Semi-hierarchical Semantic-aware Storage Architecture

Yu Hua
Huazhong University of Science and Technology

https://csyhua.github.io

Current and Future Storage

S M I L E

Current and Future Storage 1

- SMILE
- Scale : Big Data , Big Storage

Current and Future Storage 2

- SMILE
- NN(M)-Intelligent :

Current and Future Storage 3

- SMILE
- Integrated :

➢ Near Data Processing :

➢ Processing in-memory (PIM)

➢ In-storage computing (ISC)

➢ Quantx(Micron), Optane(Intel), NDP(HUAWEI), ……

Current and Future Storage 4

- SMILE
- Long-term ：

➢ Storage media and runtime context

➢ Time-sensitive and value

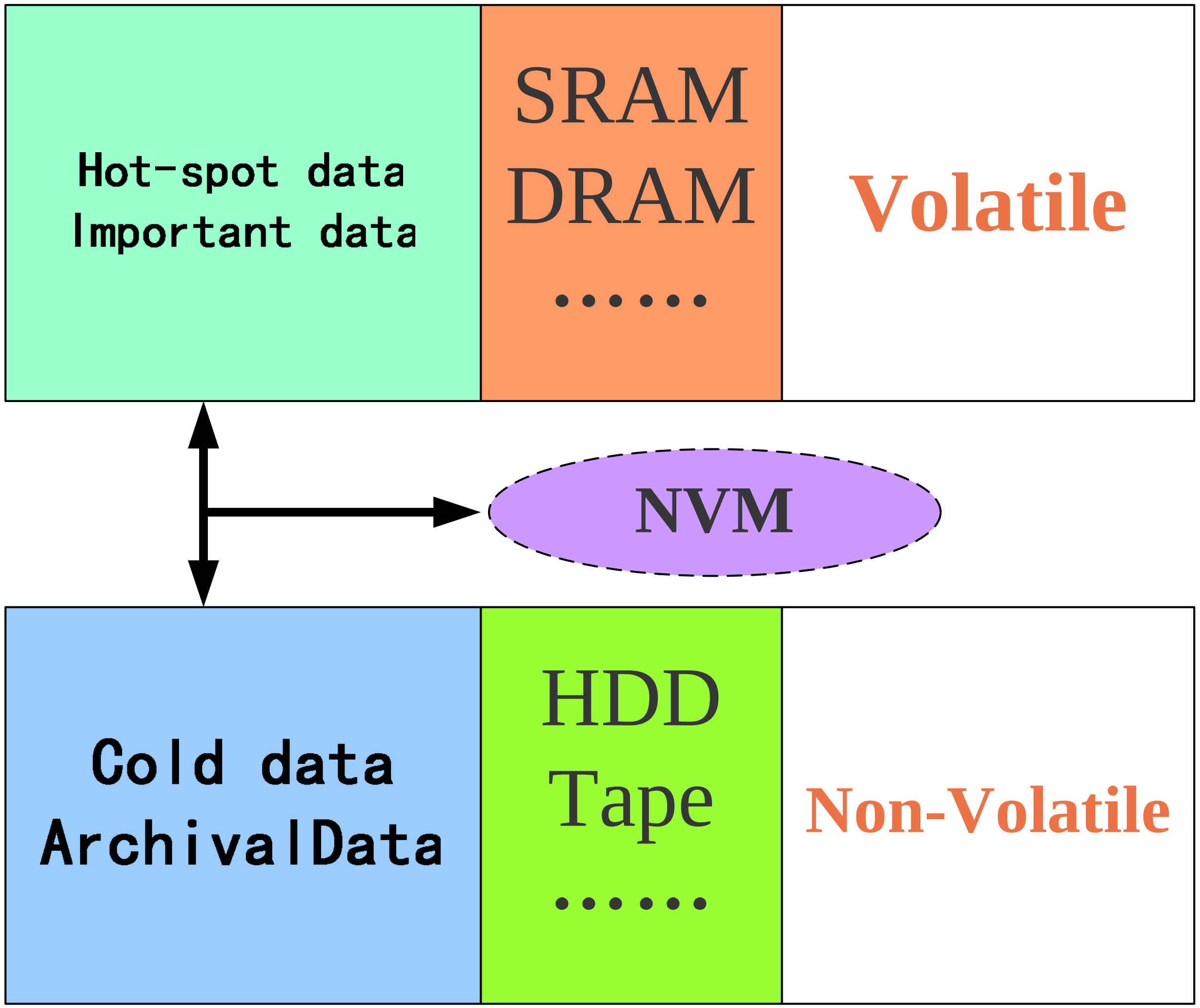
Current and Future Storage 5

- SMILE
- Edge：
- Edge computing, fog computing, proximity computing, ……

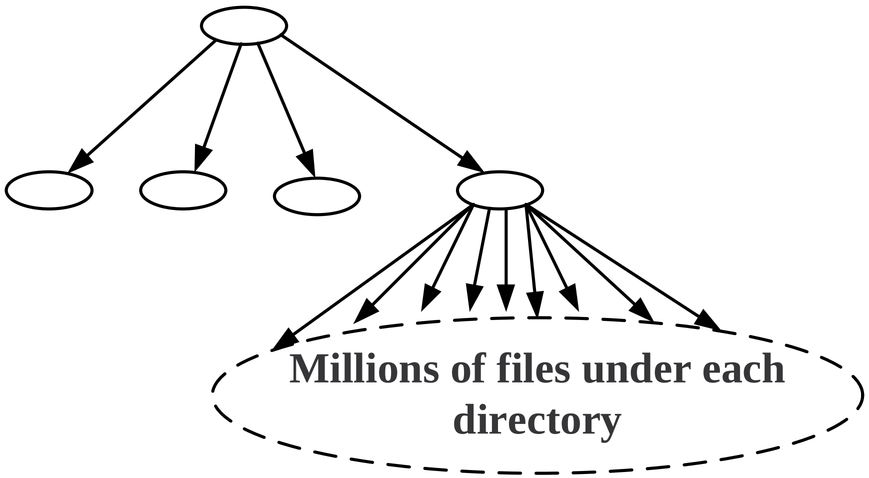
Challenge:
Hierarchical Architecture

- ➢ Heterogeneous Principle
- ➢ Differentiated Performance
- ➢ Management Complexity
- One-storey house->Skyscraper
- More and more levels

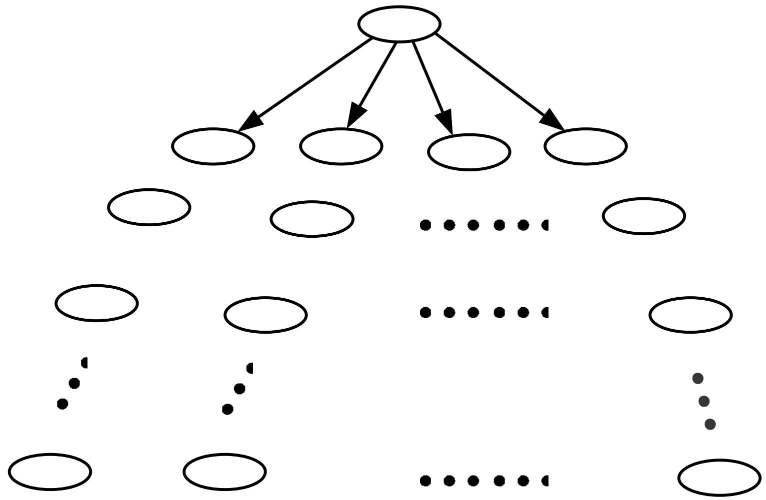
Challenge: Storage Reliability



Hierarchical Data Structure



This tree is too FAT !



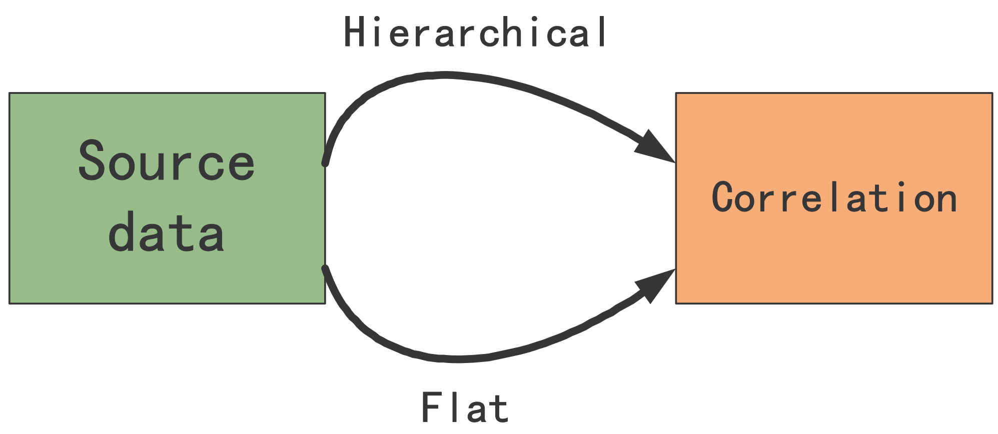
This tree is too HIGH !

Hierarchical and Vertical Architecture

- **Idea: based on locality principle, some key data consume many system resources.**
- However, in the era of big data, the efficiency of locality becomes weak, thus being difficult to improve hit ratio.

