DROP: Facilitating Distributed Metadata Management in EB-scale Storage Systems

Quanqing Xu, Rajesh Vellore Arumugam, Khai Leong Yong, Sridhar Mahadevan

Data Storage Institute

Agency for Science Technology and Research (A*STAR), Singapore

A*STAR

Challenges in distributed metadata management

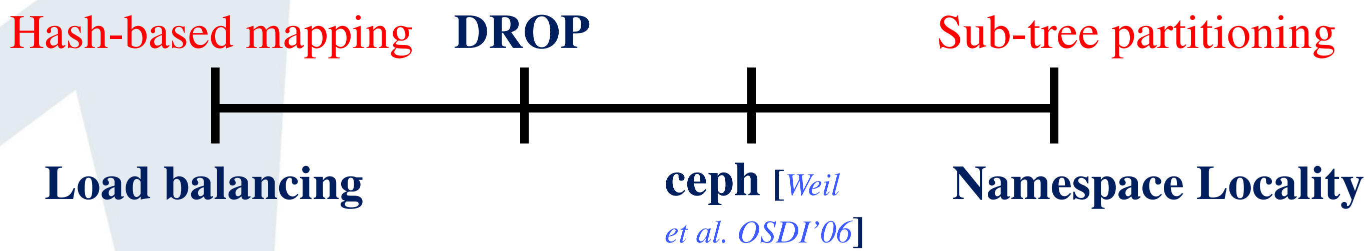
- Metadata is very huge for EB-scale data
  - ✓ *0.1% ~ 1% of data size [Miller et al. NVRAMOS08], e.g., 1PB ~ 10PB for 1EB*
- Storage load balancing in metadata servers
  - ✓ *For EB-scale storage systems including hundred billions of files, e.g., facebook managed 260 billions of images, >20 PB [Beaver et al. OSDI'10]*
- Request load balancing in metadata servers
  - ✓ *For mixed workloads generated by a large number of concurrent users, e.g., PanFS [Welch et al. FAST'08]*
- Organization and maintenance of very large directories
  - ✓ *each directory contains ten millions of files, e.g., GIGA+[Patil et al. FAST'11]*

Dynamic Ring Online Partitioning (DROP)

- Highly scalable and available key-value store
  - ✓ *using chain replication [Renesse et al. OSDI'04]*
- Simple interface
  - ✓ *lookup(key) → IP, under put and get*
- Locality-preserving Hashing (LpH)
  - ✓ *Improves namespace locality*
  - ✓ *Increases put/get success rate depending on fewer MDSs*
  - ✓ *Upgrades put/get performance involving fewer lookups*
- Histogram-based Dynamic Load Balancing (HDLB)
  - ✓ *Quickly converges to load balancing in fully distributed systems*
  - ✓ *NP-hard problem*

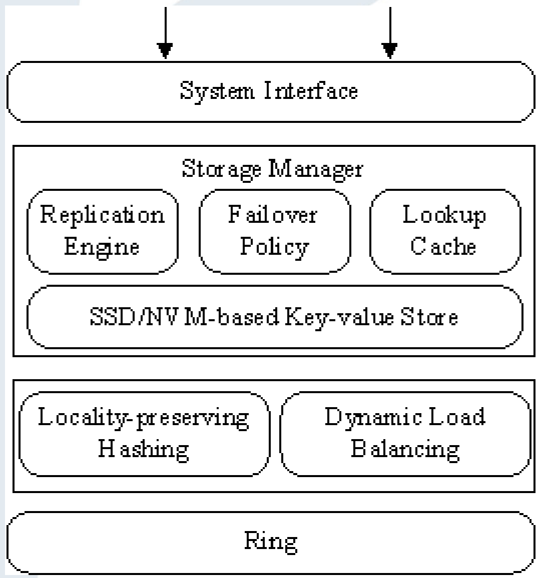
DROP compared to State-of-the-art distributed metadata management schemes

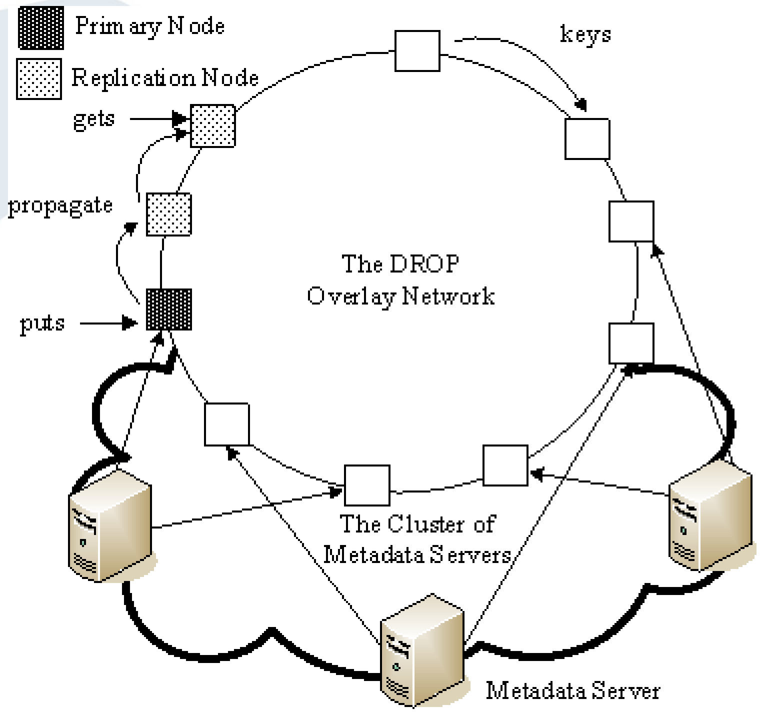
- Hash-based mapping
  - ✓ *eliminates metadata locality of files*
  - ✓ *cannot effectively handle directory operations*
    e.g., renaming a directory
- Sub-tree partitioning
  - ✓ *cannot scale well for large directories*
  - ✓ *does not evenly distribute workload among MDS cluster*
  - ✓ *metadata needs to be migrated to achieve coarse load balancing*



