Tradeoffs in Scalable Data Routing for Deduplication Clusters

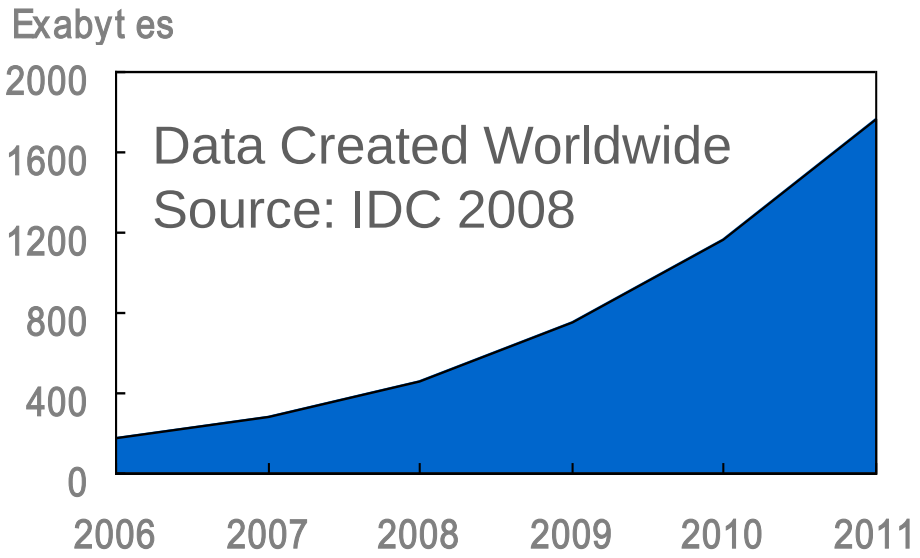
Wei Dong[†], Fred Douglis, Kai Li[†]
Hugo Patterson, Sazzala Reddy, Philip Shilane

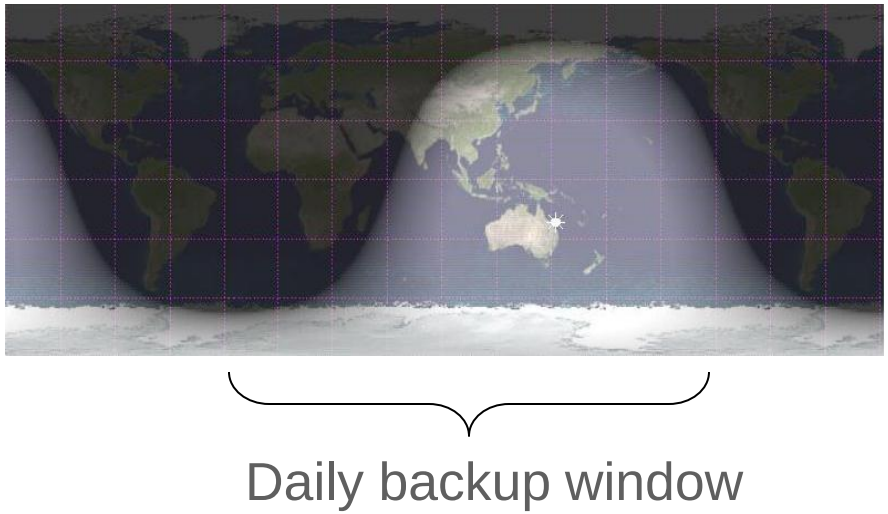
EMC, [†]Princeton University

EMC² where information lives®

Increasing Backup Demands

- Exponential growth of data
- Short backup window

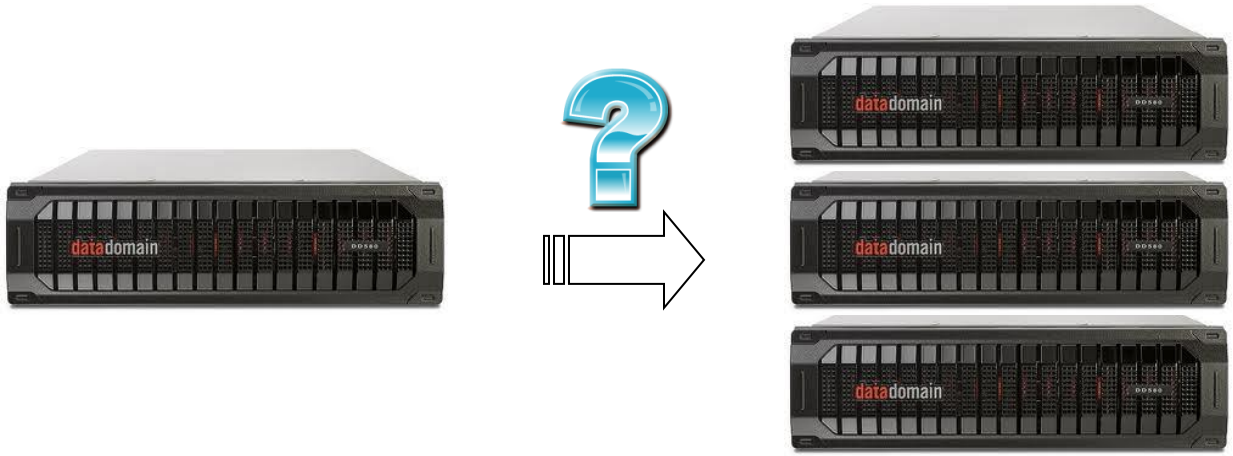




- **Capacity** and **throughput** requirements are increasing

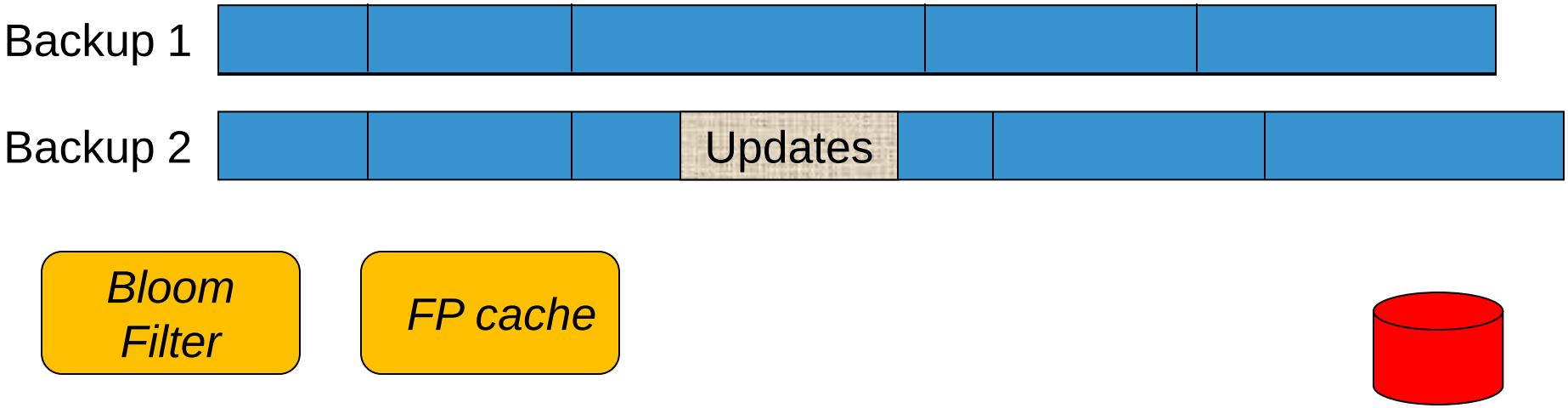
Motivation

- Backup requirements exceed the scale of an individual high-end appliance
- Build a deduplication cluster leveraging existing high-throughput single-node systems



