MAD2: A Scalable High-Throughput Exact Deduplication Approach for Network Backup Services

Jiansheng Wei†, Hong Jiang‡, Ke Zhou†, Dan Feng†

†School of Computer, Huazhong University of Science and Technology, Wuhan, China
Wuhan National Laboratory for Optoelectronics, Wuhan, China
‡Dept. of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE, USA

Target Application

- backup storage systems
- network backup services
  - decentralized peer-to-peer schemes based on peer-cooperation over distributed network
    - trade local resources for remote storage capacity
  - **centralized storage provided by storage service providers (SSPs)**
    - trade money for reliable backup and provide better quality-of-service (QoS)

Build a Scalable and Cost-effective Backup/Archiving Storage System

- deduplication technology has been widely applied in disk-based secondary storage systems
- two technical challenges
  - *duplicate-lookup disk bottleneck*
    - determine if an incoming data object is a duplicate, the index can become too large for RAM to hold in its entirety
  - *storage node island effect*
    - eliminate duplicates among multiple servers

Find the Key

- ***Duplicate Detection Methods***
  - **examine the file system metadata**
  - **adopt content-based fingerprint**
    - **Granularity: whole files, fixed-size blocks, or variable-sized chunks**
- ***Duplicate Lookup Acceleration Methods***
  - **exploit data locality – DDFS, Sparse Indexing**
  - **exploit file similarity – Extreme Binning**

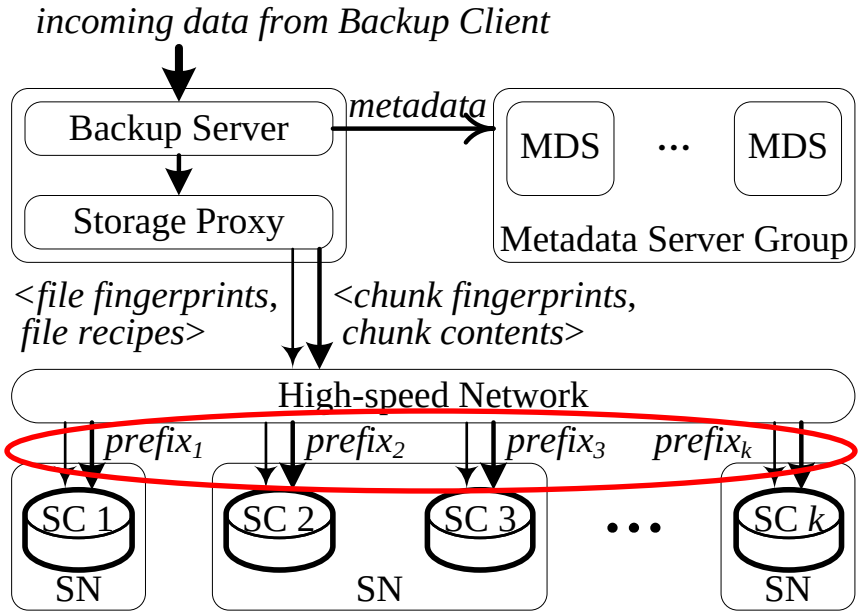
  (fast membership determination of incoming data objects)
- ***Enable Scalability***
  - **distributed hash table – Extreme Binning, HYDRAstor**

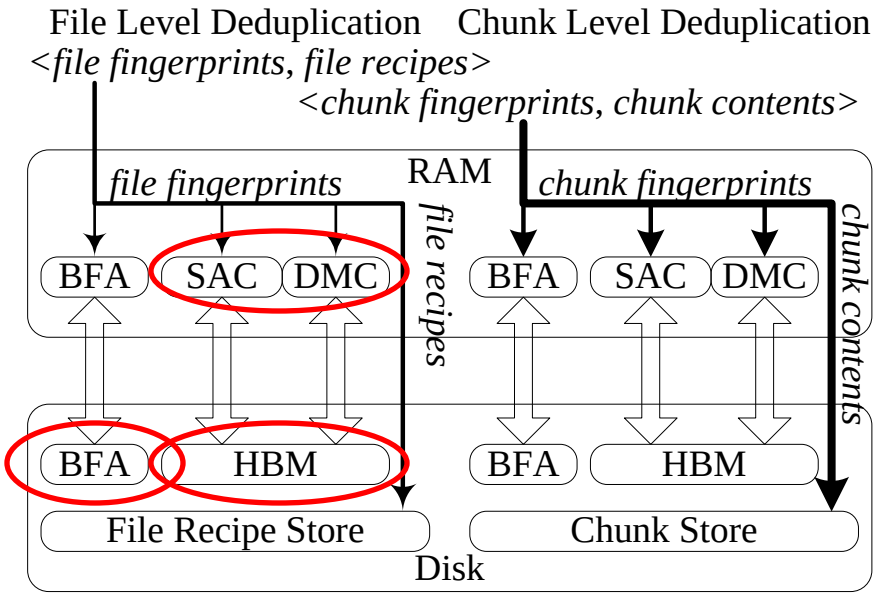
  (partition data into dissimilar or less similar groups)