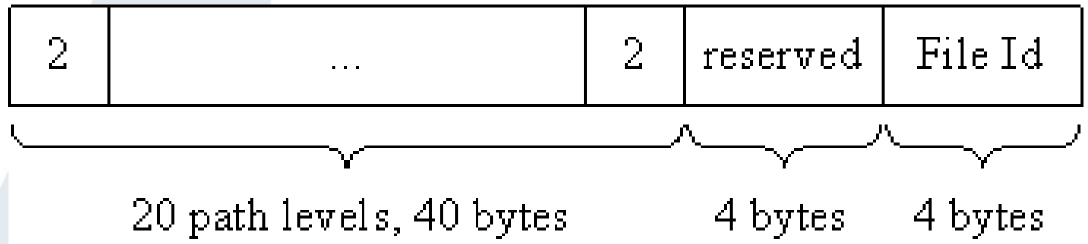
System Architecture of DROP

- Virtual nodes are used as a means of improving load balancing
  - ✓ *To build the ring-based overlay network*
- Chain replication *[Renesse et al. OSDI'04]*
  - ✓ *Updates are sent to the head of the chain*
  - ✓ *Queries are sent to the tail of the chain*





Locality-preserving Hashing (LpH)



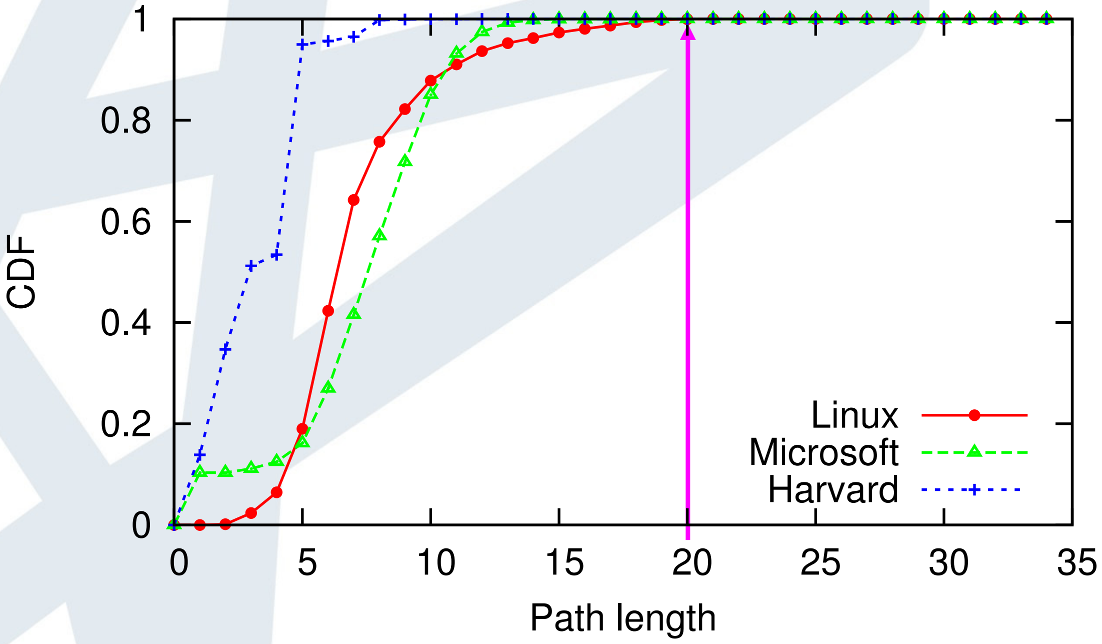
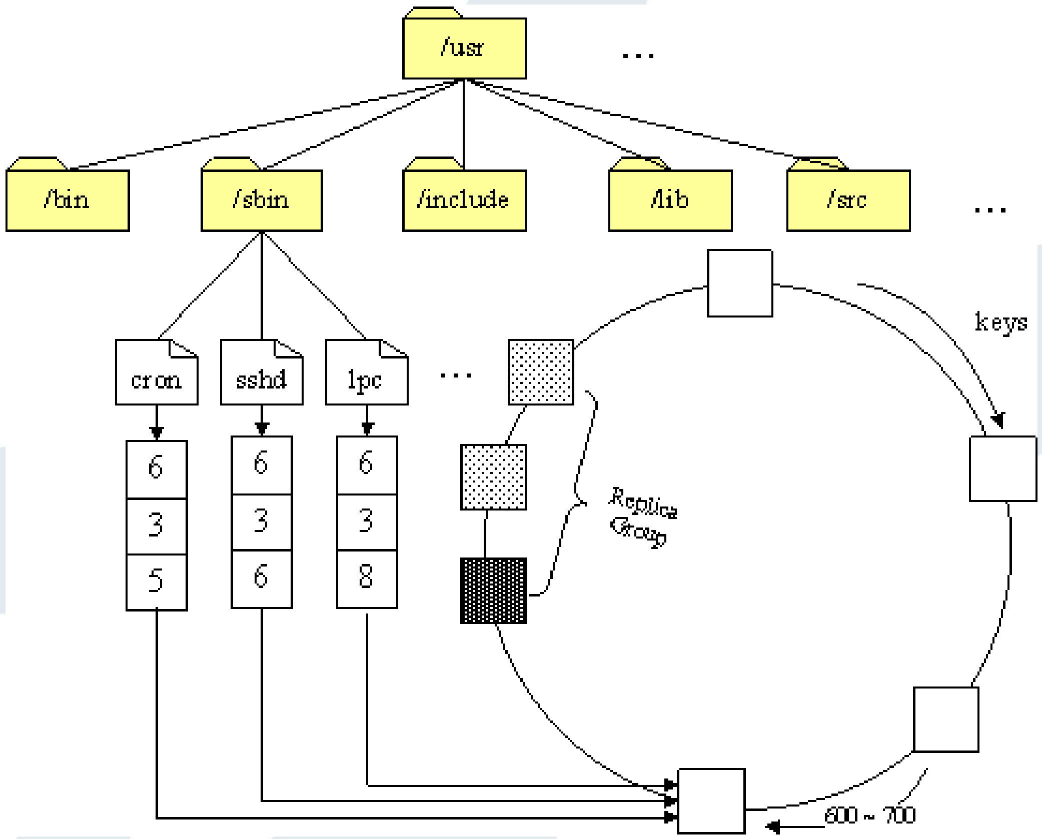