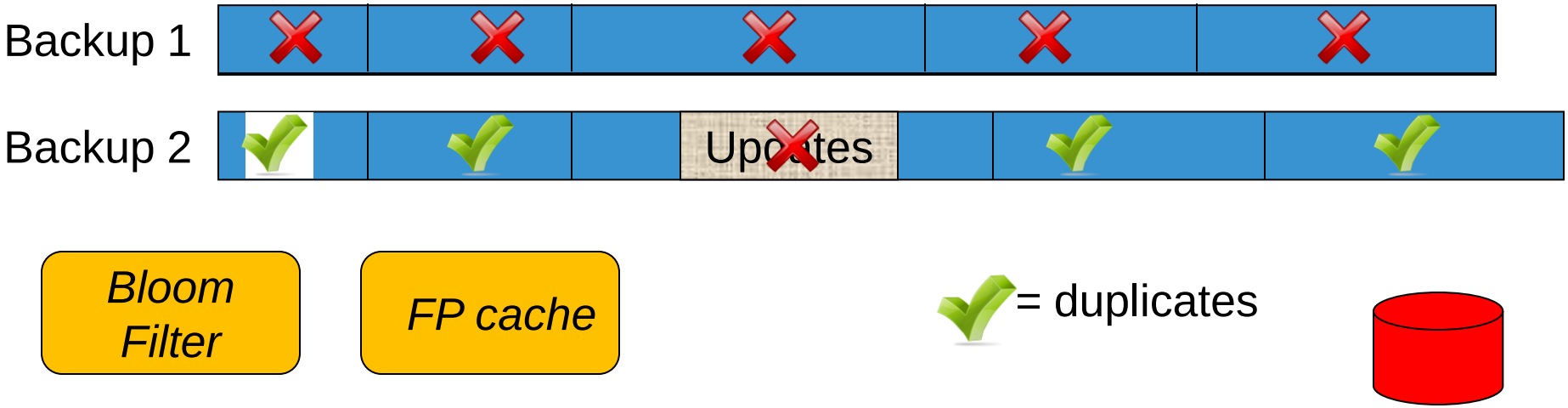
Background: Content-Based Chunking

- Smaller chunks ➔ better deduplication
  - 10X to 30X data reduction in typical deployment
- Throughput bottleneck: fingerprint lookups
  - Stream-Informed Segment Layout: cache locality and disk avoidance via containers and Bloom filters [Zhu et al 2008]



Background: Content-Based Chunking

- Smaller chunks ➔ better deduplication
  - 10X to 30X data reduction in typical deployment
- Throughput bottleneck: fingerprint lookups
  - Stream-Informed Segment Layout: cache locality and disk avoidance via containers and Bloom filters [Zhu et al 2008]



Related Work

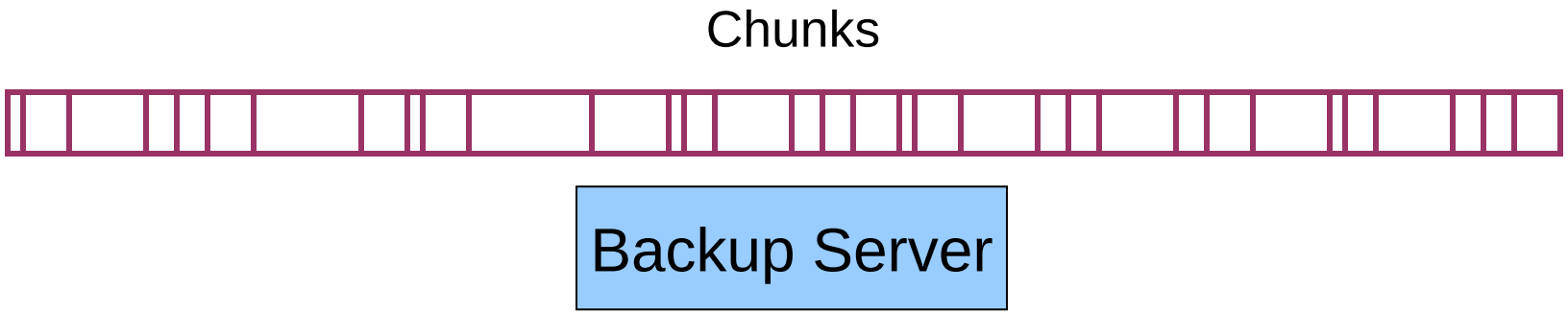
- Stream-Informed Segment Layout [Zhu et al 2008]
- HYDRAStor [Dubnicki et al 2009]
  - Routing chunks to nodes according to content
  - Good performance
  - Worse deduplication rate due to 64 KB chunks
- Extreme Binning [Bhagwat et al 2009]
  - Routing files according to minimal chunk hash
  - Dedupe primarily against earlier version of same file

Our Approach: Routing Super-Chunks

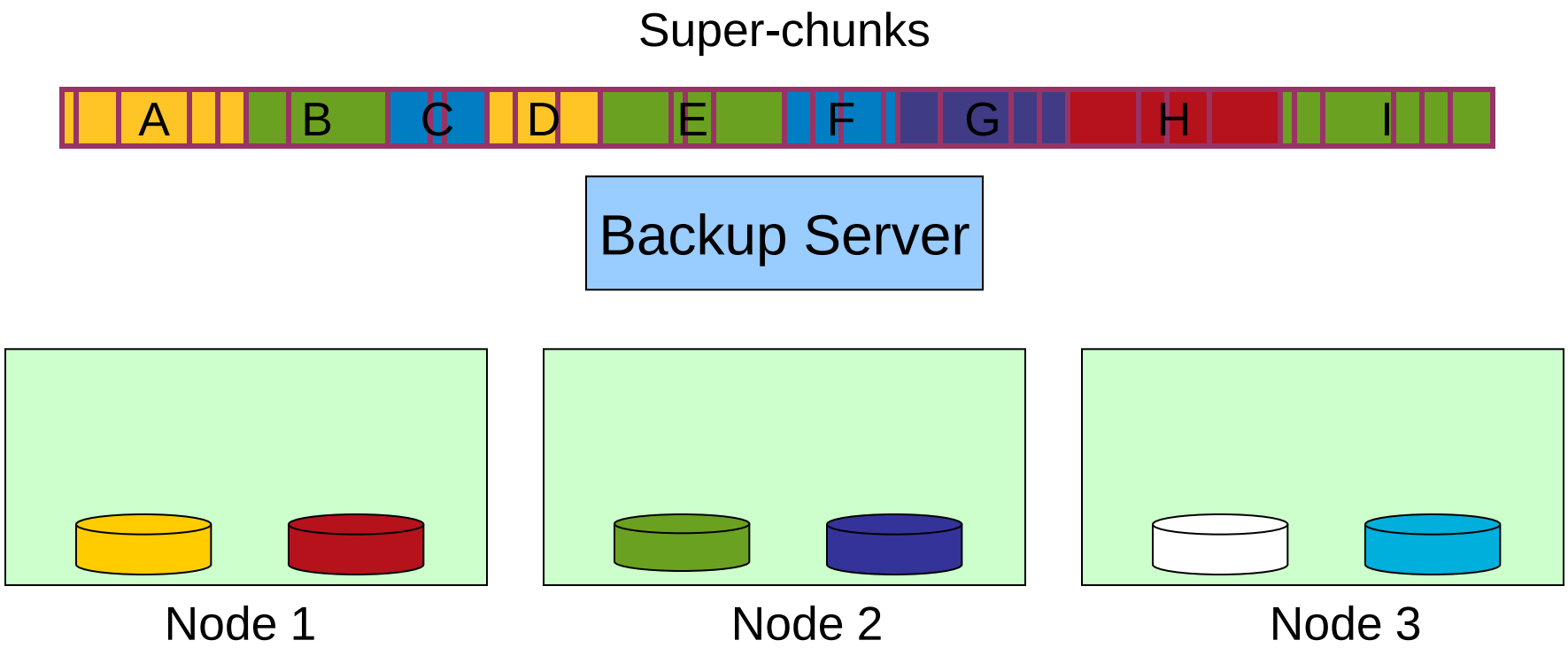
- Super-chunks: consecutive groups of chunks
- Scalable capacity
  - Deduplication node-by-node at chunk level
    - The same chunk may appear on multiple nodes
  - Balanced storage usage across cluster nodes
- Scalable throughput
  - Locality preserved within a super-chunk
  - Routing super-chunks provides better throughput than routing chunks due to caching effects