The essence behind Hierarchy

- Goal : identify the correlation
- In essence, the hierarchy is an approach to dynamic filter data to obtain correlated aggregation and on-demand allocation.
- If the flat or semi-hierarchical schemes are able to achieve the same goal, it would be much better with significant performance improvements.



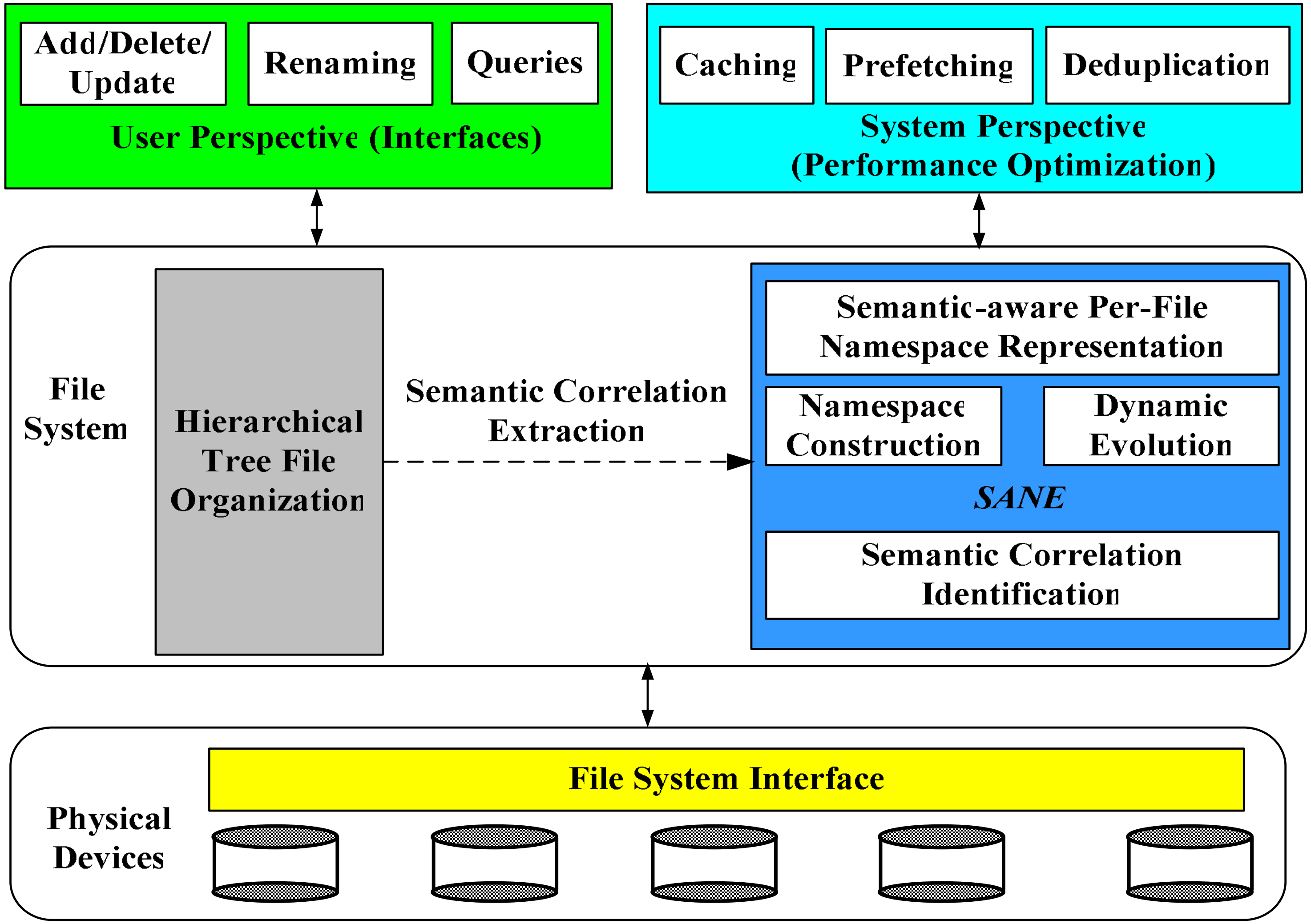
Semi-hierarchical Architecture

- Problem to be addressed:
  - ➢ How to storage data in large-scale storage systems
- The idea:
  - ➢ Semantic storage is the new form of implementing storage systems.

Our related work

- **Semantic Namespace**：SANE(TPDS14)
- **Semantic Aggregation**：FAST(SC14), HAR(ATC14), SiLo(ATC11),
- **Semantic Hash Computation**：SmartCuckoo(ATC17), DLSH(SoCC17), SmartEye(INFOCOM15), NEST(INFOCOM13)
- **Semantic On-line Service**：ANTELOPE(TC14)

SANE: The namespace

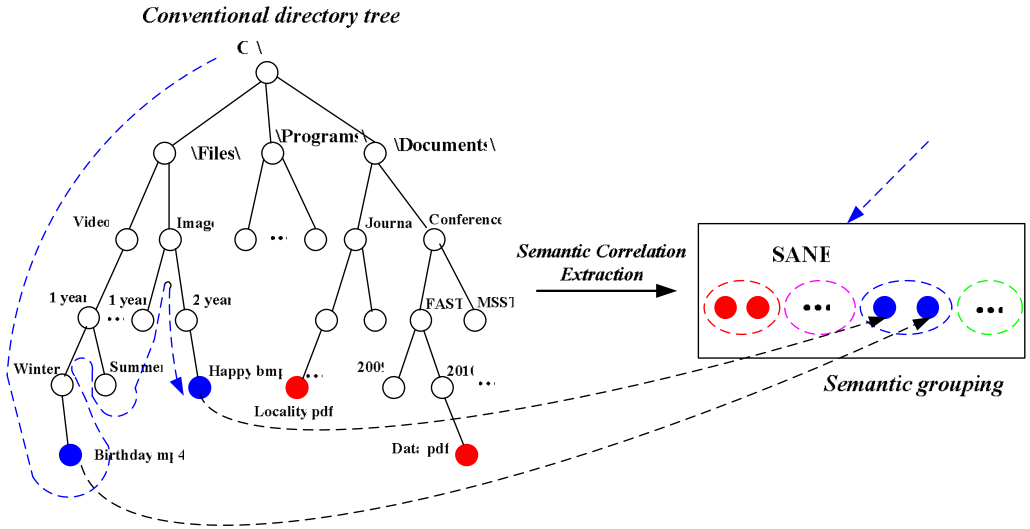


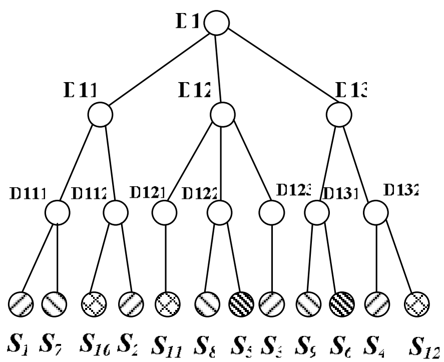
"SANE: Semantic-Aware Namespace in Ultra-large-scale File Systems", IEEE Transactions on Parallel and Distributed Systems (TPDS), Vol.25, No.5, May 2014, pages:1328-1338.

SANE: The Semantic Namespace

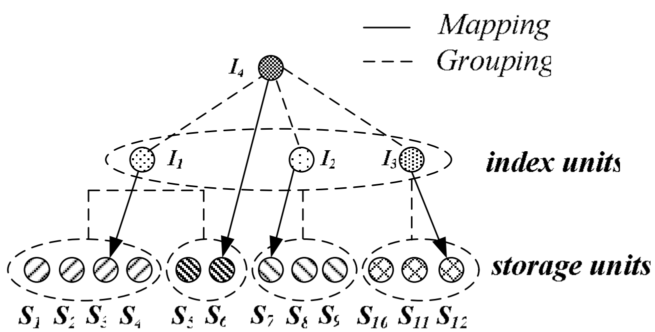
Flat Addressing

- Hierarchy becomes the performance bottleneck
- **Design goals :**
  - Searchable
  - Unique