Outline

- Background and Motivation
- The MAD2 Architecture and Design
- Prototype Implementation and Evaluation
- Conclusions, Questions

The MAD2 Architecture

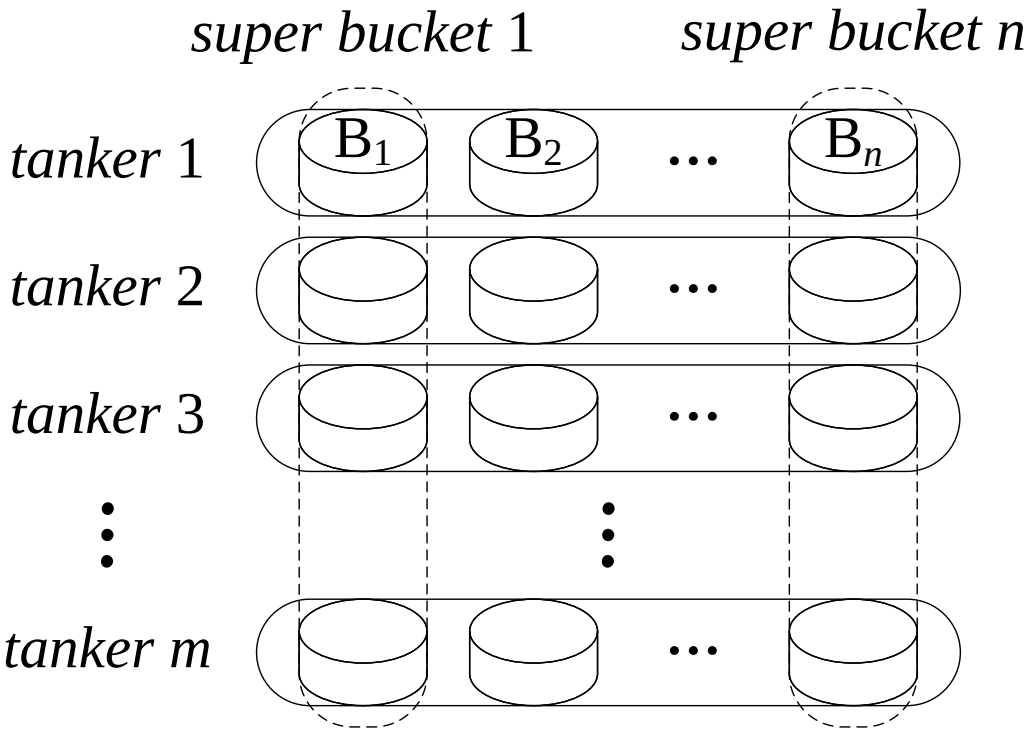




- **Hash Bucket *M*atrix (HBM)**
- **Bloom Filter *A*rray (BFA)**
- ***D*ual Cache**
- ***D*HT-based Load-Balance**

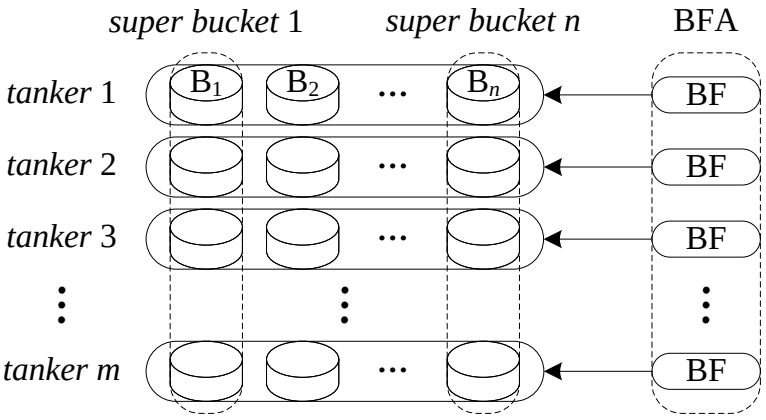
Locality-Preserved Hash Bucket Matrix

- Consecutive fingerprints belonging to the same backup job have a high probability of being stored in the same tanker.