EMC² where information lives®

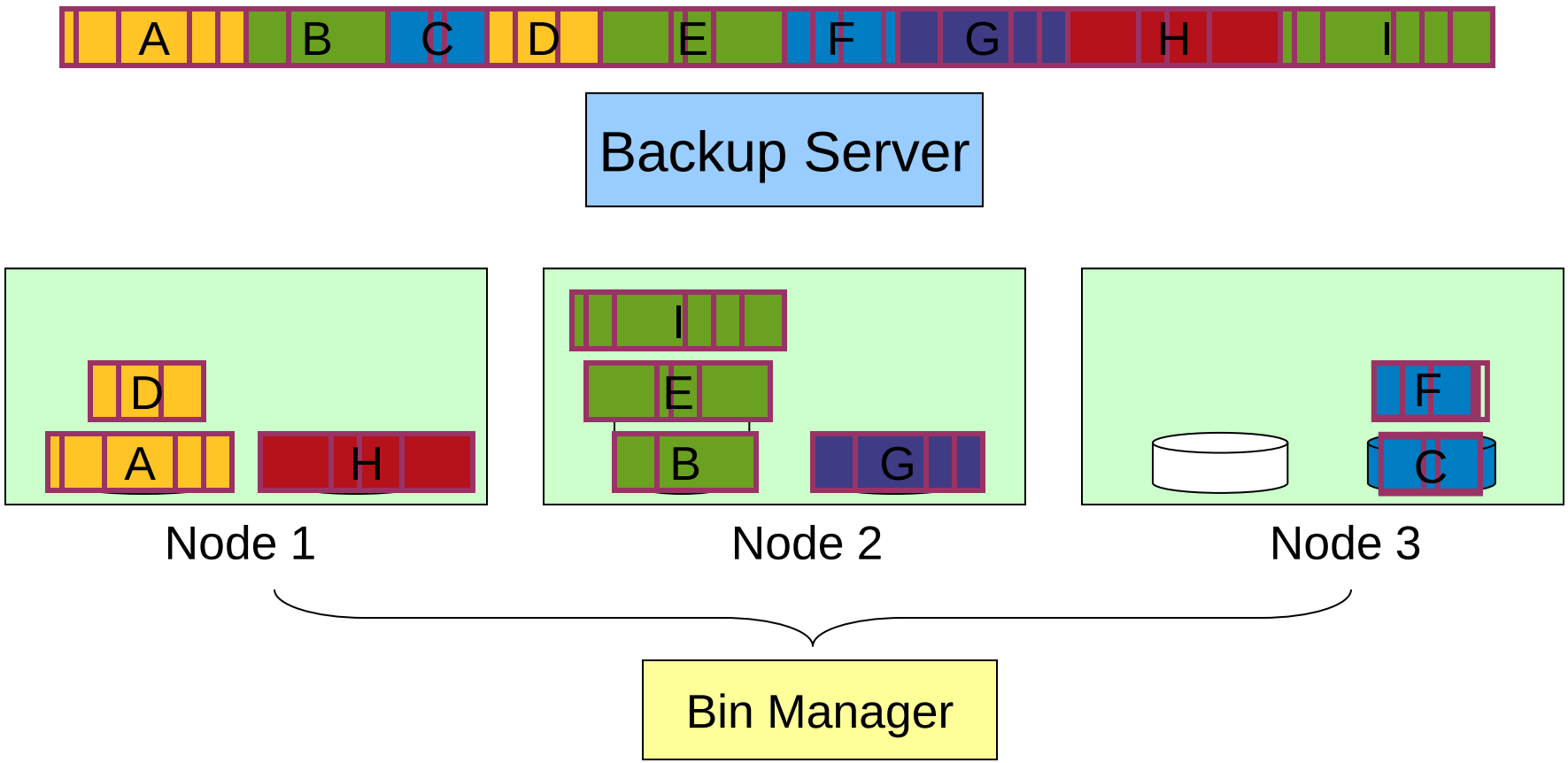
Architecture and Data Flow



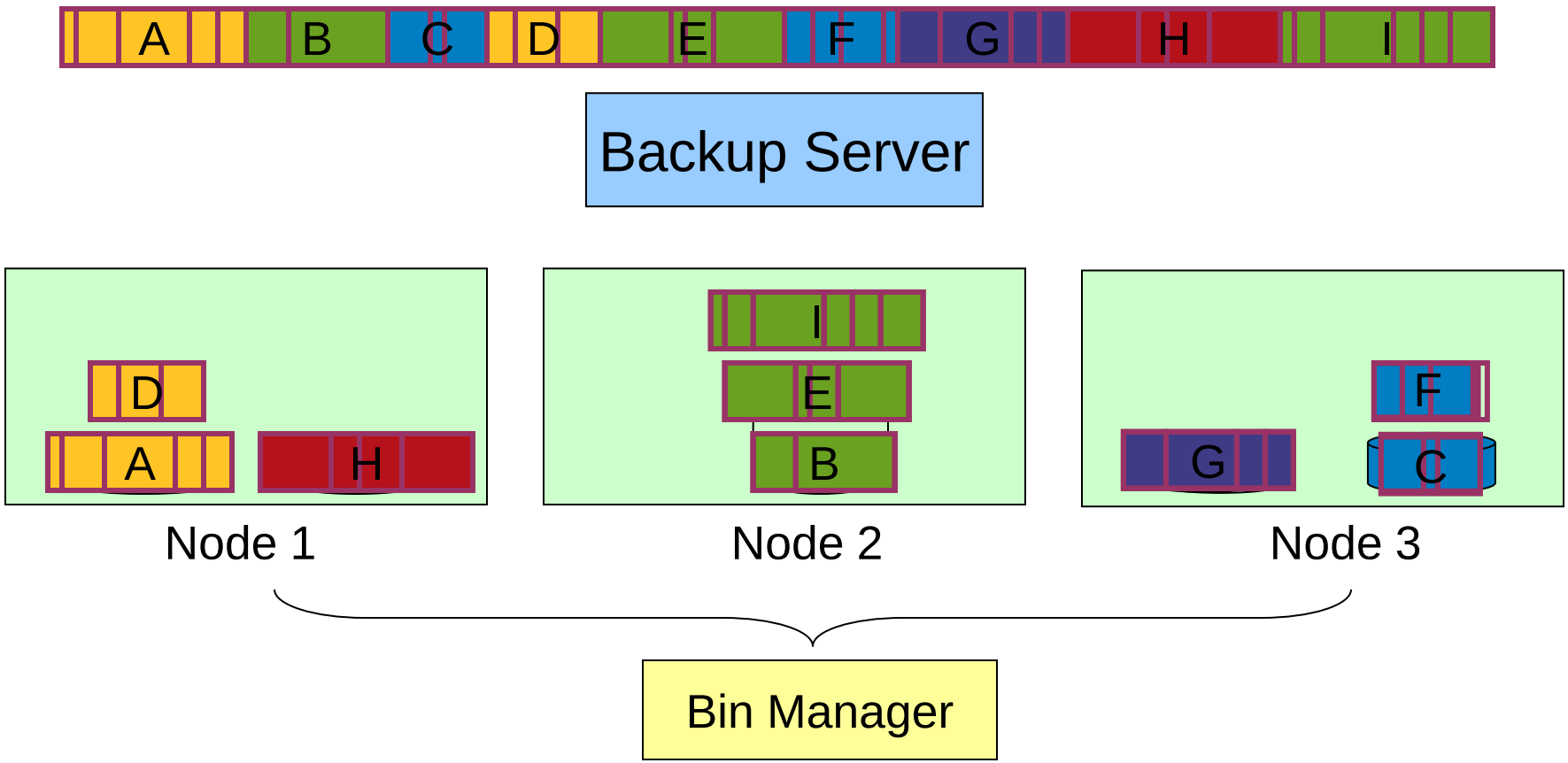
Architecture and Data Flow



Architecture and Data Flow



Architecture and Data Flow



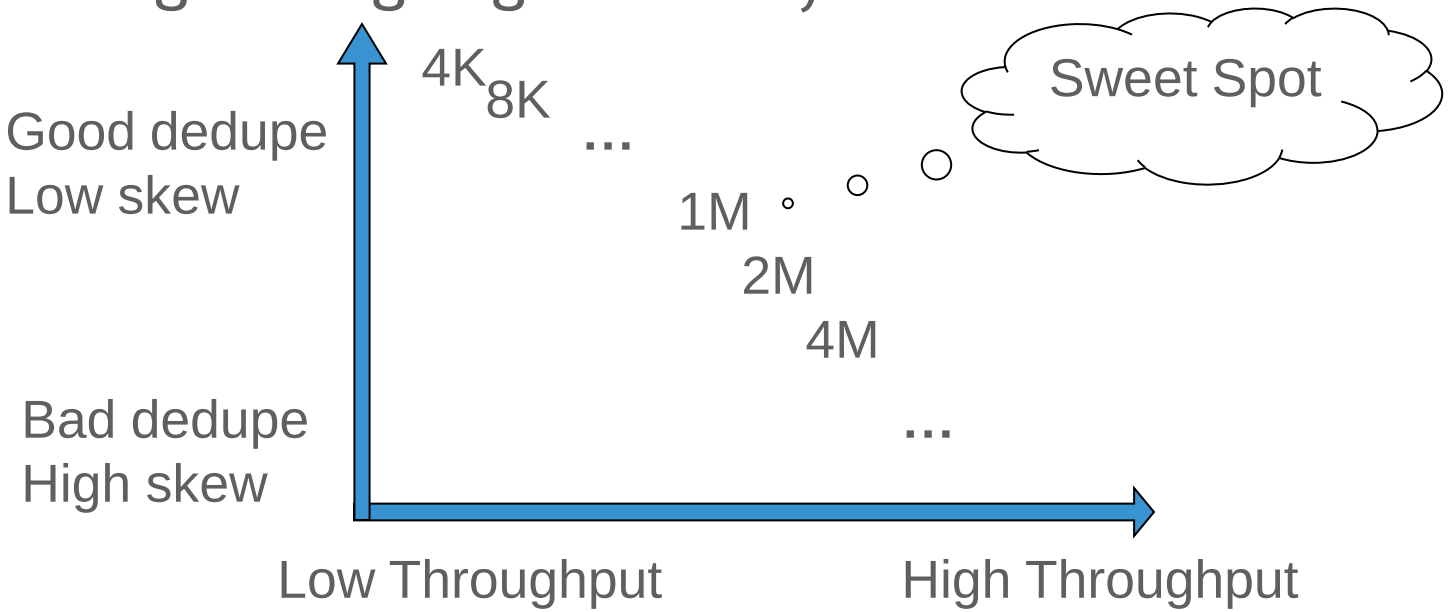
Design Questions

- Super-chunk creation: consistent formation in the face of small changes to content
  - How to group chunks into super-chunks?
  - How large should super-chunks be?
- Consistent super-chunk routing to optimize deduplication and load balance
  - What super-chunk feature to use?
  - Whether to use information on previously stored data ("stateful" routing)?

EMC² where information lives®

Super-Chunk Formation

- Anchoring (similar to chunking)
  - Produce a feature for each chunk
  - Insert a boundary when feature matches a pattern
- Choosing the right granularity



Stateless Super-Chunk Routing

Super Chunk

Hash of
- first 64 bytes "hash(64)"
- whole chunk "hash(*)"
- …

Chunk Features

20	11	38	44	82	80	4	7	46	85	20

* Real chunk features are full 32-bit integers

Super-Chunk Features

20 (first) or 4 (min)

mod N

Bin ID → Node ID (w/ Migration)

Load Balancing with Bin Migration

- Migrate when a node exceeds 105% of average disk usage
- # bins ›› # nodes, so bins are usually small
- Limitation

  If a bin becomes much larger than an average node, even migration cannot balance things out!

Stateful Super-Chunk Routing

- Goal: improve deduplication while avoiding skew
- Approach
  - Route super-chunk to node where it matches the best