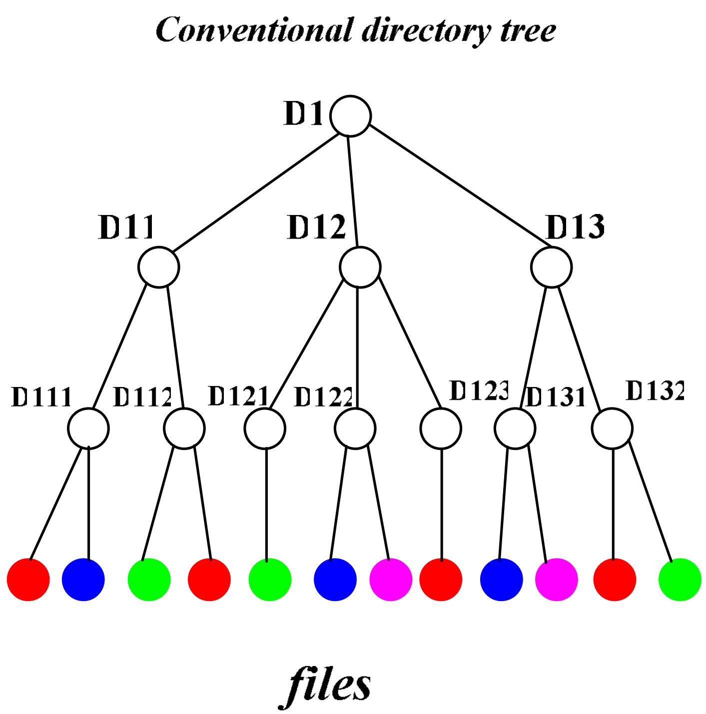
Conventional directory tree

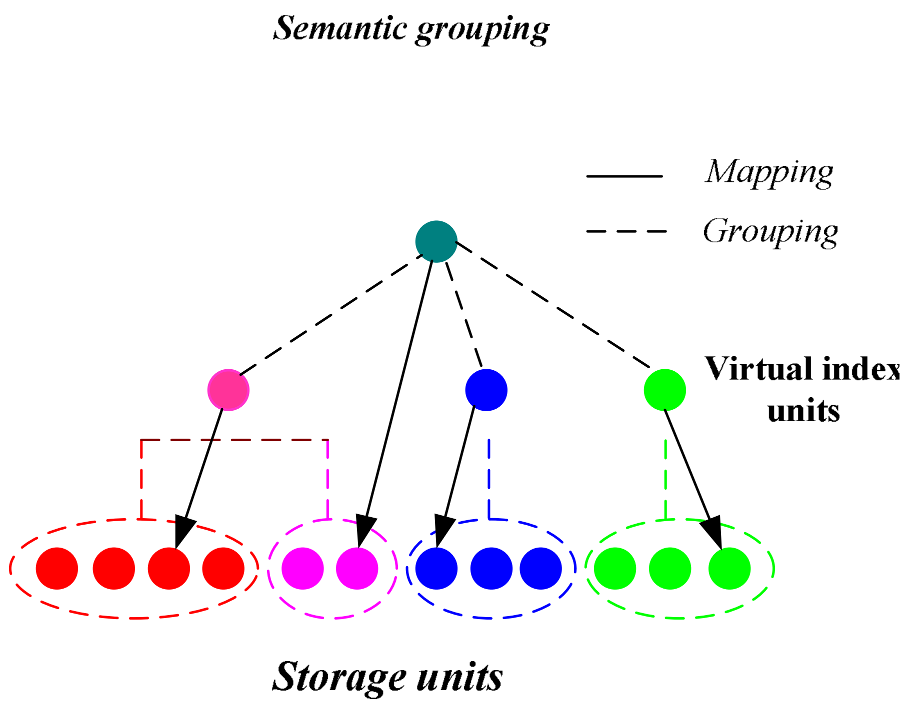


Semantic R-tree

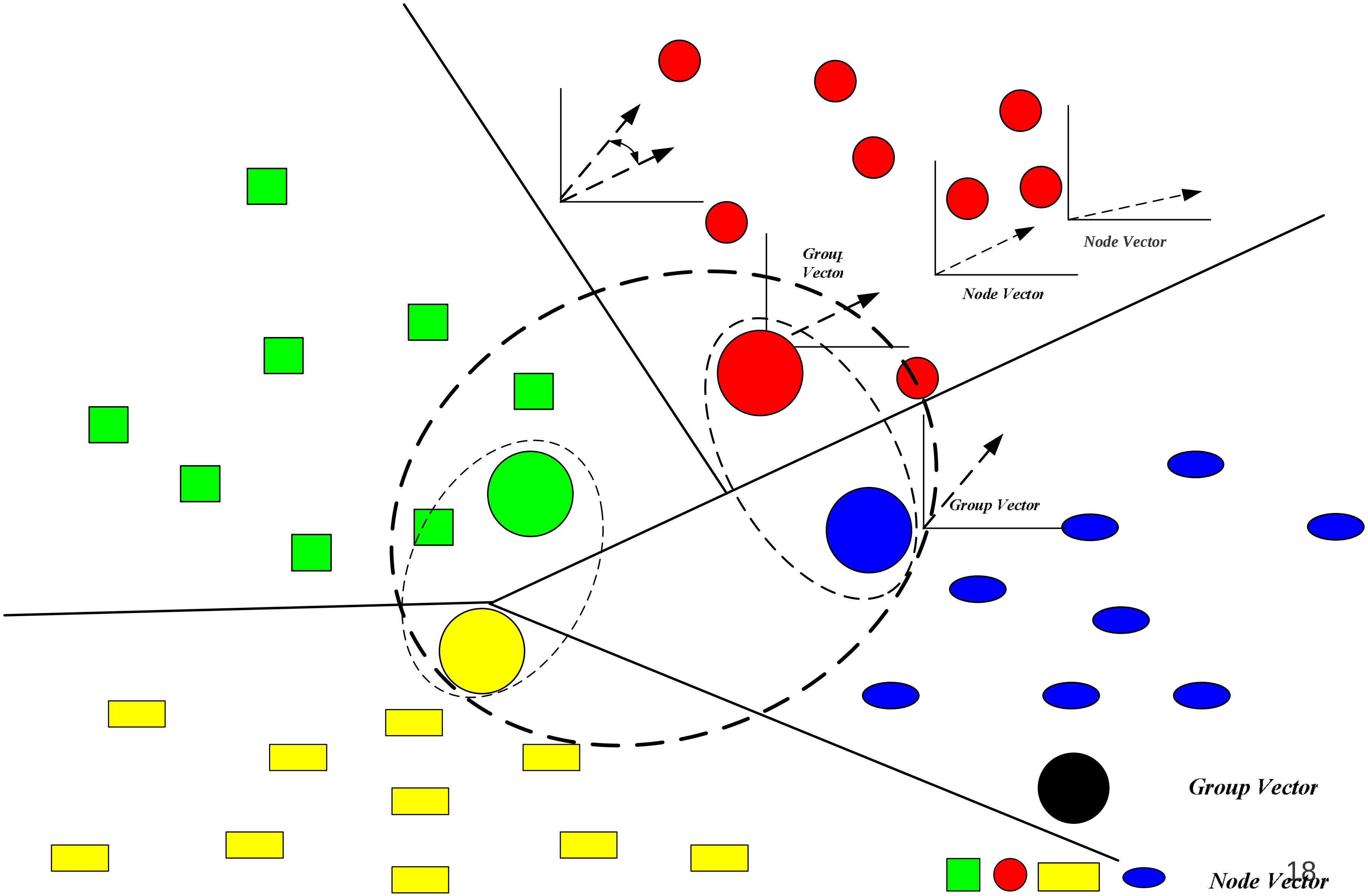
- Construct the semantic-aware namespace