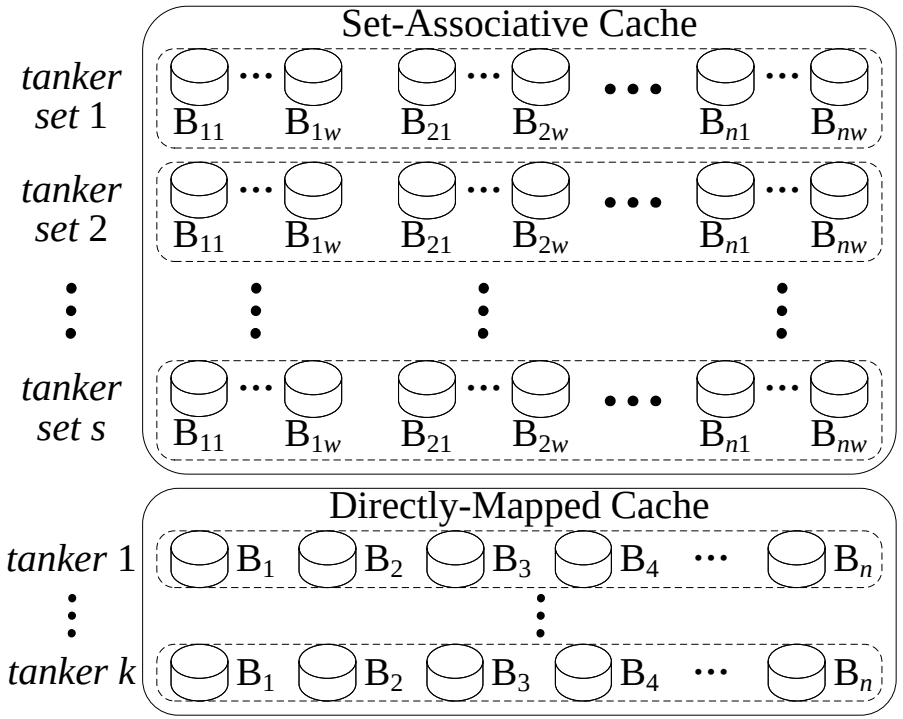
Using Bloom Filter Array as Quick Index

- Single Bloom Filter - Drawbacks
  - potential total number of fingerprints is difficult to estimate.
  - a single BF is ineffective in locating possible duplicates.
  - physical fingerprint deletion will result in rebuilding of the whole BF.
- Employ a Bloom Filter Array (BFA)
  - associate each tanker with a Bloom Filter, add a Bloom Filter along with a new tanker.
  - all the Bloom Filters are isomorphic and share the same hash functions.
  - add a Bloom Filter along with a new tanker.



Dual Cache Mechanism

- *Dual Cache* is designed to improve disk access efficiency while locating duplicate fingerprints.
  - *directly-mapped cache* (DMC)
    - capture the fingerprint locality in backup streams
    - periodically rebalance the hash bucket matrix
  - *set-associative cache* (SAC)
    - exploit the fingerprint locality in backup data



DHT-based Load Balancing

- Each SC is only responsible for file recipes and chunks with the same specific fingerprint prefix.
  - Because fingerprints with different prefixes are collision free, and if each SC performs exact deduplication in its responsible hash sub-space, the entire backend storage can achieve global exact deduplication.

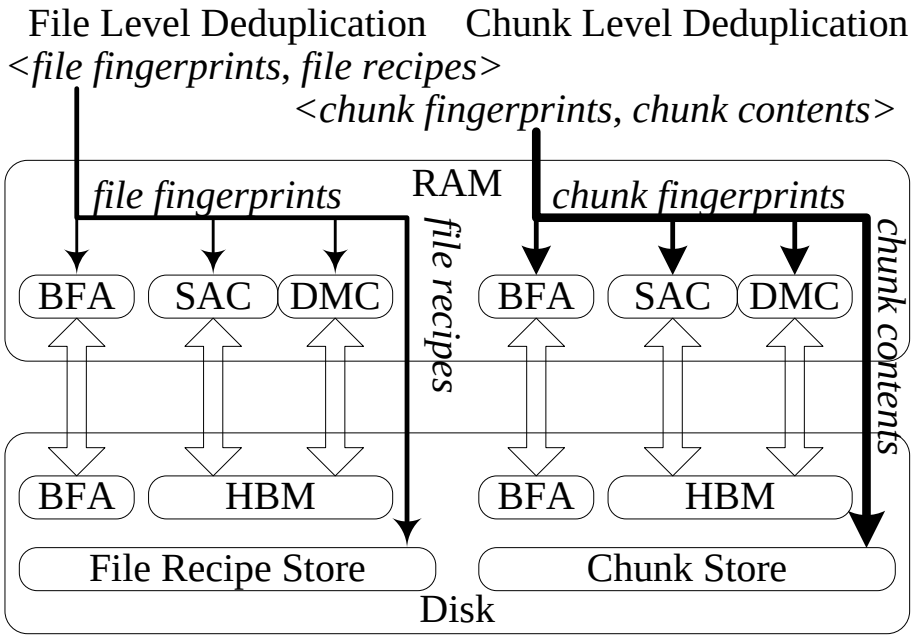
- Both file recipes and chunk contents will be distributed in their backup sequences to preserve locality.
  - Consider a sequence of fingerprints with two different prefixes (*a*1, *b*0, *c*1, *d*1, *e*0, *f*1, *g*0). MAD2 divides them into two sub-sequences (*a*1, *c*1, *d*1, *f*1) and (*b*0, *e*0, *g*0), and distributes each sub-sequence to one responsible SC.