  - Avoid flooding a node by matching some chunks but adding more and more new chunks
- Algorithm
  1. Count how many chunks are matched on each node
  2. Discount matches of heavily loaded nodes
  3. Route to the one that is significantly better than others
     - Or if no clear winner, fall back to stateless routing

Bin migration no longer needed!

EMC² where information lives®

Tracking Content of Each Node

- Maintain a Bloom filter for each node
  - Option 1: Backup server has all Bloom filters

    Memory overhead
  - Option 2: Use Bloom filters on each backend node

    Communication overhead
- Reduce overhead with sampling (rate = 1/8)

Evaluation Goals

- What is the best super-chunk feature & granularity?
  - Impact on deduplication
  - Impact on performance
- Is stateless or stateful routing preferable?
- How does bin migration help?

Datasets

- Trace data collected from production environment
  - 3 large backup sets with multiple data types
  - 5 smaller ones with individual data types
  - 1 synthesized by blending the 5 smaller ones

Name	Size (TB)	Dedup.	Months
Collection 1	40.7	6.1	1–2
Collection 2	44.5	11.5	4–6
Collection 3	51.6	6.1	3
Perforce	4.6	20.8	6
Workstations	4.9	5.6	6
Exchange	5.3	6.8	7
System Logs	5.4	38.7	4
Home Dirs.	12.9	19.3	3
Blended	33.1	12.5	N/A

EMC² where information lives®

Evaluation Metrics

- Capacity (local compression ignored)

$$\text{total dedupe} = \frac{\text{original size}}{\text{dedupe size}}$$

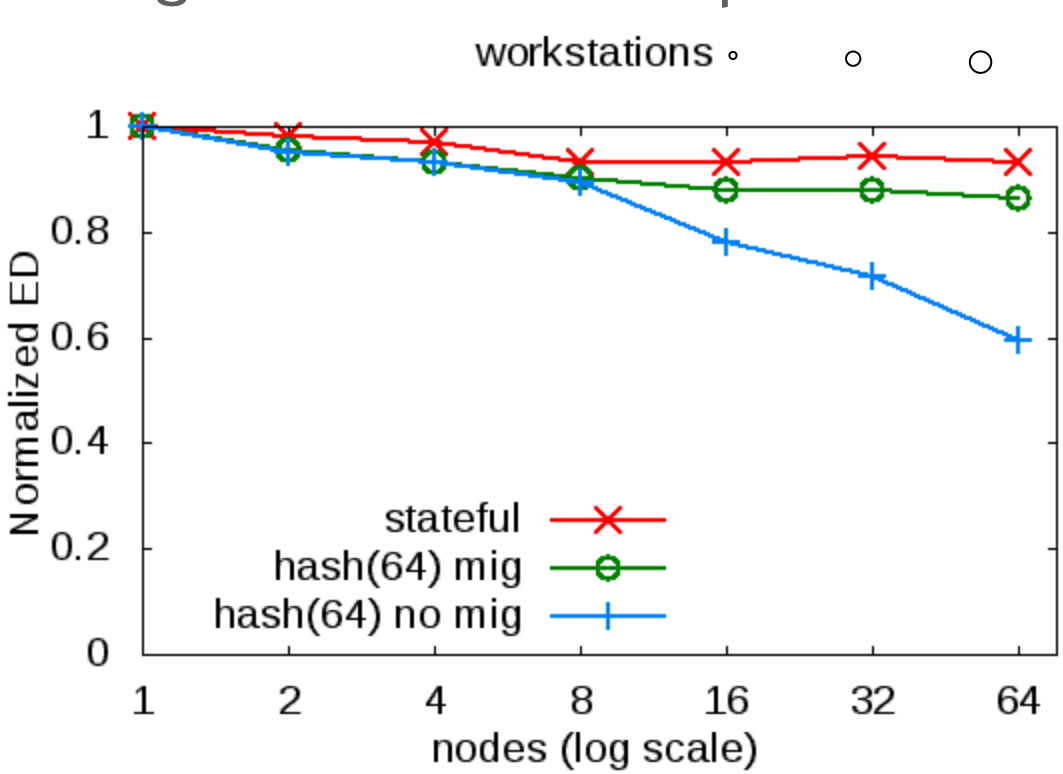
$$\text{data skew} = \frac{\text{max node utilization}}{\text{avg node utilization}}$$