Comparisons with Conventional File Systems



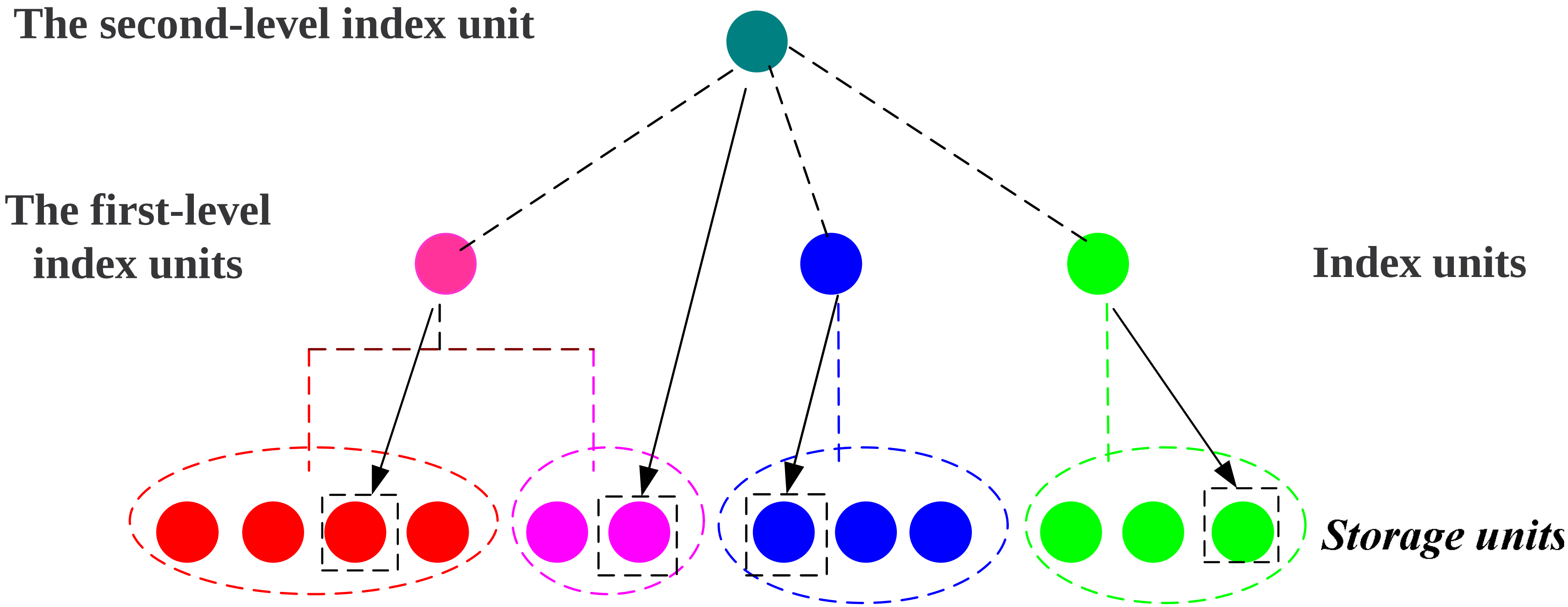


Grouping Procedures

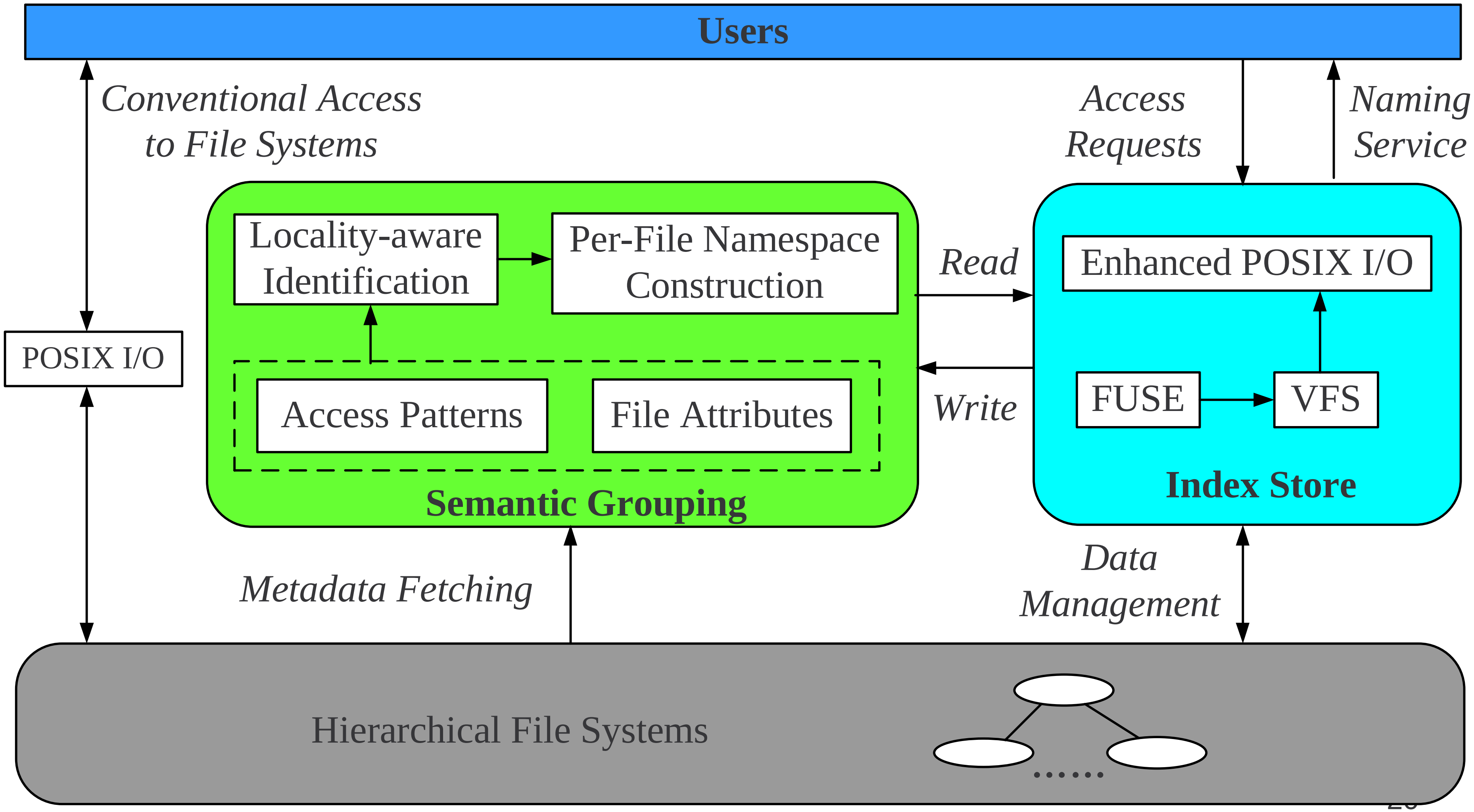


Mapping of Index Units

- **Our mapping is based on a simple bottom-up approach that iteratively applies random selection and labeling operations.**



Components

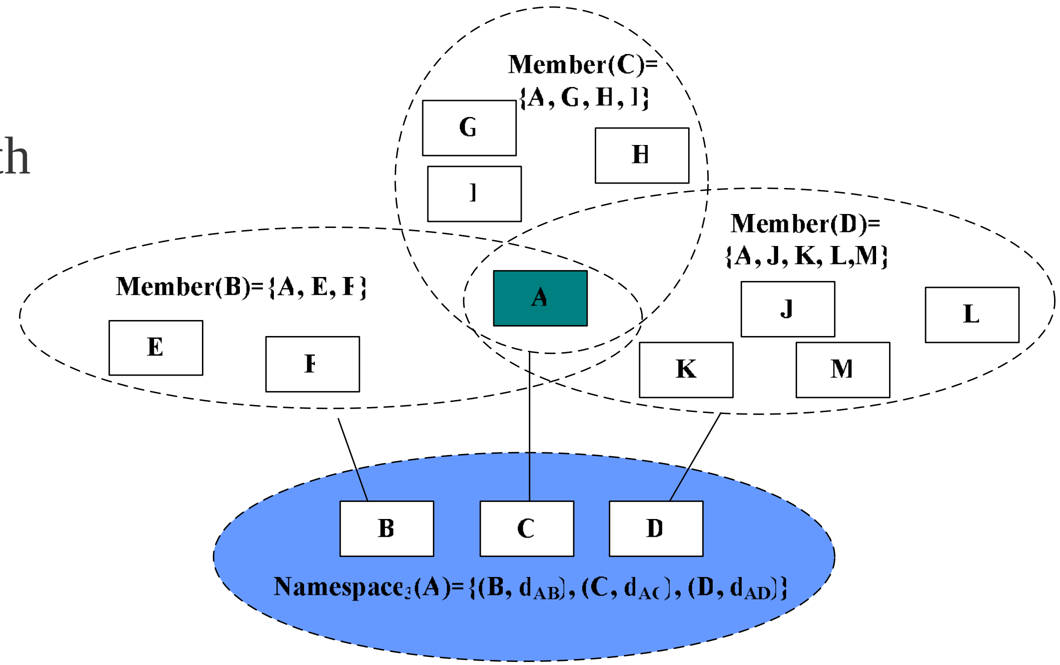


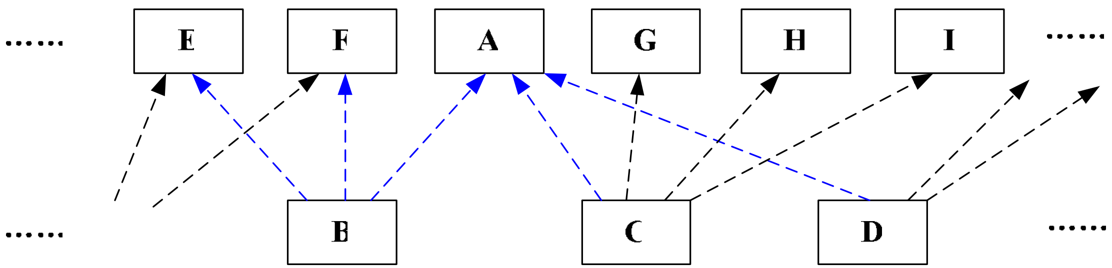
Users
Conventional Access
to File Systems
Access
Requests
Naming
Service
POSIX I/O
Locality-aware
Identification
Per-File Namespace
Construction
Access Patterns
File Attributes
Semantic Grouping
Read
Write
Enhanced POSIX I/O
FUSE
VFS
Index Store
Metadata Fetching
Data
Management
Hierarchical File Systems
……

Naming and Rename

- **Submodular Maximization**
- Select a subset of namespaces with distinct names

$$S^* \in \underset{S \subseteq V}{\operatorname{argmax}}\, F(S) \quad s.t.\ |S| \leq T.$$





