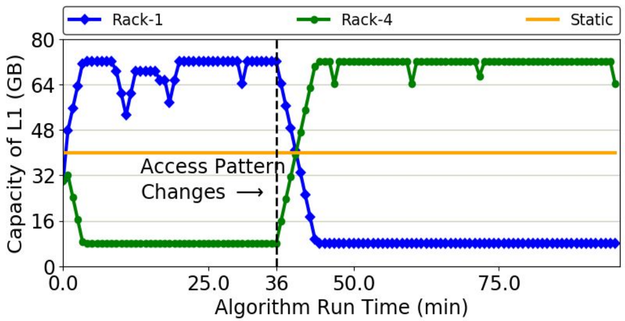


Multi-layer provides higher throughput than single layer cache.

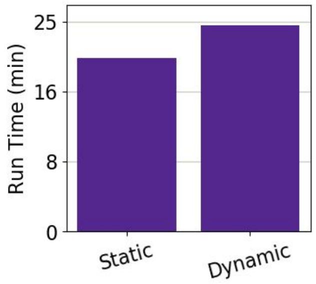
Evaluation of Cache Management

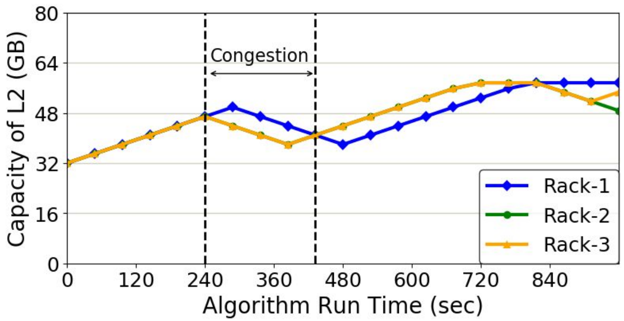
Adaptability to different access patterns





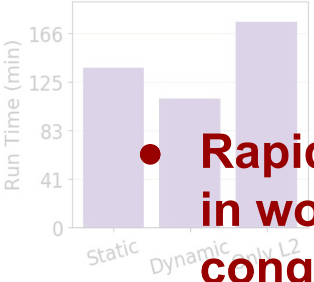
Adaptability to network load changes



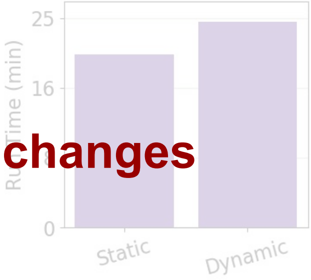


Evaluation of Cache Management

Adaptability to different access patterns



Adaptability to network load changes



- **Rapidly and automatically adjust changes in workload access pattern and congestion on network links.**

Impact of D3N on Realistic Workload

Workloads: Facebook Traces

- 75% reuse
- 40TB data
- Requests were randomly assigned

Benchmark:

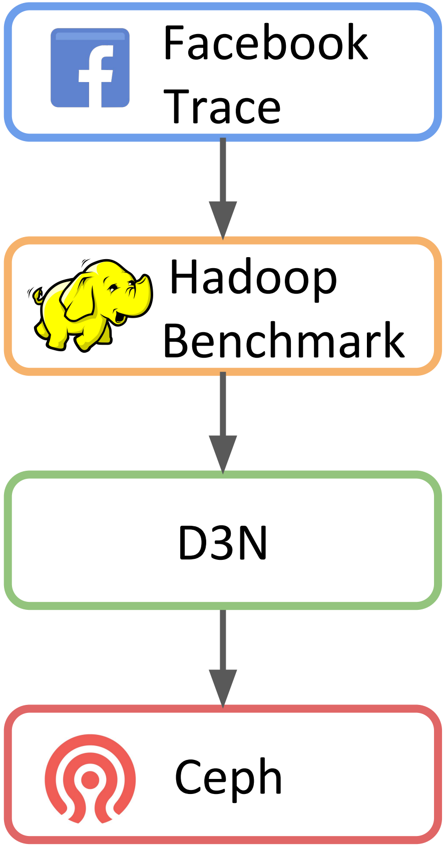
- Mimic the hadoop mappers oncurrent
- Concurrent 144 read requests using “curl”

D3N: 2 cache servers each have

- 1.5 TB NVMe SSDs (RAID 0)
- Fast NIC: 2 x40 Gbit & Slow NIC: 2 x6 Gbit

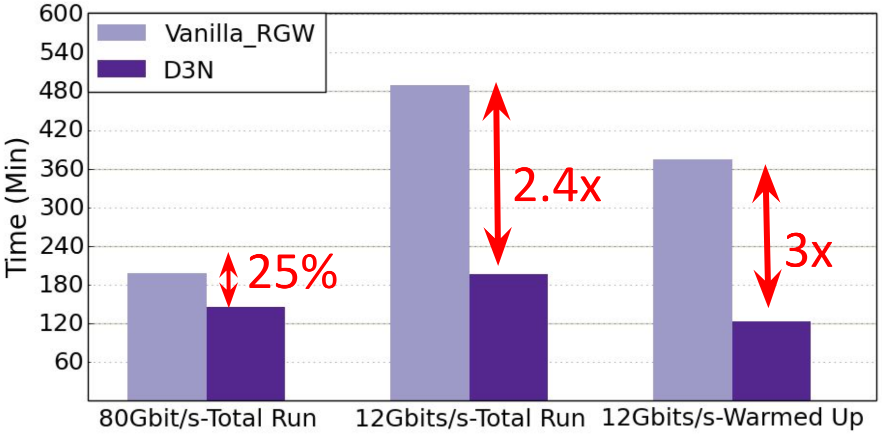
Data lake:

- Ceph (90 HDDs)



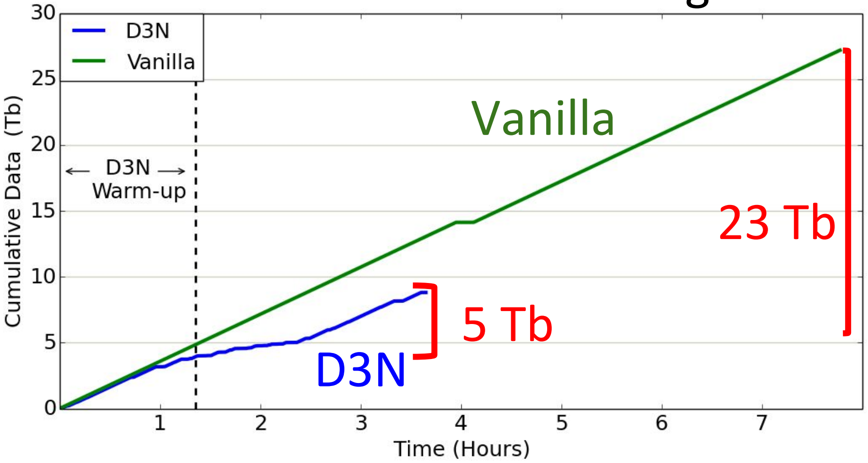
Impact of D3N on Realistic Workload

The trace completion time



D3N improves performance significantly.

Cumulative data transferred from back-end storage



More than 4x reduction to backend traffic.

Concluding Remarks

Proposed a transparent multi layer caching

- Extension of the data lake
- Implemented two layer prototype

Results:

- Cache partitioning algorithm dynamically adapt changes
- Reduces demand datacenter wide
- Improve the analytic workloads performance

Red Hat is currently productizing D3N.

- https://github.com/ekaynar/ceph

Project Websites

- https://www.bu.edu/rhcollab/projects/d3n/
- https://massopen.cloud/d3n/

Thank you