$$\text{effective dedupe (ED)} = \frac{\text{total dedupe}}{\text{data skew}}$$

  - ED normalized for easy comparison
- Throughput
  - # of on-disk fingerprint index lookups

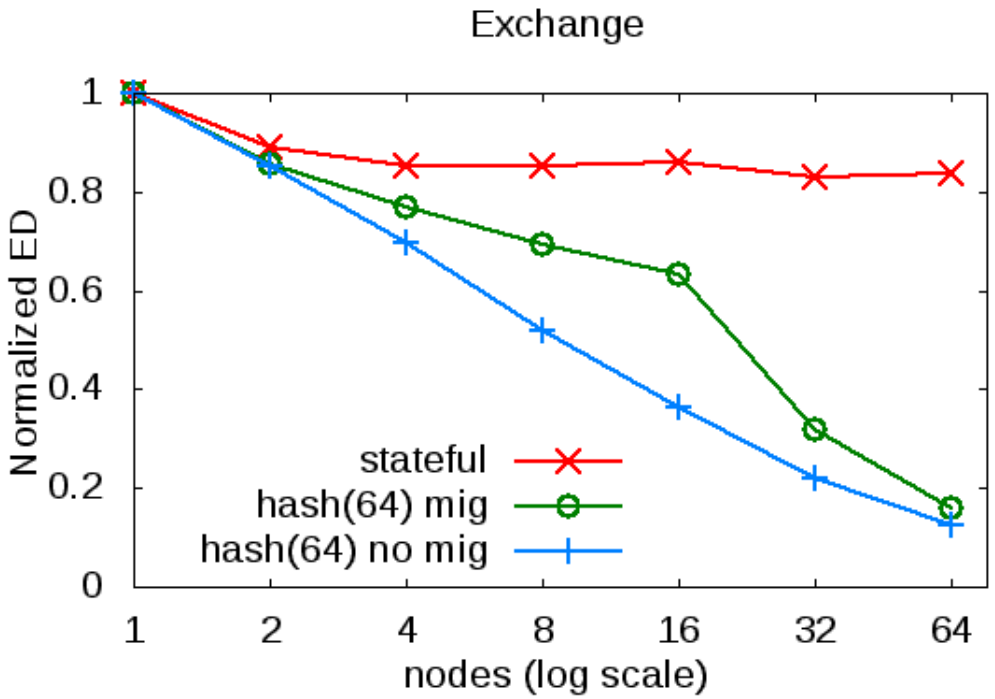
Overall Effectiveness

- Hash(64) works well for small clusters
- Bin migration helps to reduce skew and increase ED
- Stateful routing offers further improvements

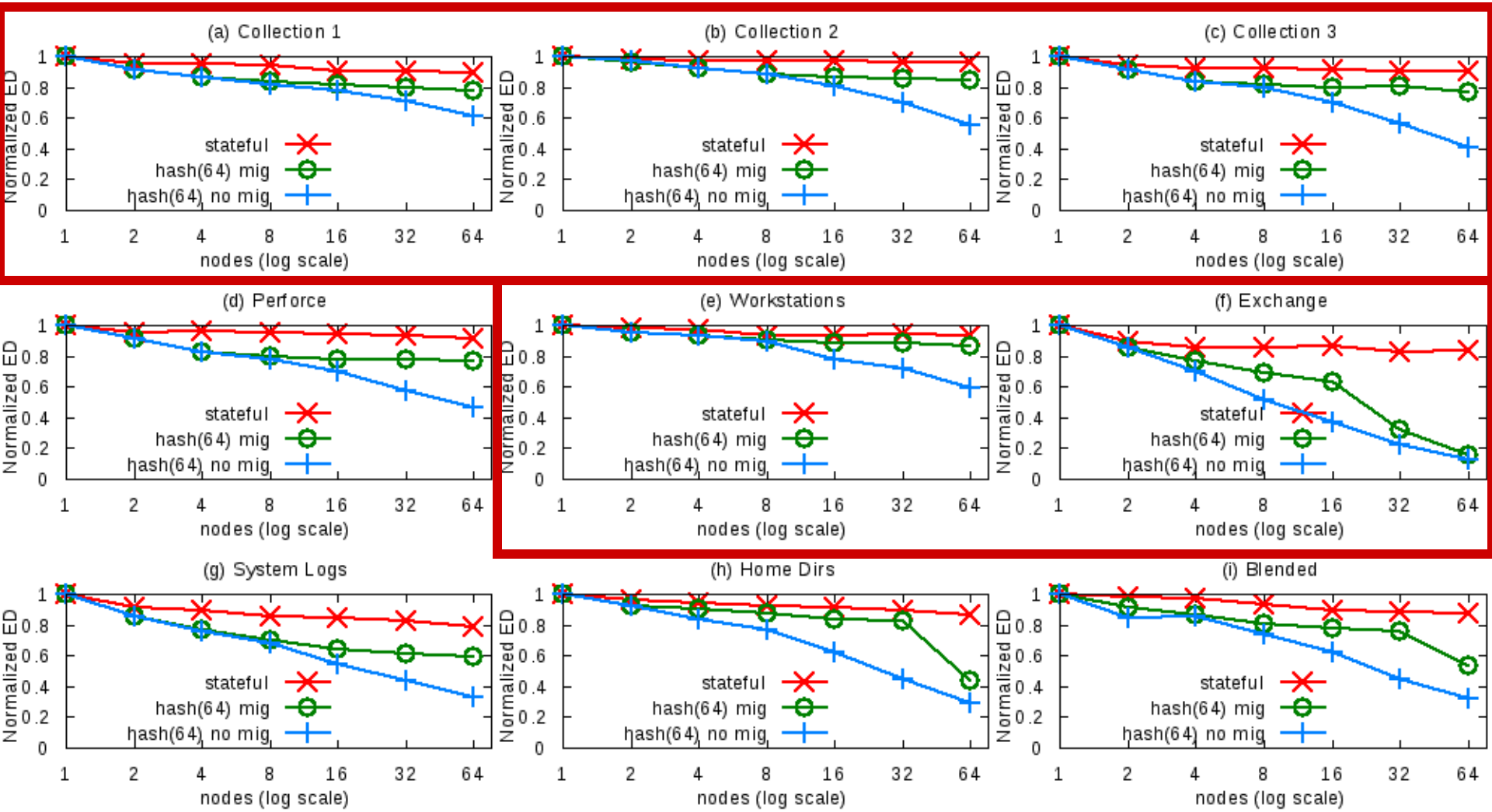


Exchange: Anomalous Dataset

- A frequent pattern in data happens to qualify as super-chunk boundary and routing feature
- A bin becomes much larger than an average node