Data Organization and Deletion Support

- all the chunk contents are kept in *chunk store*, which consists of *chunk tankers* corresponding to *tankers* in HBM.
- inside each chunk tanker, chunks are grouped and packaged into *chunk containers* in a stream-locality-preserved manner.
- *counting fingerprint:* structured as <*fingerprint*, *data length*, *reference count*>
  - a file or a chunk will not be physically deleted until the associated reference count drops to *zero*.
  - adjacent tankers can be merged if they are sparse enough.
  - all the involved Bloom Filters will be reconstructed.
  - a physical delete operation is executed in a batch mode. all involved tankers must be changed to the read-only mode to maintain data consistency.
  - exposes only a file-level delete interface to SC clients.

Workflow of the MAD2 Approach

- two phases:
  - ***eliminate duplicate files***.
  - ***eliminate duplicate chunks***.
- two inline deduplication modes:
  - ***exclusive mode***: targets at high-speed backup streams that can finish data transmission in short time windows.
  - ***round-robin mode***: aims at low-speed backup streams that will be buffered by SP (storage proxy).



Outline

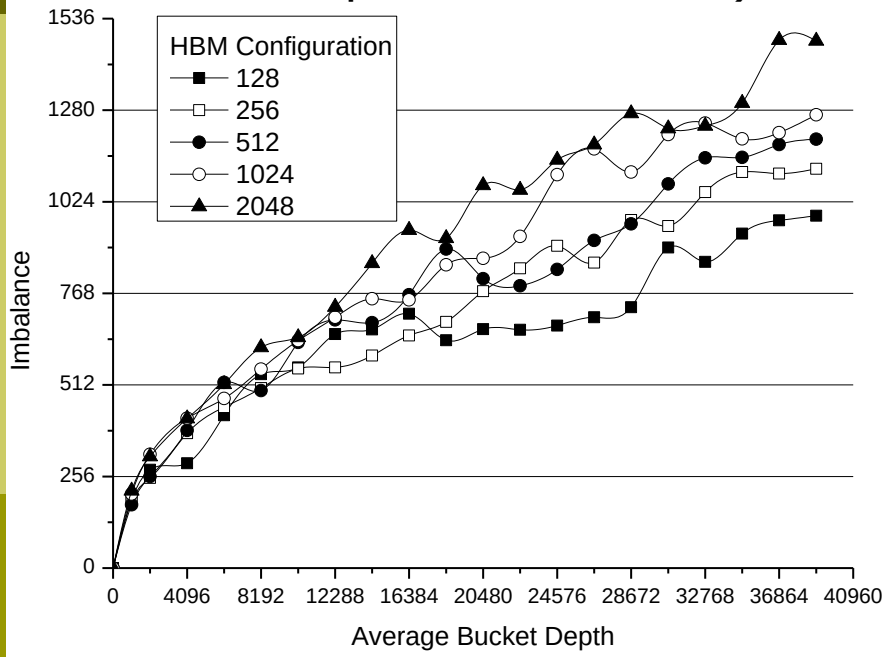
- Background and Motivation
- The MAD2 Architecture and Design
- Prototype Implementation and Evaluation
- Conclusions, Questions

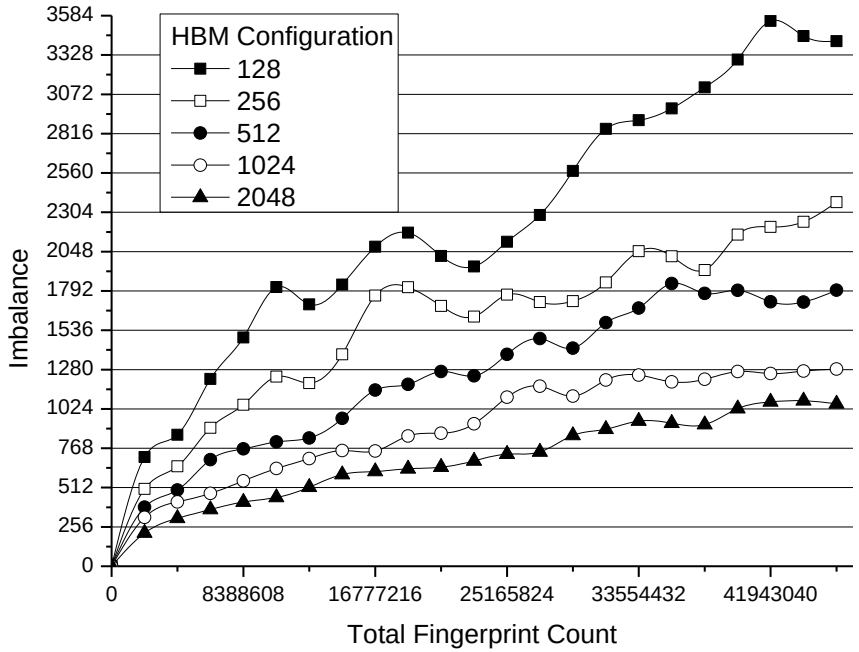
Experiment Datasets

- *Workgroup* set
  - collected from an engineering group consisting of 15 graduate students
  - 12.1 million files, 6.0TB data
- *Campus* set
  - collected from 26 users on a campus network, including personal website owners, small file transfer site managers and other individuals.
  - 15.4 million files, 4.7TB data