Fig.1 CDF of path length in three traces

Metadata Publishing



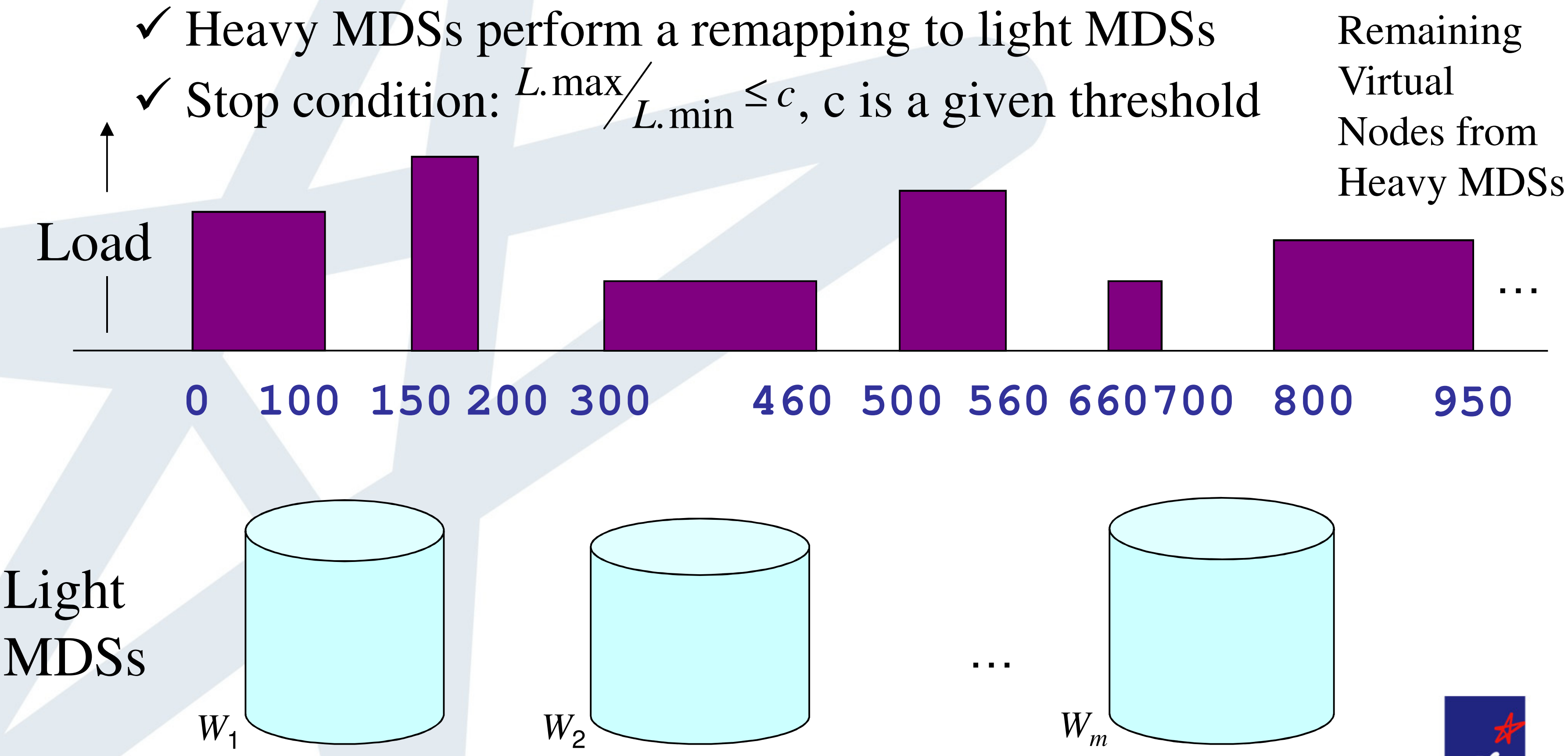
- To ensure the structural integrity of storage systems
- To improve overall performance and availability

Histogram-based Dynamic Load Balancing (HDLB)

- A metadata server is load balancing
  - ✓ *Its load satisfies* $1/t \le \frac{L_i}{\bar{L}} \le t$
- Load imbalance
  - ✓ *Metadata placement is no longer uniform*
- Simple dynamic load balancing
  - ✓ *Mercury [Bharambe et al. SIGCOMM'04], Karger [Karger et al. SPAA'04]*
  - ✓ *One-to-Many and Many-to-Many [Godfrey et al. INFOCOM'04]*
  - ✓ *Converge to load balance quickly*

Histogram-based Dynamic Load Balancing …

- Basic idea: remapping
- Steps
  - ✓ Find average, check if heavy or light
  - ✓ Heavy MDSs perform a remapping to light MDSs
  - ✓ Stop condition: $L.\max / L.\min \le c$, c is a given threshold