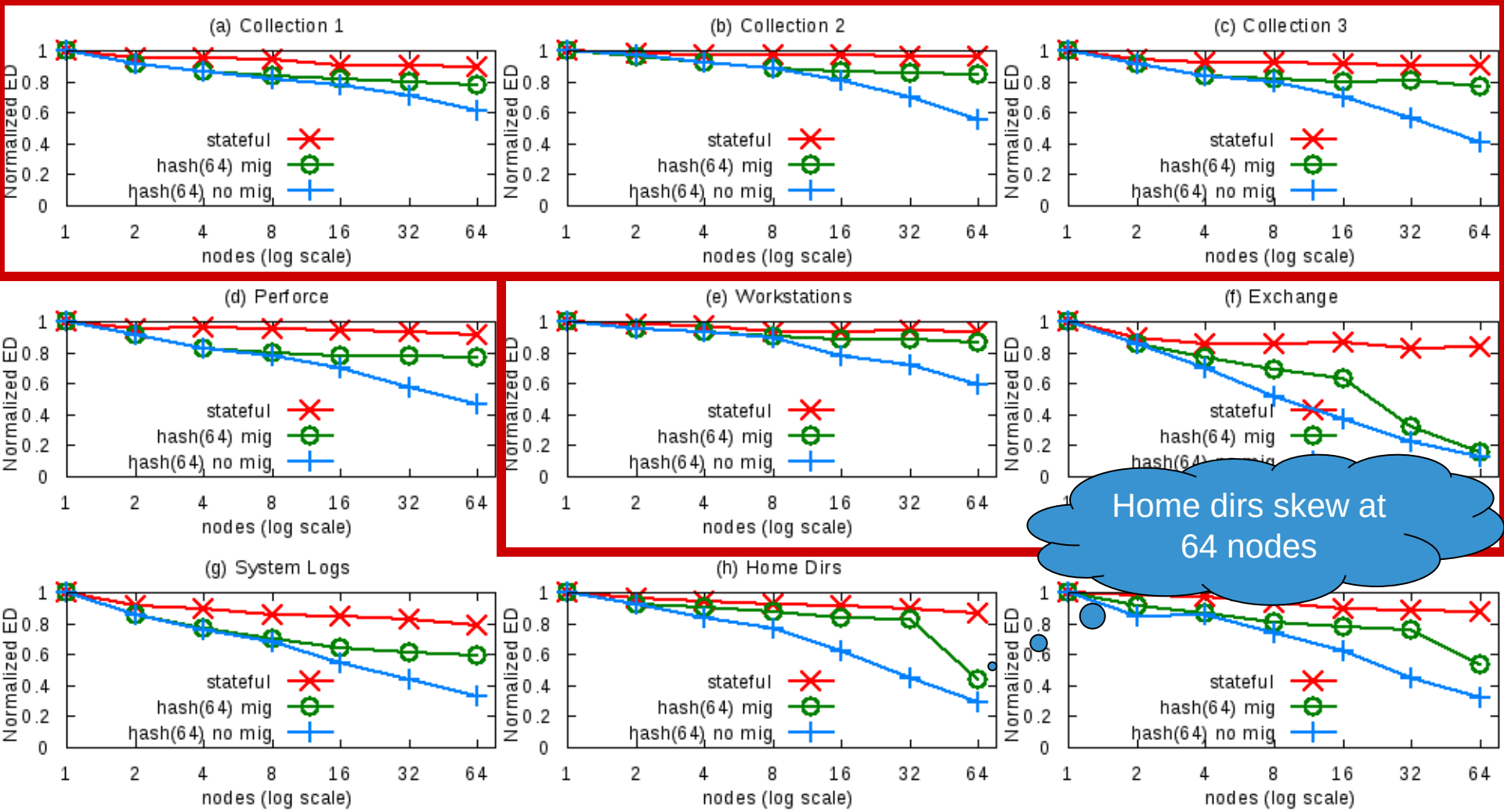


Overall Effectiveness



Overall Effectiveness

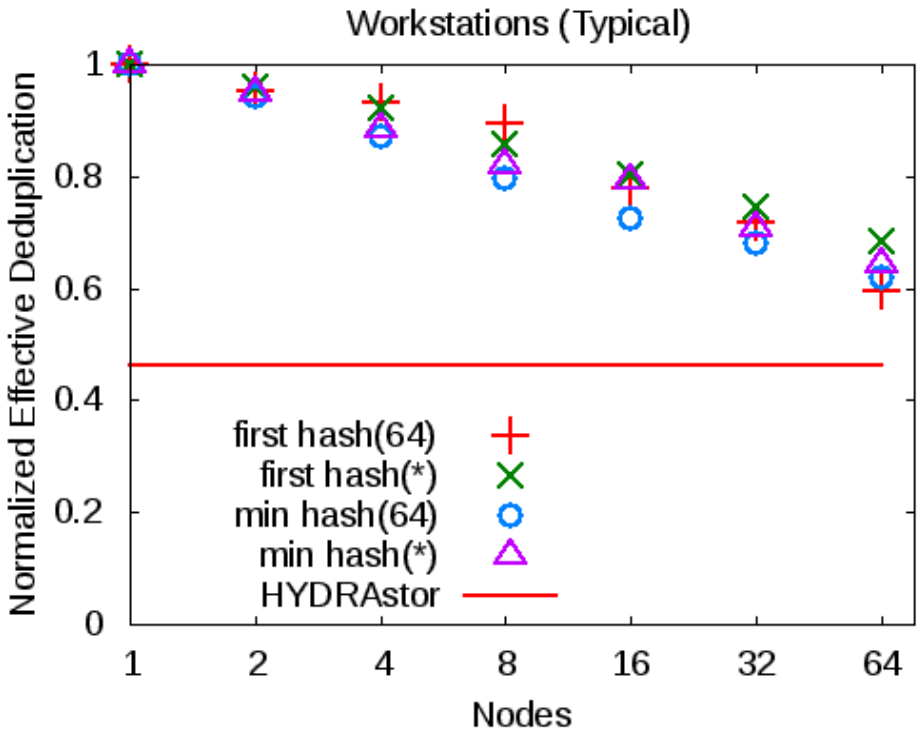
"real-world" deployments



Home dirs skew at 64 nodes

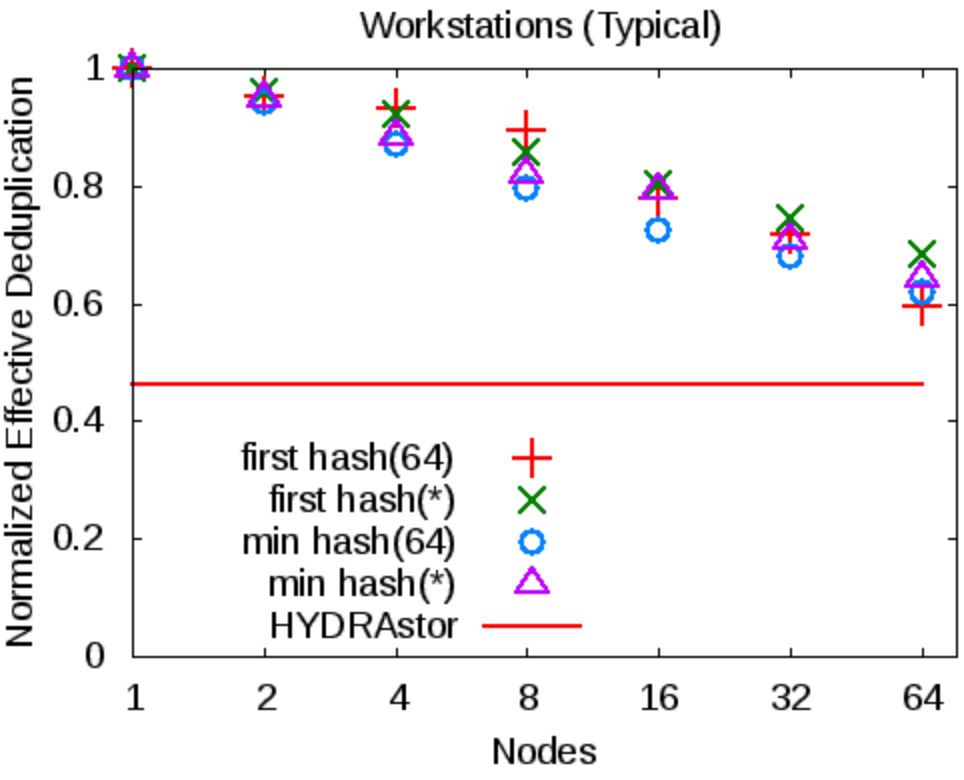
Feature Selection

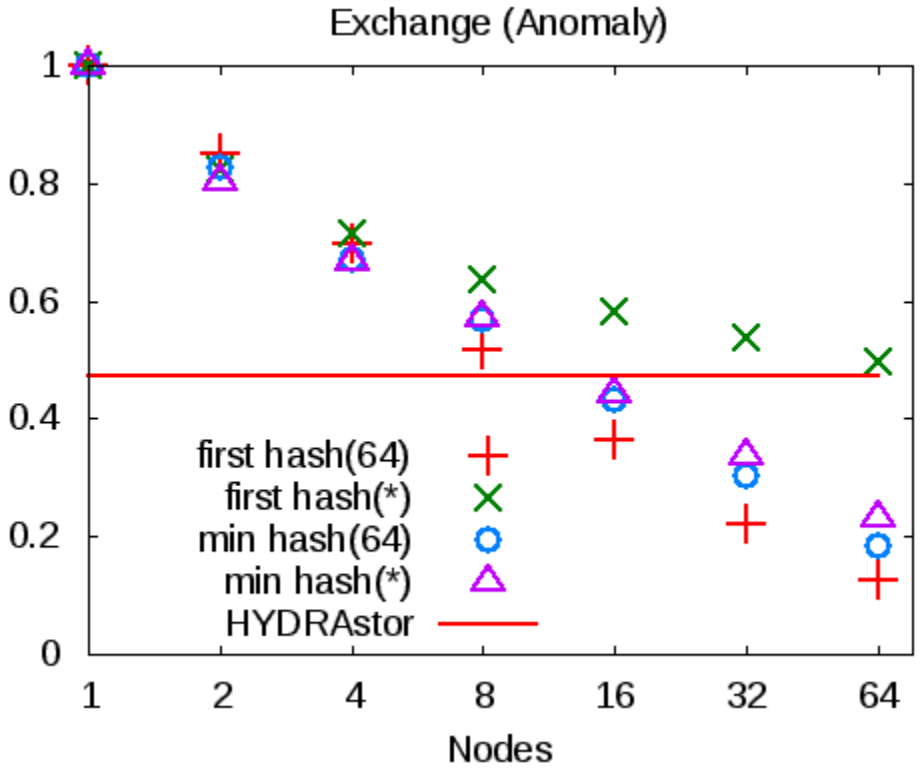
- All super-chunk routing methods are similarly effective
  - First of Hash(64) is preferred due to simplicity
  - HYDRAstor: routing 64K chunks (w/o super-chunks)



Feature Selection