Locality-Preserving Capability of HBM

- Let each super bucket consist of only one bucket, we examine five different configurations of HBM (i.e., 128-, 256-, 512-, 1024-, and 2048-super-bucket HBM)





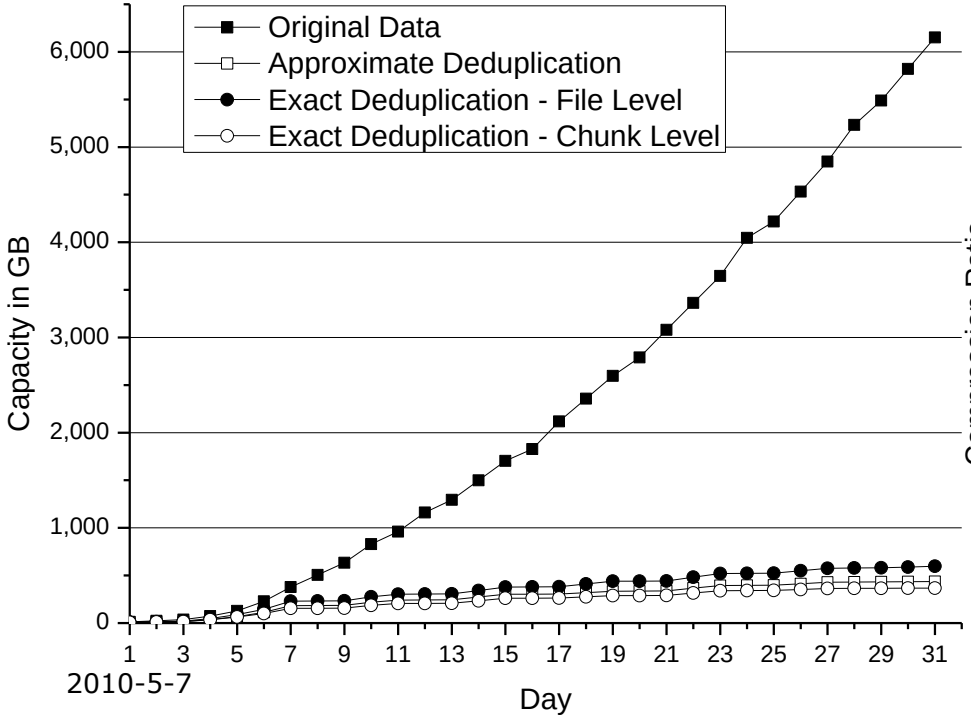
- Tanker Structure: 1,024 buckets plus 1,024 fingerprint cells
- Rebalancing Threshold: 1,024