Histogram-based Dynamic Load Balancing …

- Given a set of *m* metadata servers and a set of *n* virtual nodes
  - ✓ $S=\{s_i, i=1,\ldots,m\}$, $V=\{v_j, j=1,\ldots,n\}$
  - ✓ v_j *has a weight* w_j, s_i *has a remaining capacity (weight)* W_i
- Formulated as a 0-1 Multiple Knapsack Problem (MKP)
  - ✓ *To minimize the wasted space in the MDSs*
  - ✓ *When determining how to reassign n virtual nodes to m metadata servers*

Histogram-based Dynamic Load Balancing …

$$\text{maximize} \quad z = 1/\sum_{i=1}^{m} s_i \tag{1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = 1, j \in N = \{1, \ldots, n\} \tag{1b}$$

$$\sum_{j=1}^{n} w_j x_{ij} + s_i = W_i y_i, i \in M = \{1, \ldots, m\} \tag{1c}$$

$$x_{ij} \in \{0, 1\}, y_i \in \{0, 1\}, i \in M, j \in N \tag{1d}$$

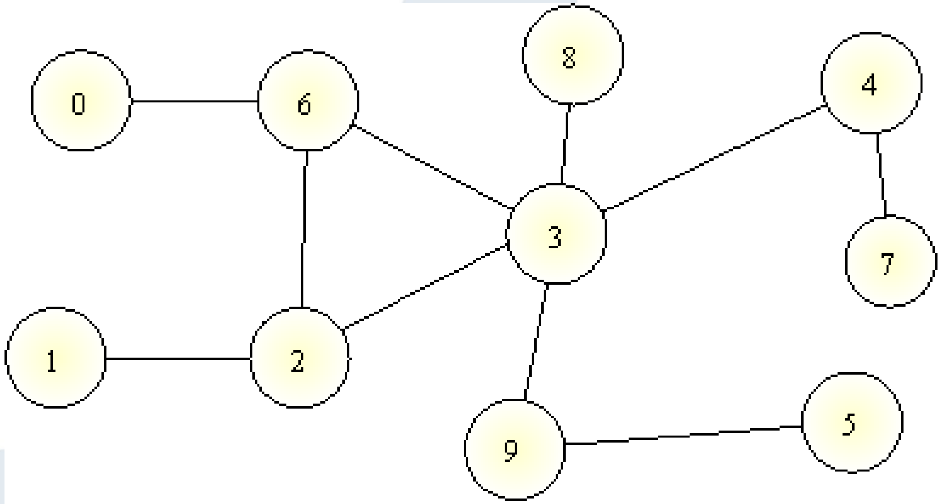
where

$$x_{ij} = \begin{cases} 1 & \text{if virtual node } j \text{ is reassigned to MDS } i \\ 0 & \text{otherwise} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{if MDS } i \text{ is used} \\ 0 & \text{otherwise} \end{cases}$$

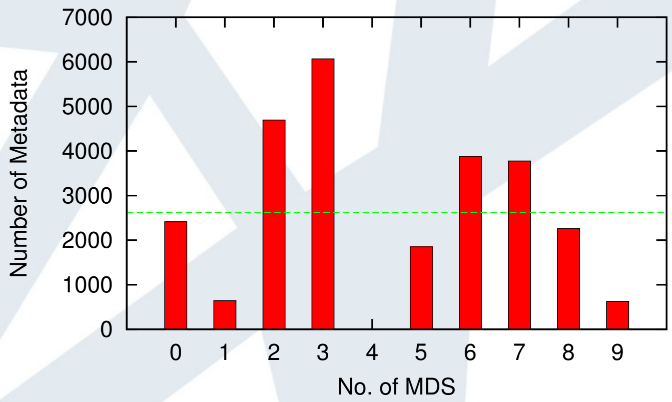
$$s_i = \text{space left in MDS } i$$

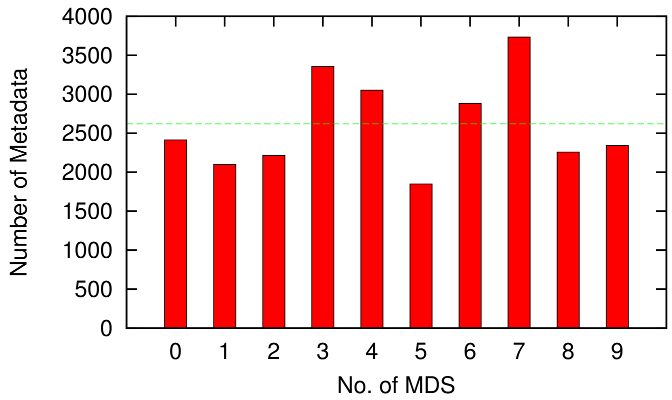
An Example of Convergence of HDLB




```
'1': {157, 487}, '0': {1147, 877, 389},
'3': {640, 432, 605, 3602, 786}, '2':
{2025, 191, 1011, 1469}, '5': {82, 18,
737, 1011}, '4': {1}, '7': {3, 3771},
'6': {3277, 47, 548}, '9': {419, 209},
'8': {941, 16, 508, 794}
```

```
'1': {487, 157, 1011, 395, 47}, '0':
{1147, 877, 389}, '3': {3356}, '2':
{2025, 191}, '5': {1011, 737, 82, 18},
'4': {1, 786, 640, 605, 432, 37, 3, 548},
'7': {3734}, '6': {2882}, '9': {419, 209,
246, 1469}, '8': {941, 794, 508, 16}
```





DSI
A*STAR

Evaluation (simulation)