- All super-chunk routing methods are similarly effective
  - First of Hash(64) is preferred due to simplicity
  - HYDRAstor: routing 64K chunks (w/o super-chunks)
  - Exchange is the extreme case with large clusters

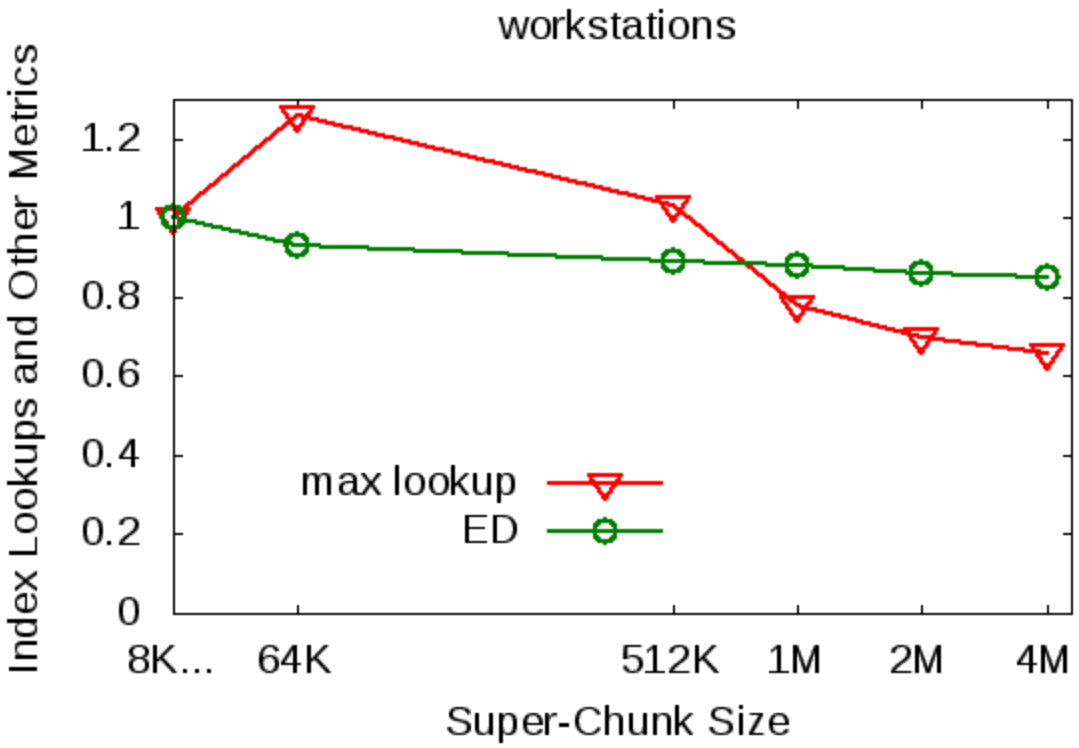




Cache Locality and Throughput

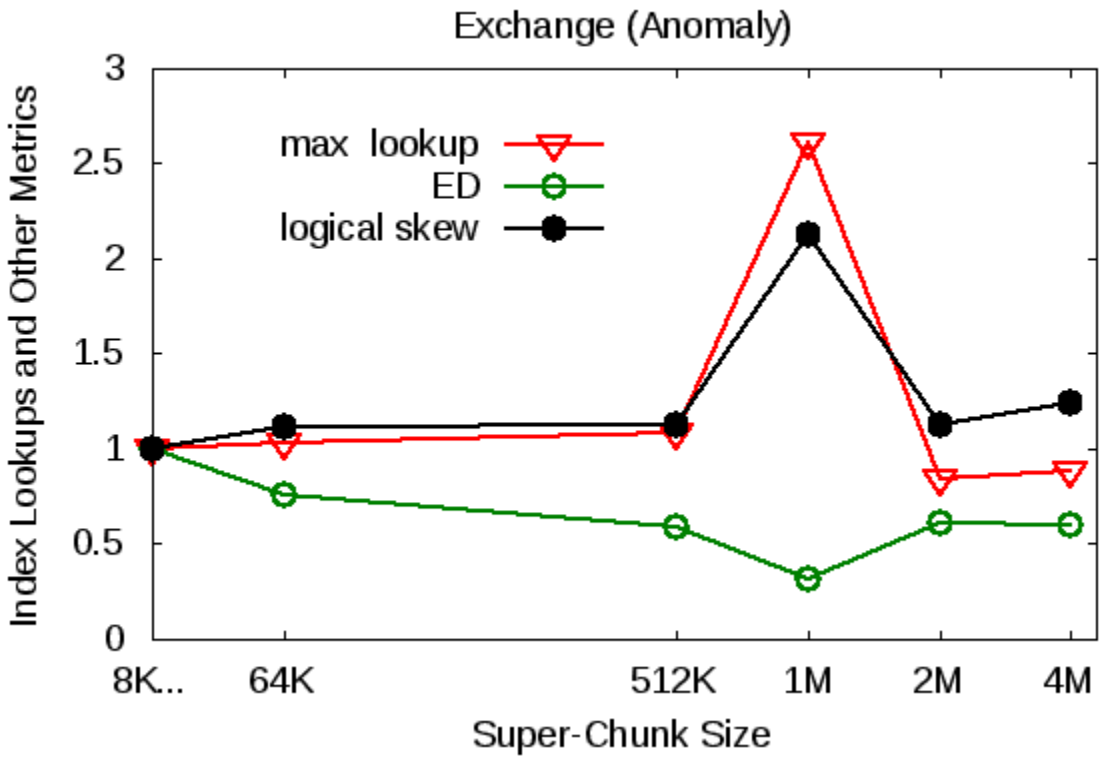
- Throughput measured by max # fingerprint lookup
  - Lower value suggests higher throughput
- Larger super-chunk ➔ better throughput