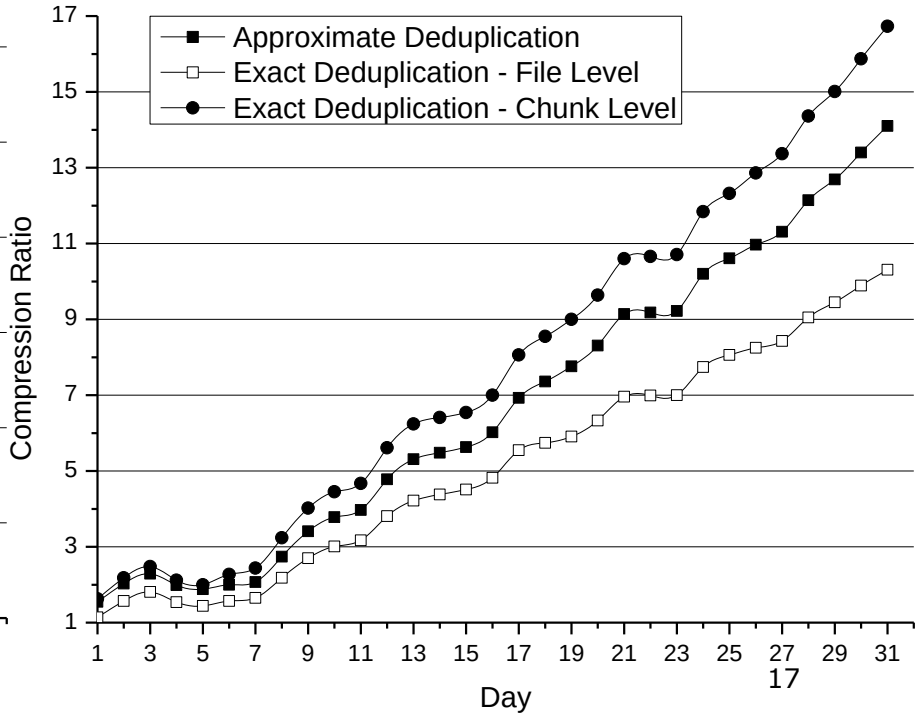
Hot Fingerprint

- *detected 84,876,504 duplicate chunks with the same content of 1,024-byte zeros*.
- *zero-chunks* may be widely shared even among dissimilar files.
  - can disrupt the chunk locality and affect the efficiency of our cache mechanism.
  - pre-calculate the SHA-1 hash of 1KB *zero-chunk*, and define it as a built-in fingerprint.

Deduplication Efficiency

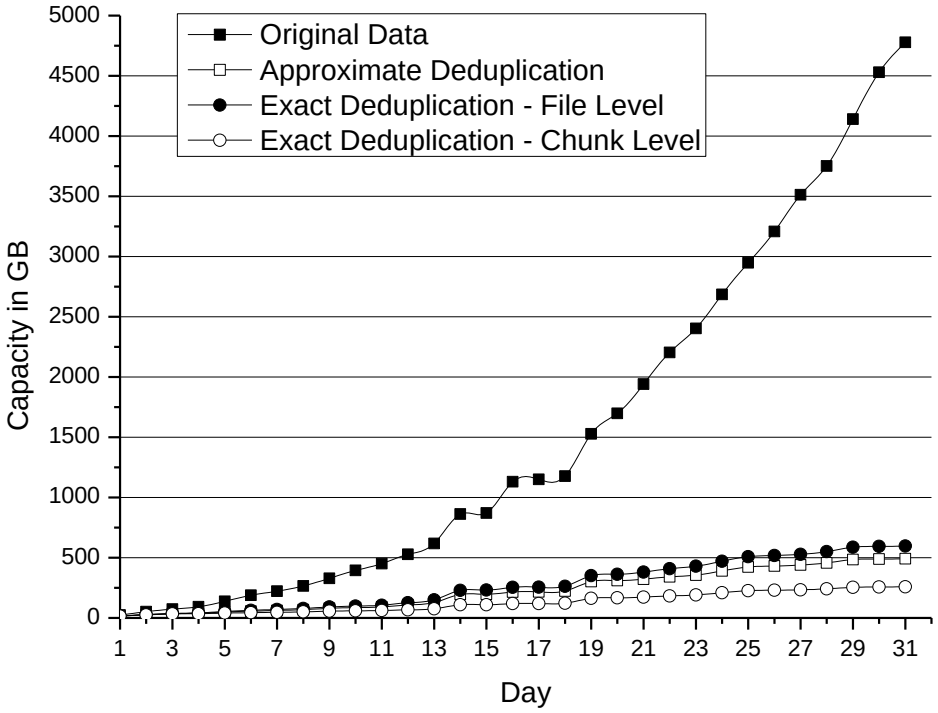
- Workgroup Set
  - implemented a simple version of Extreme Binning to represent *approximate deduplication*.