Namespace$_3$(A)={(B, d_{AB}), (C, d_{AC}), (D, d_{AD})}

Member$_3$(B)={(A, d_{AB}), (E, d_{BE}), (F, d_{BF})}

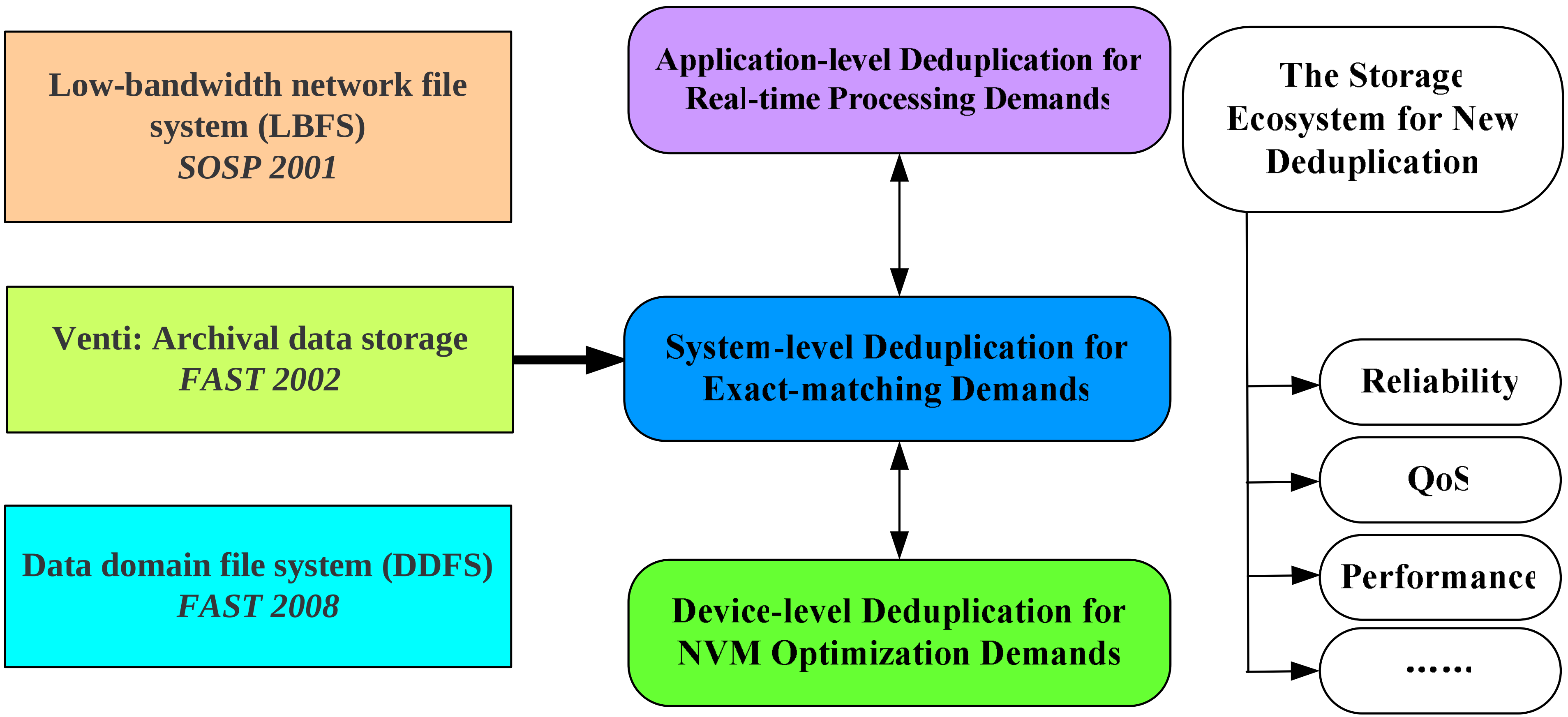
Member$_4$(C)={(A, d_{AC}), (G, d_{CG}), (H, d_{CH}), (I, d_{CI})}

Member$_5$(D)={(A, d_{AD}), …… }

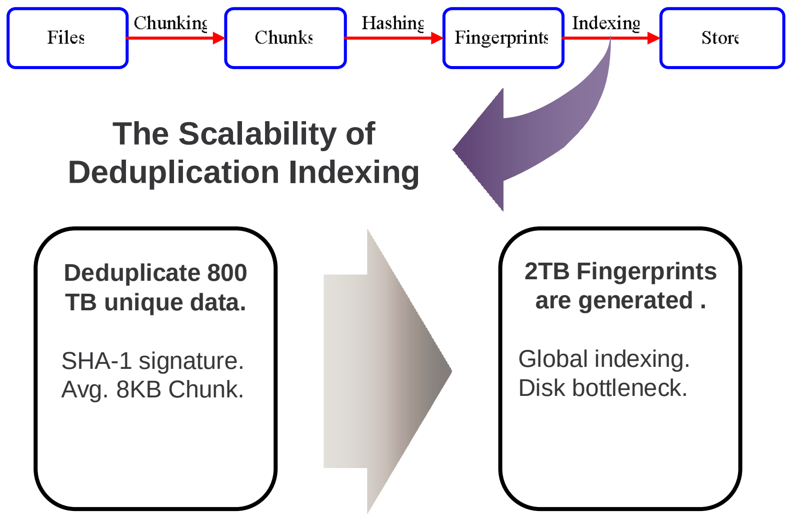
Maximization for Monotone Submodular functions

- Scoring Function is a monotone submodular function
  - Greedy algorithm
  - Constant-scale mathematical quality guarantee

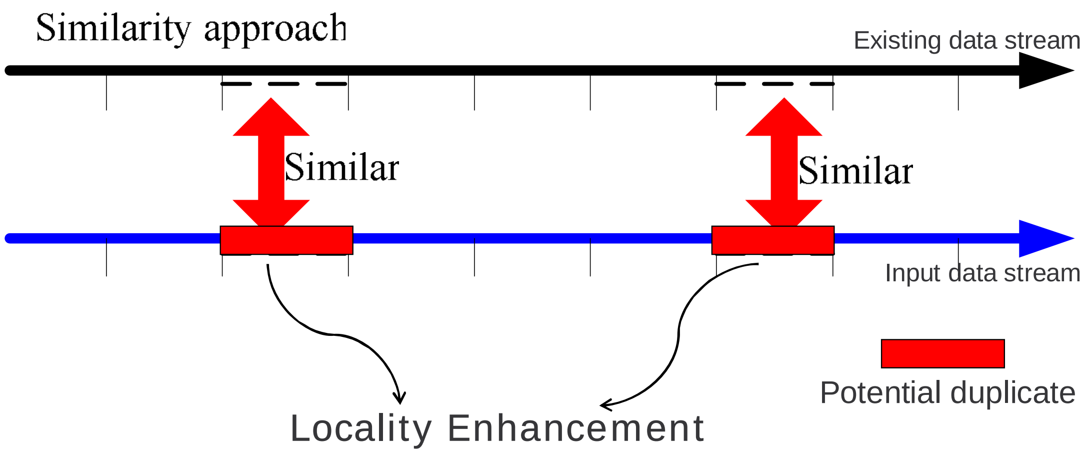
Example 1:
New Deduplication Ecosystem



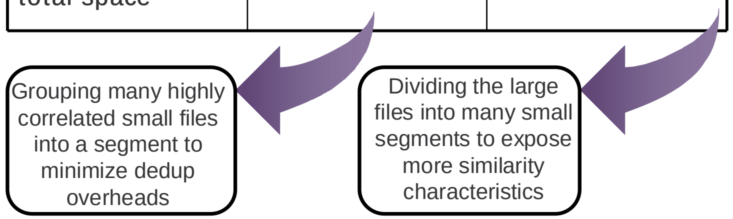
The Synergization of Similarity and Locality——SiLo



- Expose and exploit more similarity by grouping strongly correlated small files into a segment and segmenting large files
- Leverage locality in the backup stream by grouping contiguous segments into blocks to capture similar and duplicate data missed by the probabilistic similarity detection.



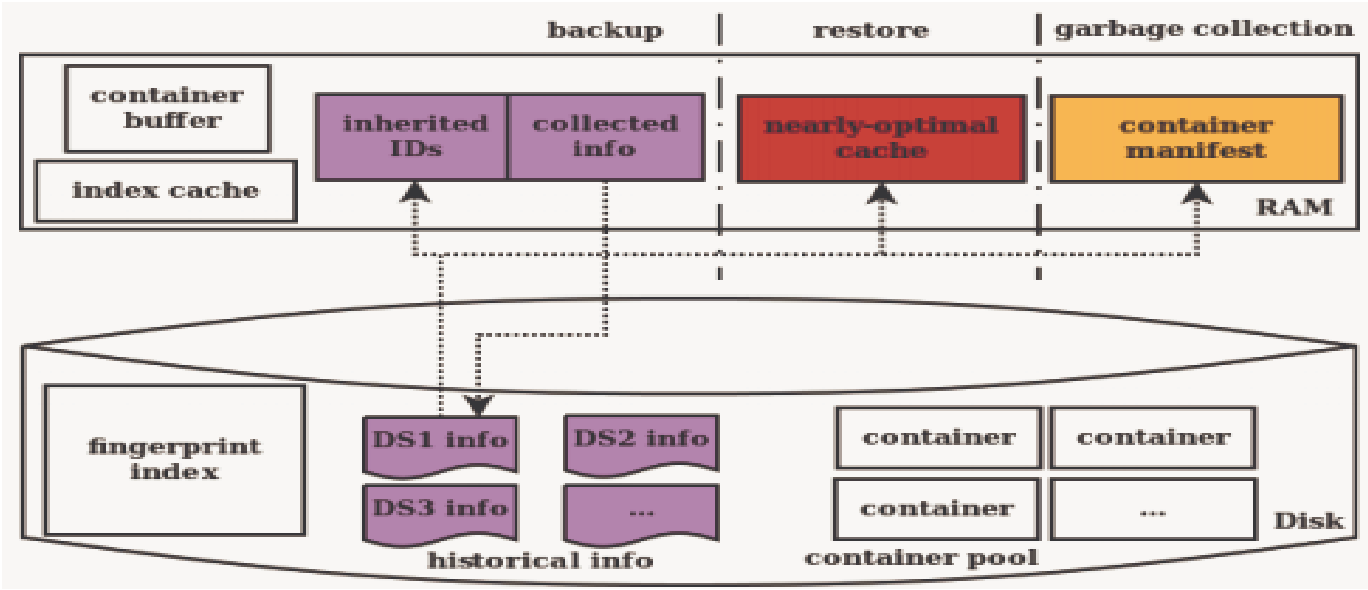
	Small files (⩽ 64KB)	Large files (⩾ 2 MB)
Percentage of total file number	⩾ 80%	⩽ 20%
Percentage of total space	⩽ 20%	⩾ 80%