Experimental setting

- ✓ *Three traces*
- ✓ *Virtual node's ID*

  SHA-384
- ✓ *Competitors*

  FileHash, DirHash and Subtree

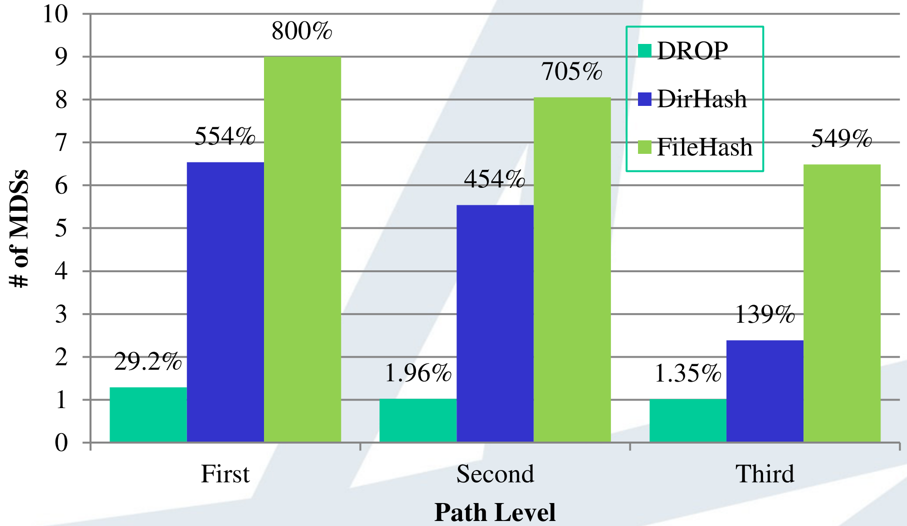
Evaluation metrics

- ✓ *Namespace locality*
- ✓ *Convergence rate*
- ✓ *Load distribution*
- ✓ *Scalability*
- ✓ *Migration overhead*

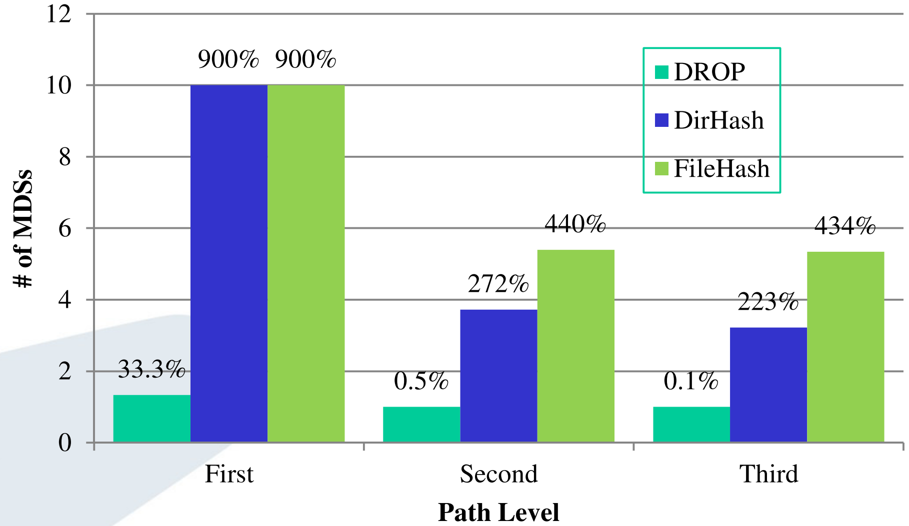
Trace	# of unique files	Path data	Maximum length
Linux	2,216,596	147M	22
Microsoft	7,725,928	416M	34
Harvard	9,213,524	188M	18