Throughput studies performed @ 32 nodes



Cache Locality and Throughput

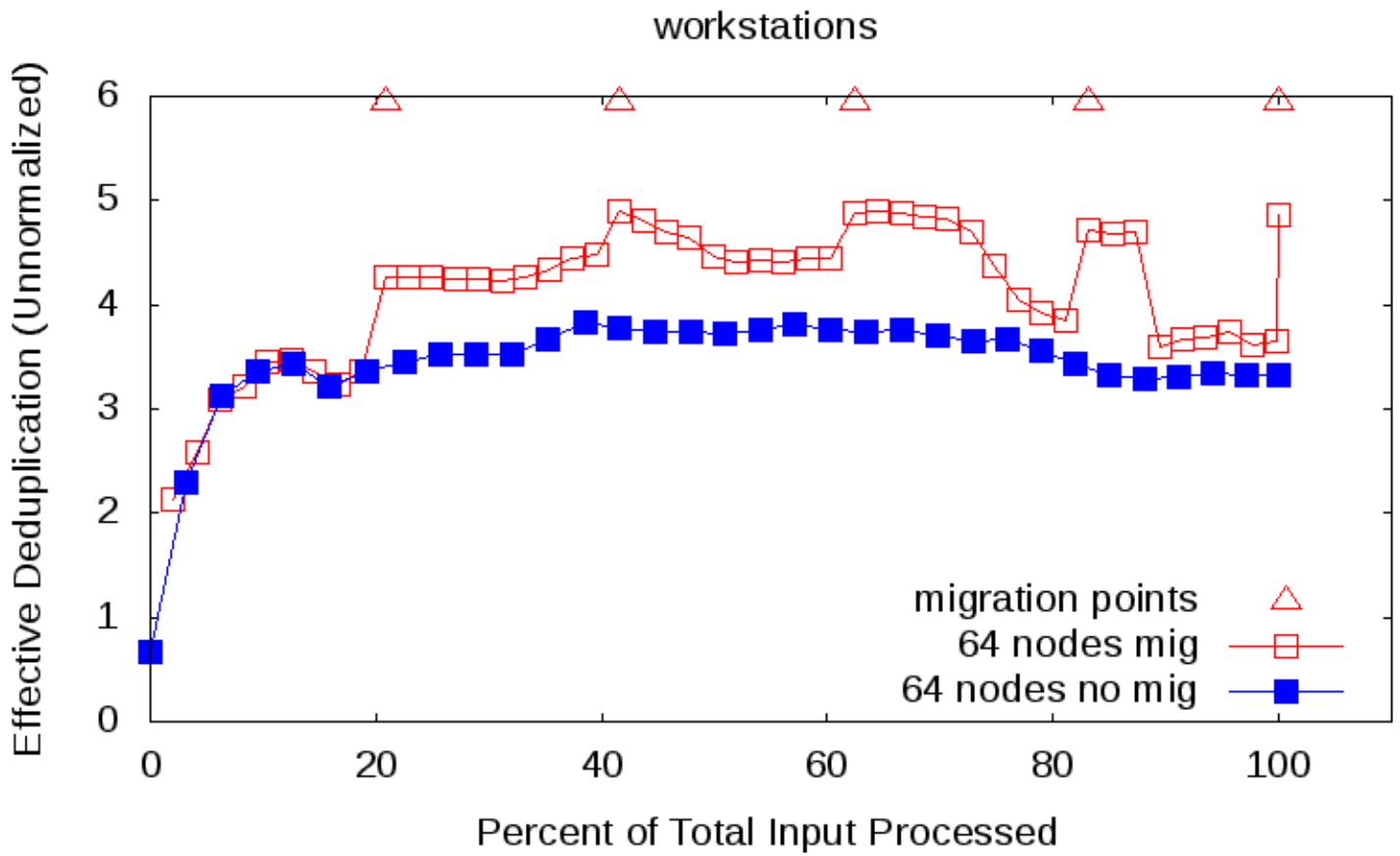
- Exchange: anomaly again…



Logical Skew: max(size before dedupe) / avg (size before dedupe)

Effect of Bin Migration

- Deduplication improves after migration but may drop between migration intervals



Product: Global Deduplication Array

- Two-node configuration, each with
  - 2x6 cores @ 2.8GHz, 96 GB memory, 10-Gb Ethernet
  - 285 TB storage in 12 shelves, 2+2 RAID-6
- Performance numbers
  - Logical capacity: 11.4PB (with 20x dedupe)
  - Write throughput: 7.3 GB/s
  - Prototype scaled performance linearly from 1-4 nodes

Recently upgraded, different from paper

EMC² where information lives®

Summary

- Super-chunk routing for building deduplication clusters
  - Scalable throughput
  - Maximizing deduplication

EMC² where information lives®

Summary

	Stateless		Stateful
	Small clusters	Large clusters	All
Deduplication			
Data Skew			
Migration Cost			
Overhead			

Future Work

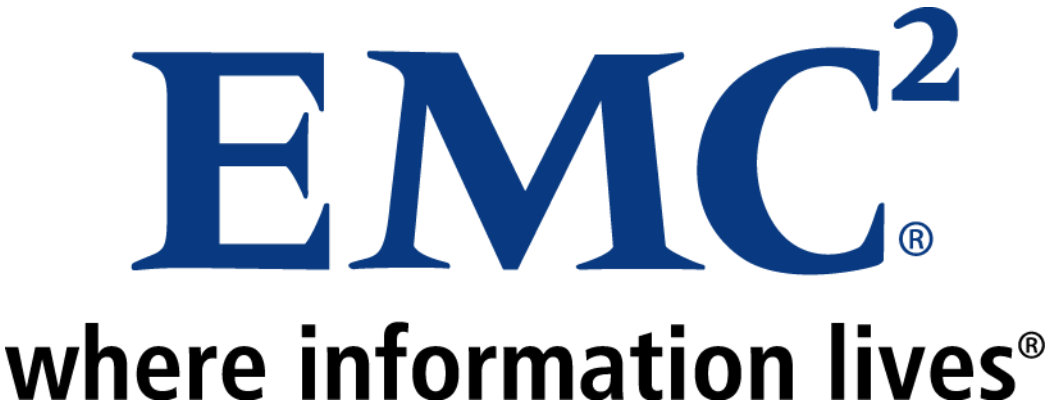
- Investigate conditions causing data skew
- Examine scalability over broad range of clusters
- Support cluster reconfiguration with bin migration
- Build a prototype cluster with stateful routing

EMC² where information lives®

Acknowledgments

- Thanks to the many engineers in EMC Data Domain who created the GDA product
- Thanks to our shepherd, Cristian Ungureanu, for his extensive comments, and the many others who provided feedback or helped with experiments: Dhanabal Ekambaram, Paul Jardetzky, Ed Lee, Dheer Moghe, Naveen Rastogi, Pratap Singh, and Grant Wallace

EMC² where information lives®


EMC²
where information lives®