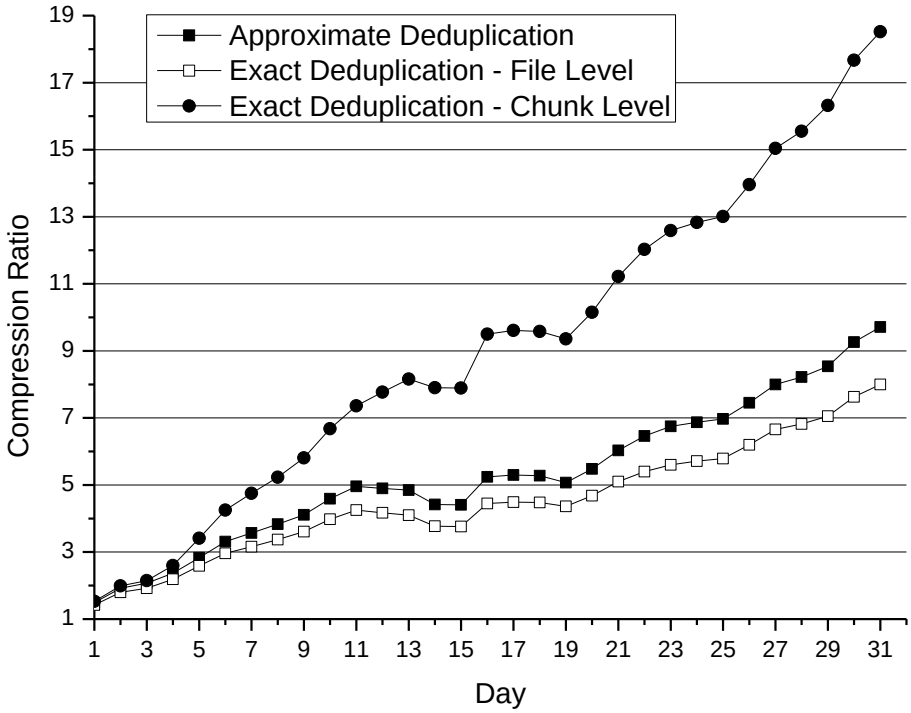




Deduplication Efficiency

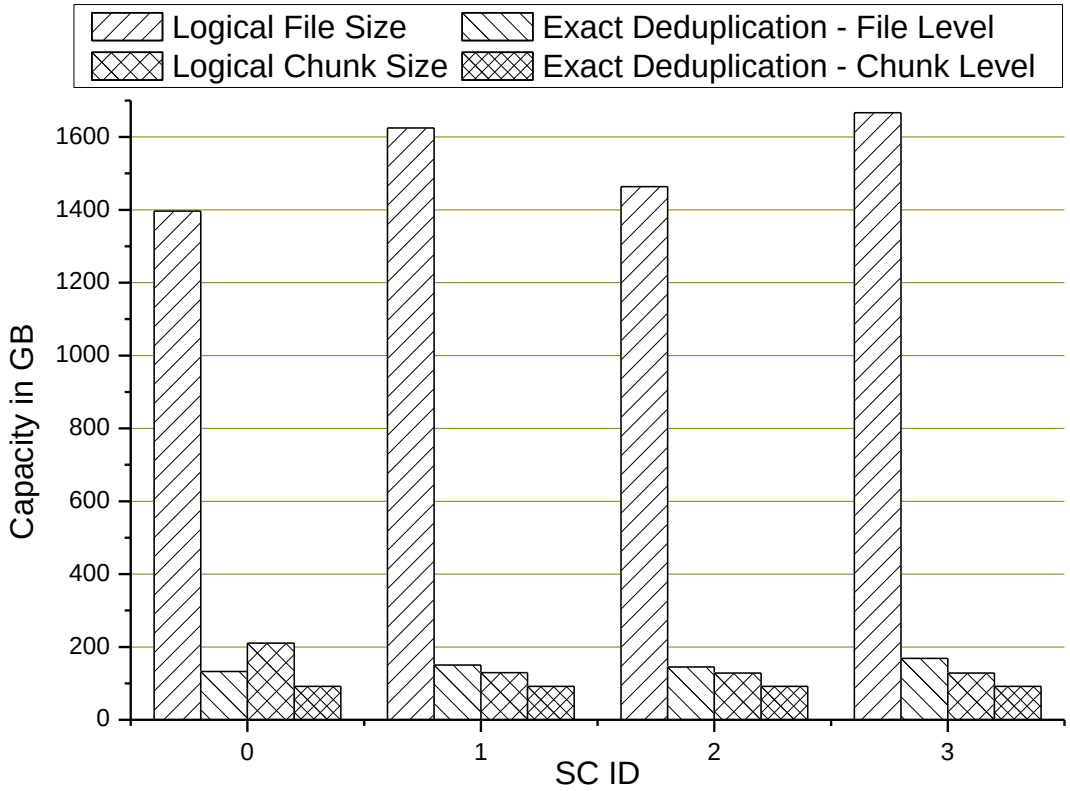
- Campus Set





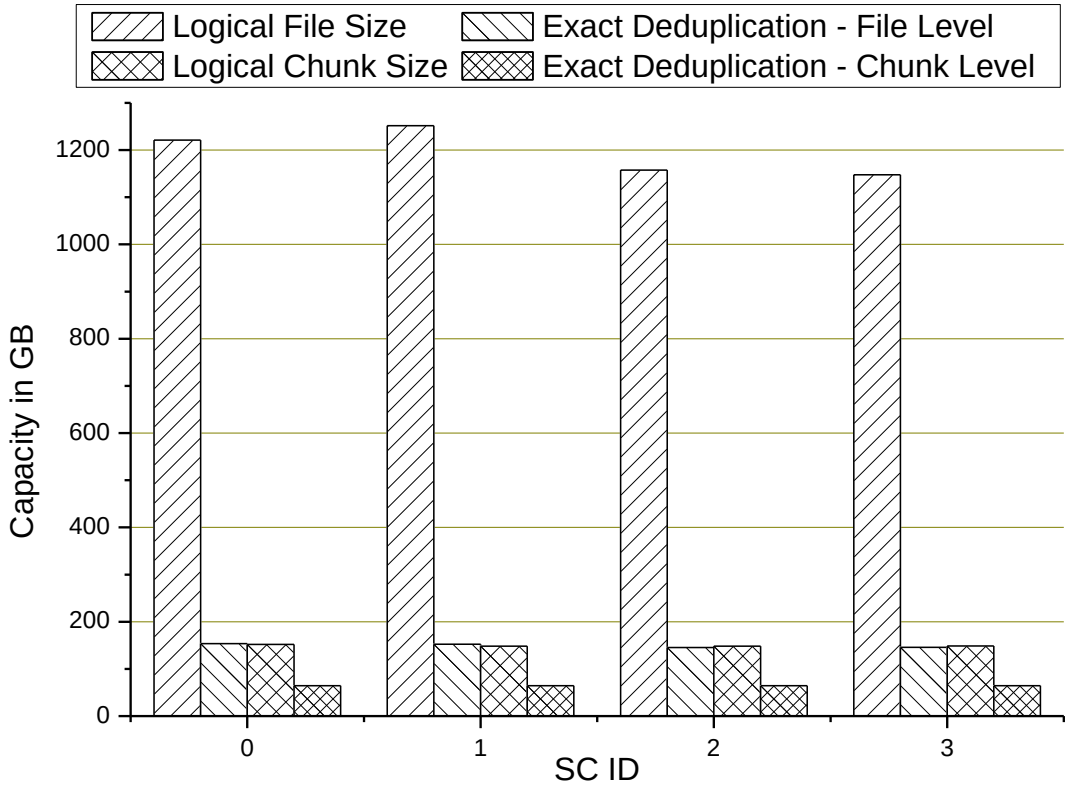
Load Balancing

- Workgroup Set
  - 84,876,504 hot fingerprints were detected at the chunk level, which means that there are about 80.9GB *zero-chunks* being distributed among files.



Load Balancing

- Campus Set
  - A total of 3,953,486 hot fingerprints are detected in the *Campus* set, corresponding to approximately 3.8GB *zero-chunks*.



Throughput

- **We trace and report the fingerprint deduplication efficiency.**
- Considering an average chunk size of 4KB, 25,600 chunk fingerprints must be deduplicated per second to achieve a 100MB/s raw deduplication throughput.
  - Note that one duplicate fingerprint found at the file level means that all the chunk fingerprints belonging to that file can be directly skipped.
- *Workgroup* set: 12,154,807 file fingerprints and 207,856,782 chunk fingerprints are actually transferred and deduplicated in a period of 982 seconds.
  - Chunk level: 211,667 fingerprints/sec, 827MB/s
  - Overall: **6,415MB/s**
- *Campus* set: 15,391,112 file fingerprints and 132,110,642 chunk fingerprints are actually transferred and deduplicated in a period of 814 seconds.
  - Chunk level: 162,298 fingerprints/sec, 634MB/s
  - Overall: **6,011MB/s**

RAM Usage

- 10TB ***deduplicated*** data set – 10GB RAM
  - 40×2^{20} files, assuming the average file size is 256KB
  - 2.5×2^{30} chunks, assuming the average chunk size is 4KB
  - Assuming a capacity of 2^{20} fingerprints
    - 40 tankers to hold the file fingerprints
    - 2,560 tankers to hold the chunk fingerprints.
  - limiting the false positive rate of Bloom Filter Array (BFA) to an extremely low level of $1/2^{20}$