A*STAR DSI

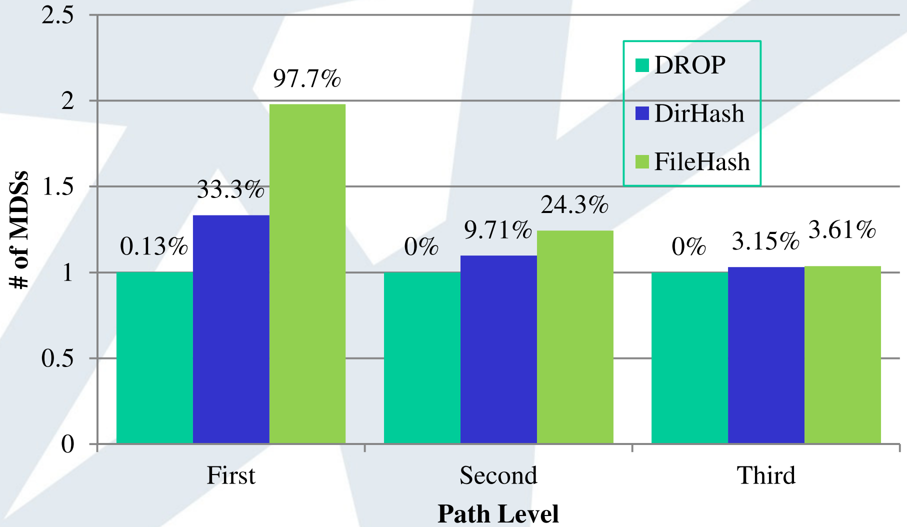
Locality Comparison



(a) Linux trace



(b) Microsoft Windows trace

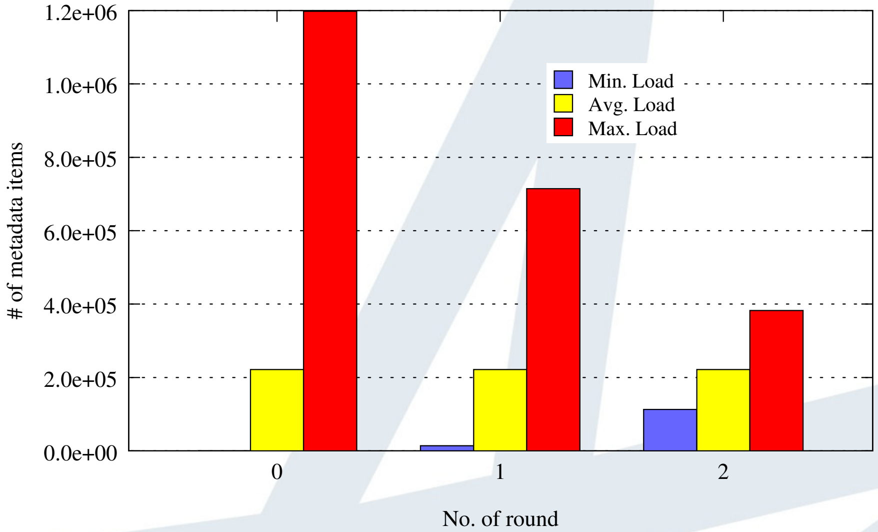


(c) Harvard trace

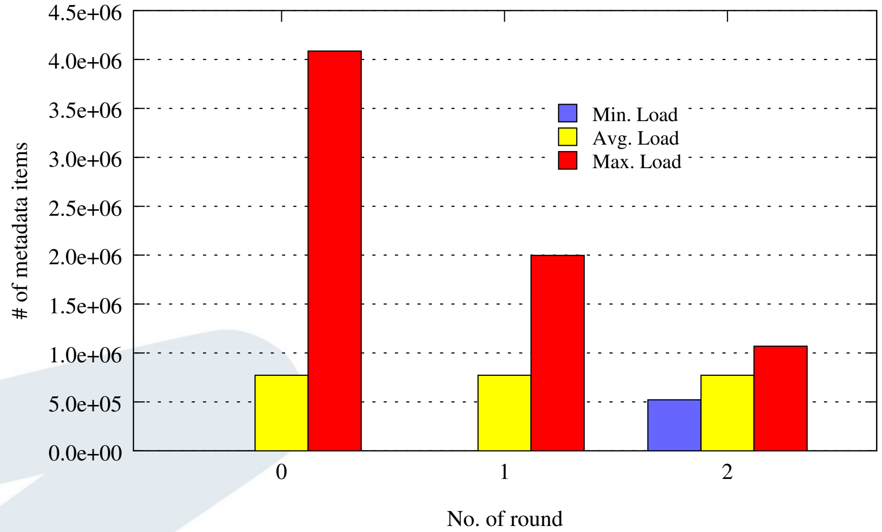
- DROP assigns keys that are consistent with the order of pathnames.
- DirHash and FileHash don't consider the order of pathnames so that namespace locality is lost thoroughly.