"SiLo: A Similarity-Locality based Near-Exact Deduplication Scheme with Low RAM Overhead and High Throughput," Proceedings of ***USENIX ATC****, June 2011.*

Fragmentation in Deduplication

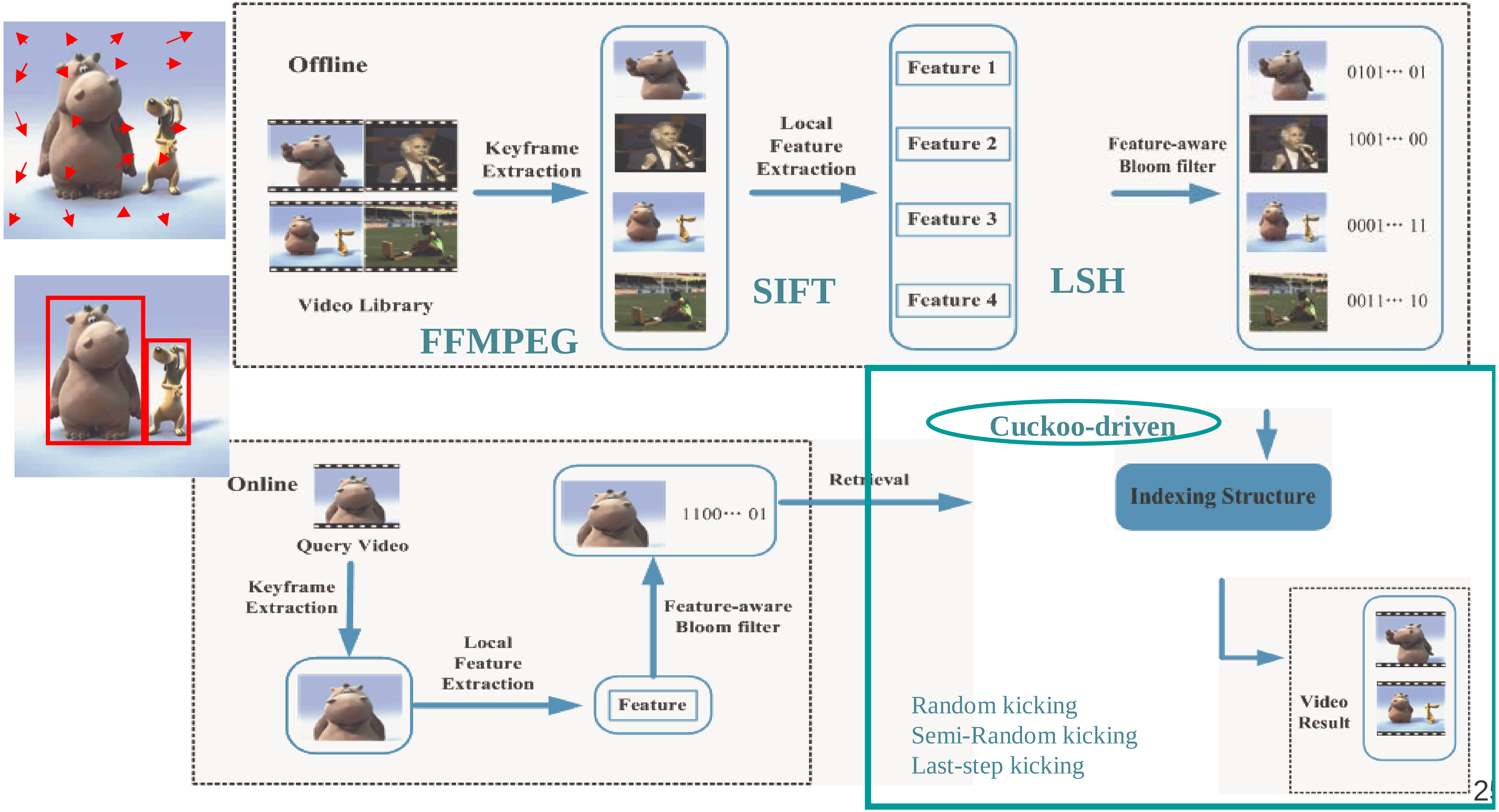
- History-Aware Rewriting algorithm (HAR)



- The fragmentation decreases restore performance and results in invalid chunks becoming physically scattered in different containers after users delete backups.
- HAR exploits historical information of backup systems to more accurately identify and rewrite fragmented chunks.

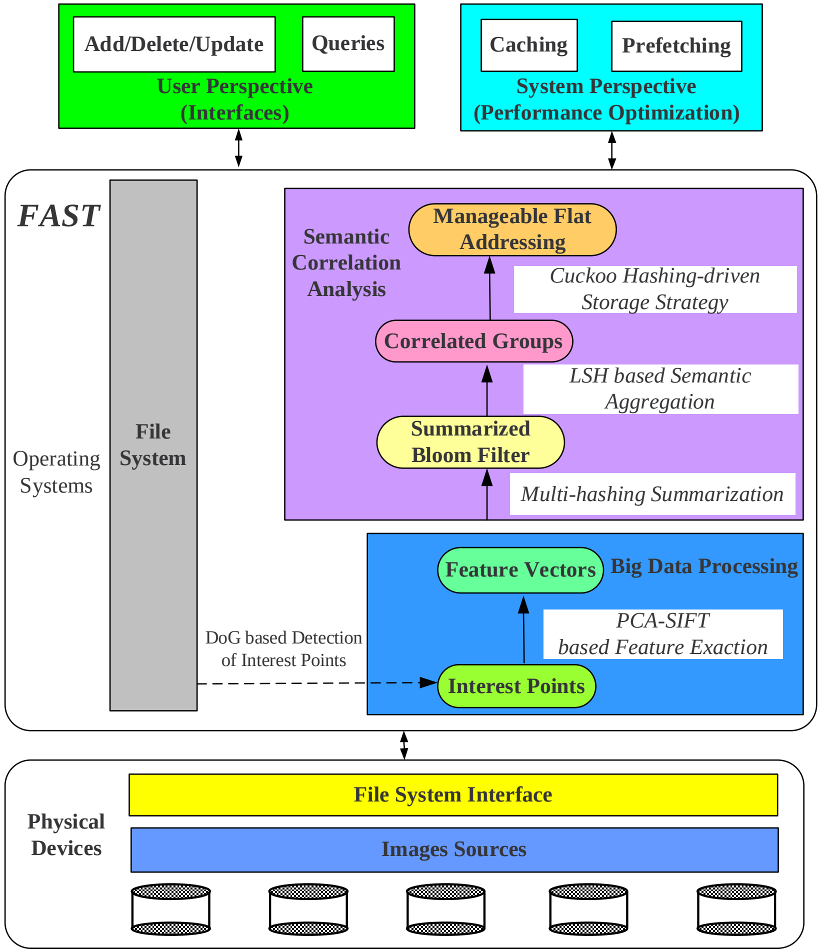
"Accelerating Restore and Garbage Collection in Deduplication-based Backup Systems via Exploiting Historical Information", Proc. ***USENIX ATC****, 2014,*

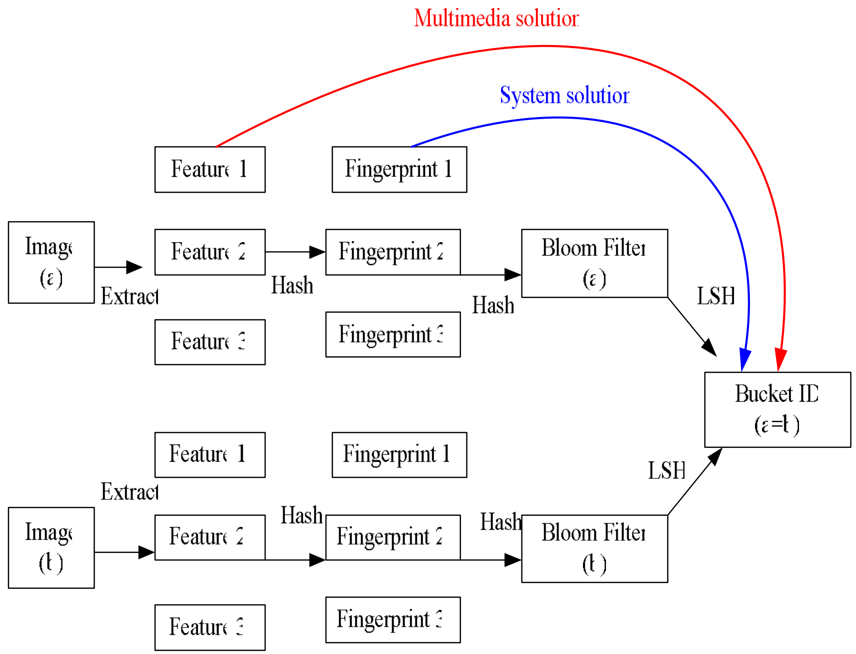
Example 2: Application-level Approximate Methodology--FAST