    - 144MB to hold the file-level BFA
    - 9.2GB to hold the chunk-level BFA
    - 800MB to construct the in-memory cache

Minimizing the RAM Consumption

- Increase the average chunk size.
  - at the expense of less detectable duplicate data.
- Allow a much higher false positive rate.
  - by increasing the false positive rate from 1/220 to 1/29, the total RAM consumption by BFA will be reduced from 9.2GB to 4.2GB.
  - cause more cache replacement operations and affect the throughput.
- Configures the chunk-level deduplication to run on a round-robin manner among multiple SCs on the same storage node.
  - with *n* SCs rotating to execute the chunk-level deduplication one at a time on a round-robin basis, the memory requirement will be reduced to approximate 1/*n.*
  - at the cost of reduced chunk-level deduplication throughput.

Outline

- Background and Motivation
- The MAD2 Architecture and Design
- Prototype Implementation and Evaluation
- Conclusions, Questions

Conclusions, Questions

- Organizes fingerprints into a **Hash Bucket *M*atrix (HBM)**, whose rows can be used to **preserve** the data locality in backups.
- Uses **Bloom Filter *A*rray (BFA)** as a quick index to quickly **identify** non-duplicate incoming data objects or indicate where to find a possible duplicate.
- Integrates in-memory ***D*ual Cache** to **capture** and **exploit** locality.
- Employs a ***D*HT-based Load-Balance** technique to evenly **distribute** data objects among multiple storage nodes in their backup sequences to further enhance performance with a well-balanced load.
- Experimental results show that the MAD2 approach is effective and efficient.

Thank you! Questions?