A*STAR DSI

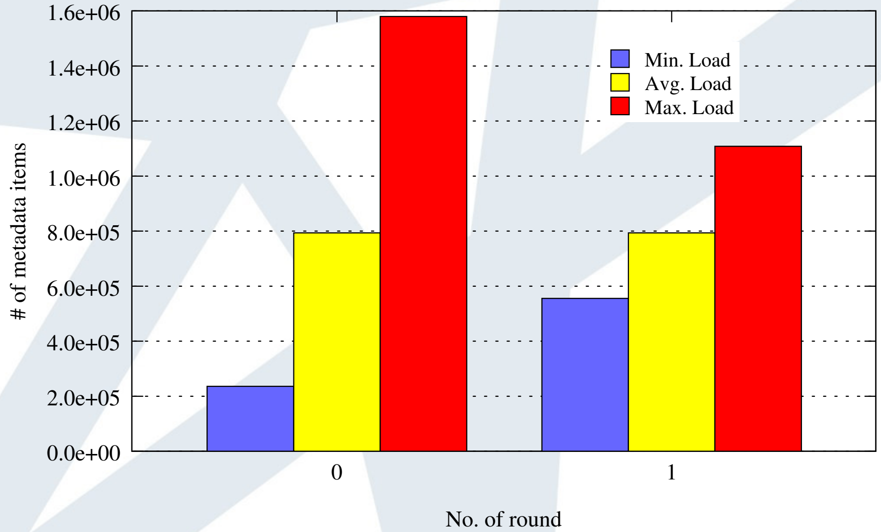
Convergence Rate



(a) Linux trace



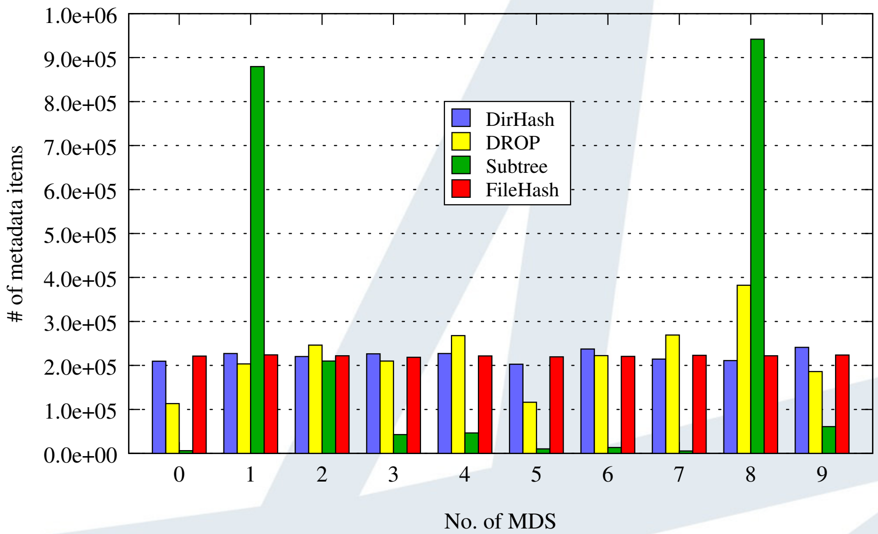
(b) Microsoft Windows trace



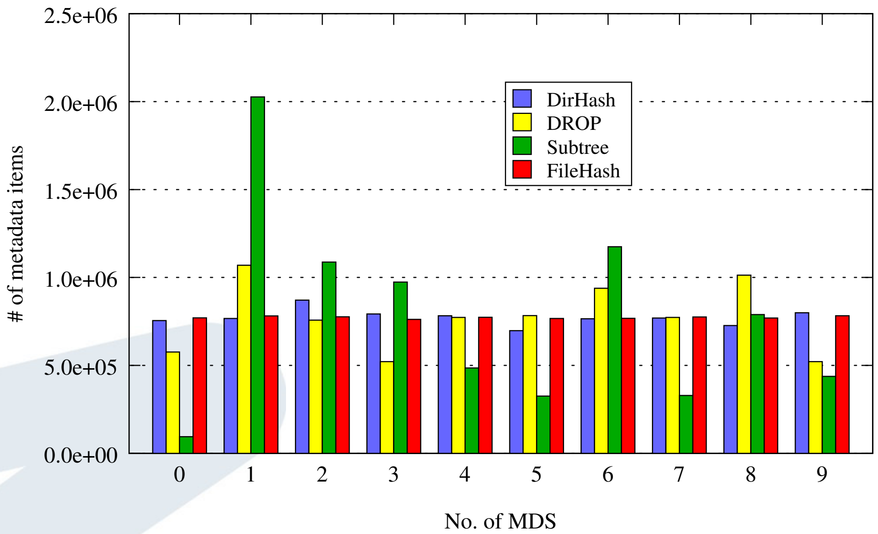
(c) Harvard trace

- DROP has excellent convergence rate.
- There are much more directories with fewer files in Harvard trace than other two traces, so the convergence rate is the best.

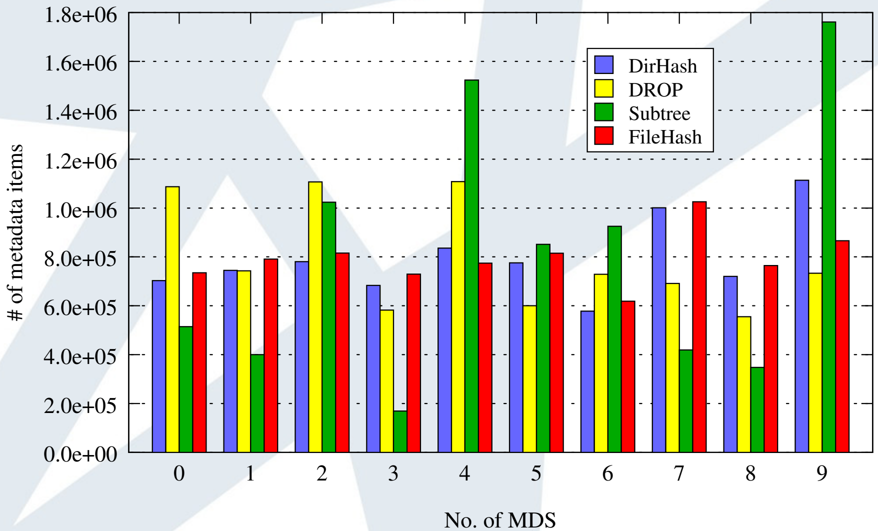
Load Distribution



(a) Linux trace



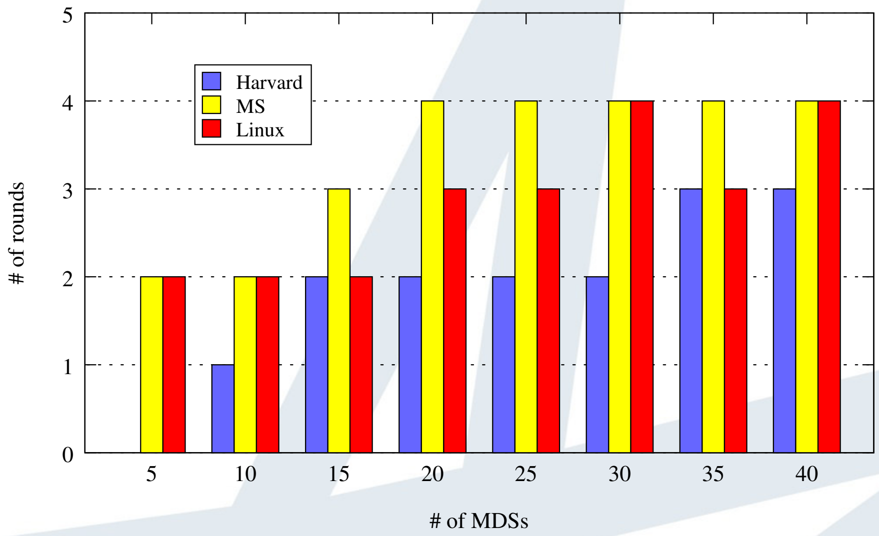
(b) Microsoft Windows trace



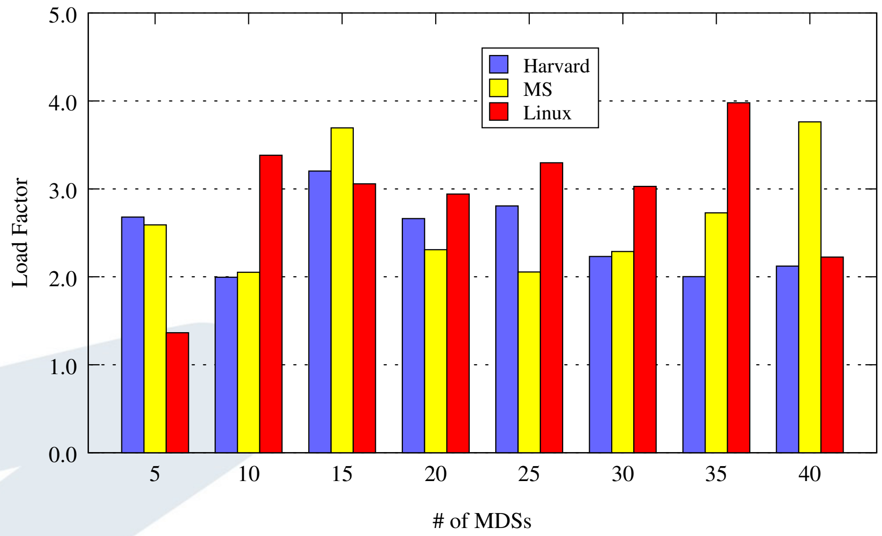
(c) Harvard trace

- DROP has excellent load distribution.
- FileHash and Subtree are the best one and worst one respectively.