*"FAST: Near Real-time Searchable Data Analytics for the Cloud", Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (**SC**), November 2014*

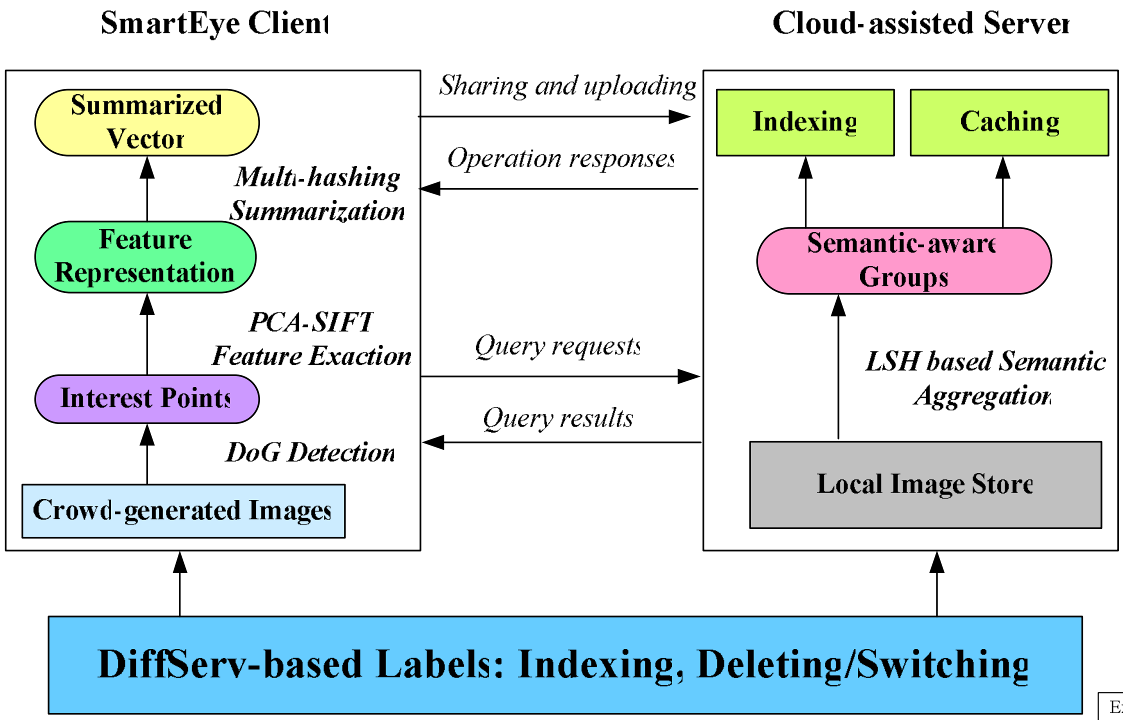
Application-level Approximate Methodology: FAST

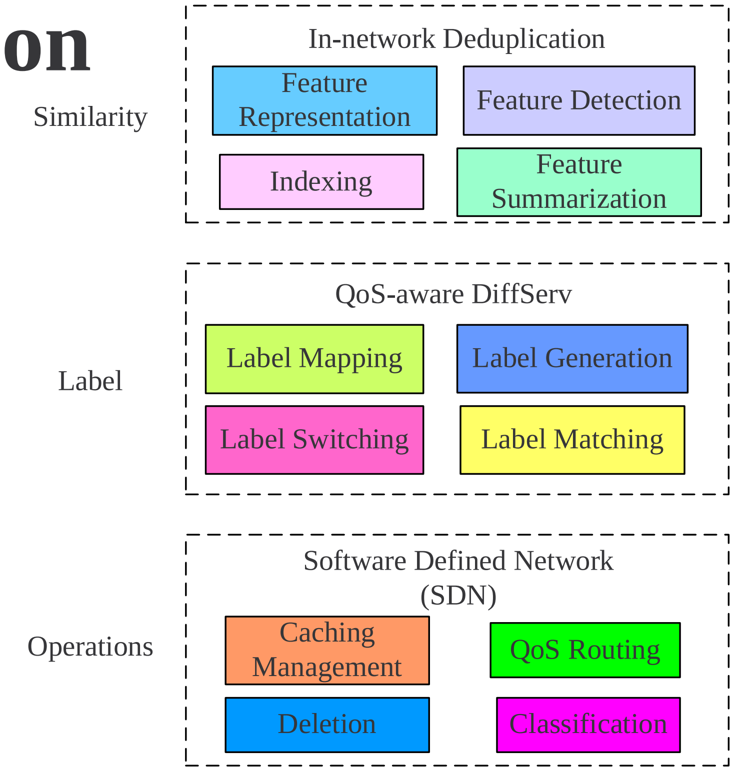


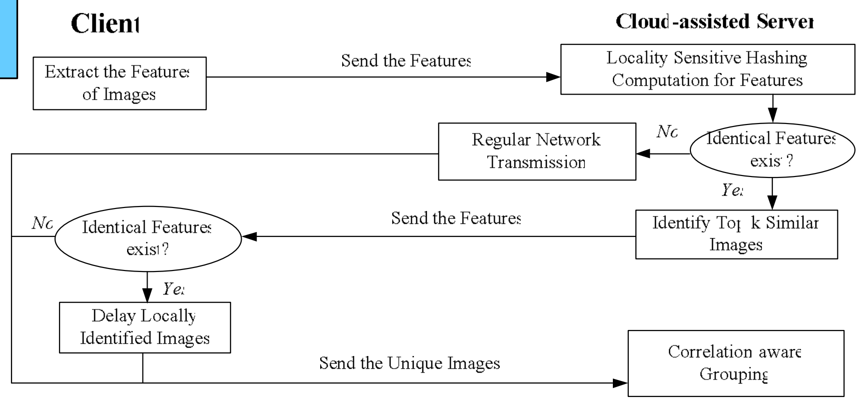


*"FAST: Near Real-time Searchable Data Analytics for the Cloud", Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (**SC**), November 2014*

Approximate Image Transmission in Networking: SmartEye



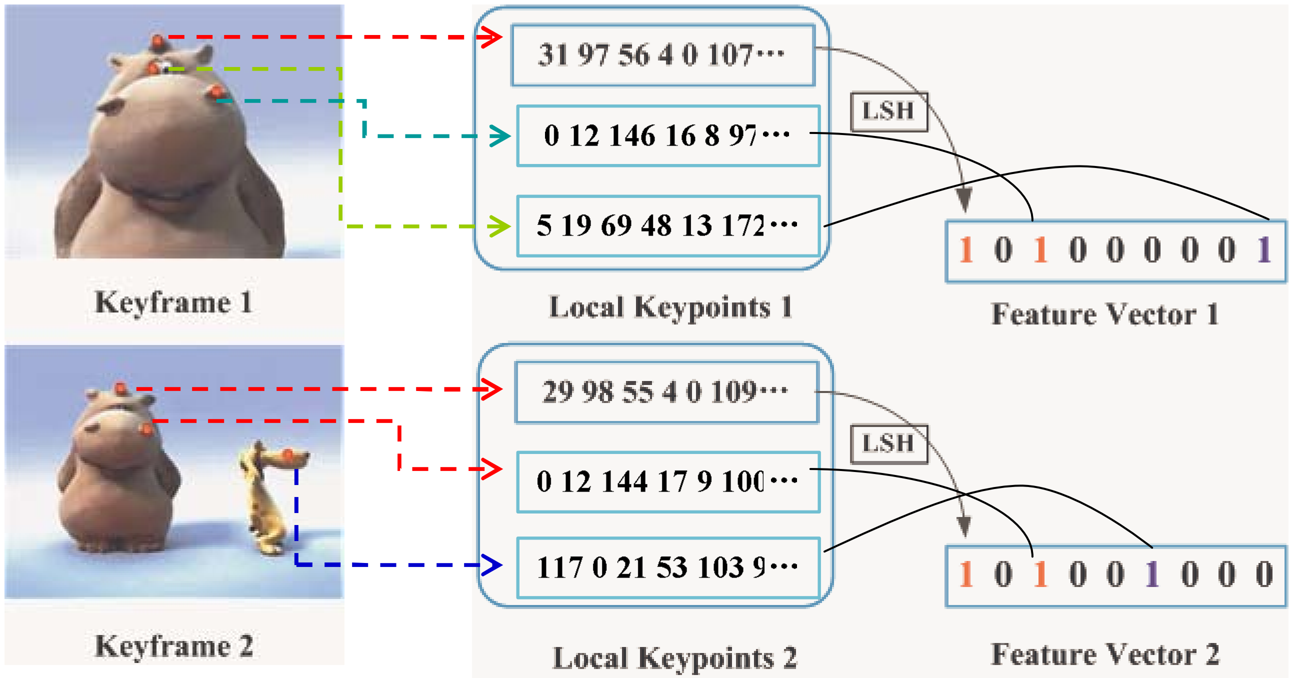




"SmartEye: Real-time and Efficient Cloud Image Sharing for Disaster Environments"
Proceedings of INFOCOM, 2015, pages: 1616-1624