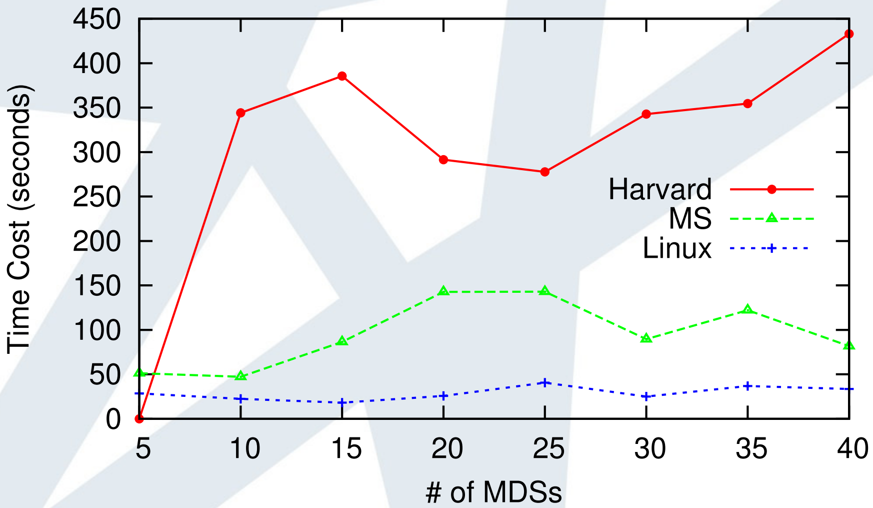
Scalability



(a) Convergence rate



(b) Load Balancing

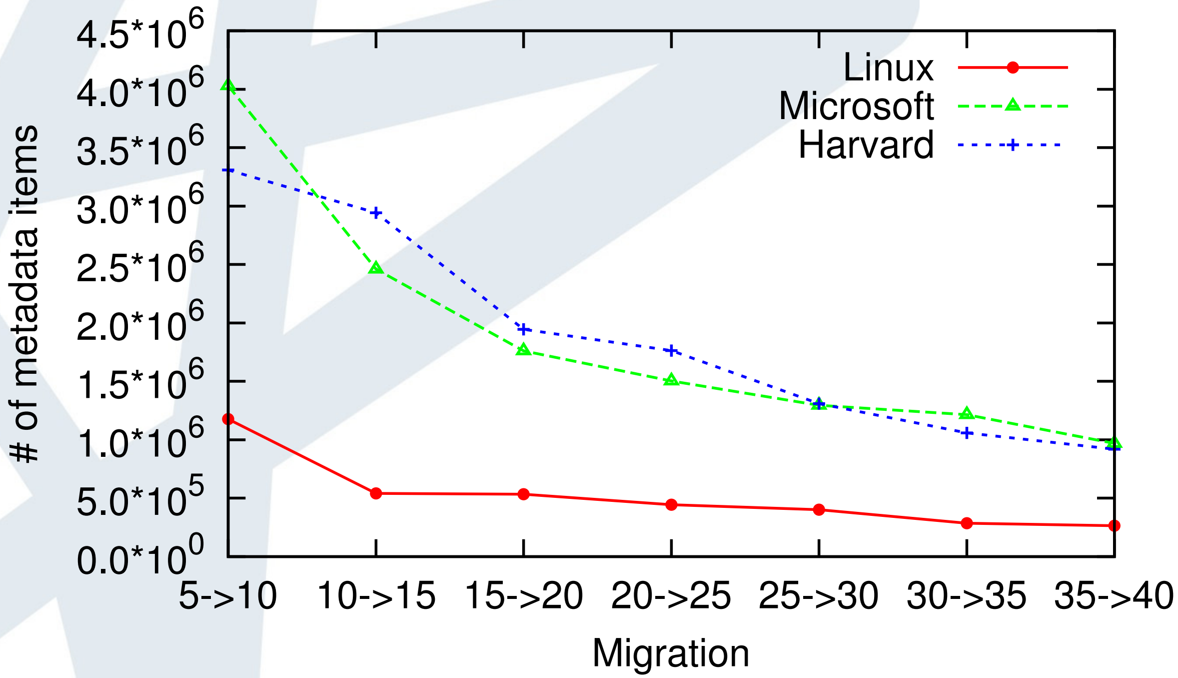


(c) Time Cost

- HDLB can balance the storage load over time.
- It can enable fast and efficient load balancing.
- It has excellent efficiency with different cluster sizes.

Migration Overhead

- Excellent incremental scalability
  - ✓ *HDLB tries, at each step, to reduce the migration overhead by making virtual node assignments among the same group.*



Conclusions

- Locality-preserving Hashing (LpH)
  - ✓ *To keep excellent namespace locality*
- Histogram-based Dynamic Load Balancing (HDLB)
  - ✓ *Converges quickly in fully distributed systems*
- Advantages of DROP
  - ✓ *Excellent locality that is close to static Subtree*

    It only loses 29.2%, 33.3% and 0.13% locality in Linux, Windows and Harvard traces in the first path level.
  - ✓ *Good load distribution that is close to DirHash*

    DirHash is an industrial standard that is used by Lustre.
  - ✓ *Highly scalable*

A*STAR DSI

Thank You

For further query, please contact:

Xu_Quanqing@dsi.a-star.edu.sg