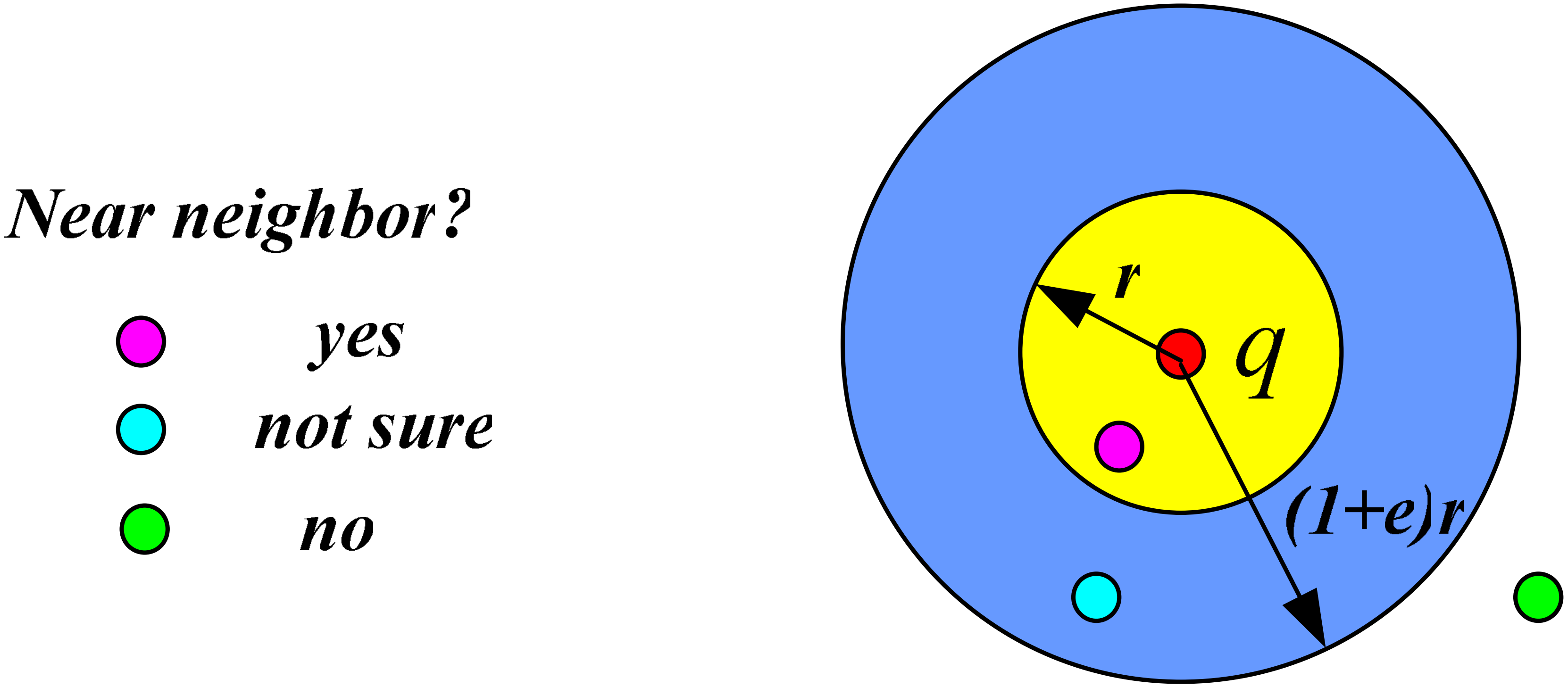
The design of SmartEye

- Compact Feature Representation

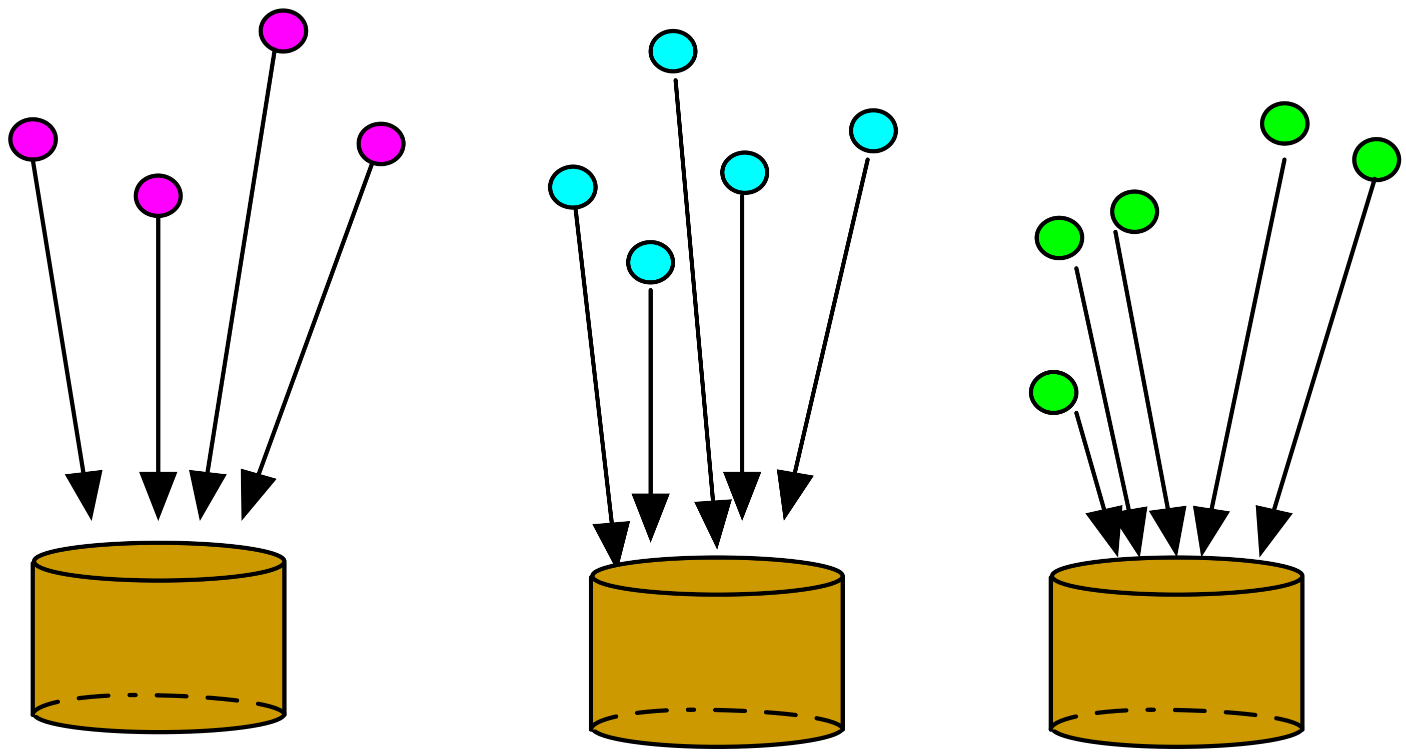


Locality Sensitive Hashing (LSH)

- If $||p,q||_s \leq R$ then $Pr_{\mathbb{H}}[h(p) = h(q)] \geq P_1$,
- If $||p,q||_s > cR$ then $Pr_{\mathbb{H}}[h(p) = h(q)] \leq P_2$.



Locality-Sensitive Hashing (LSH)

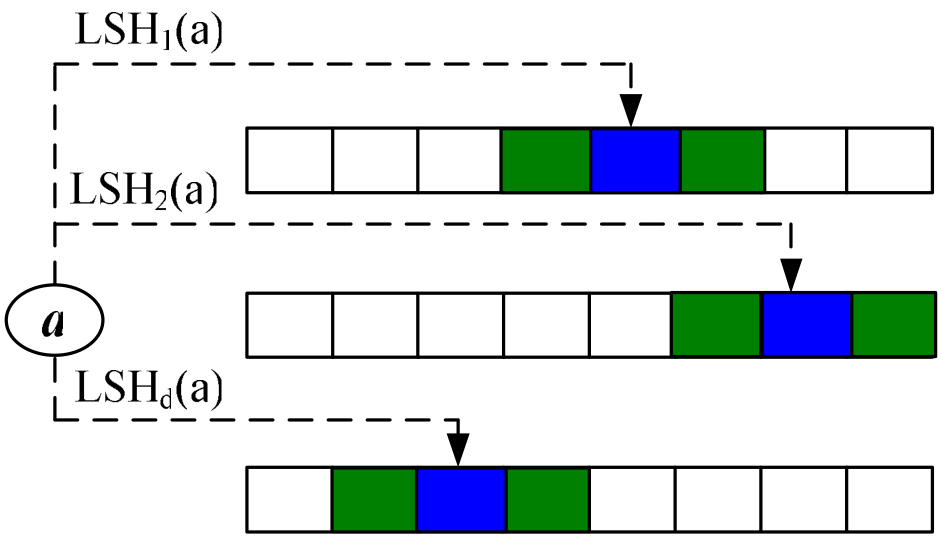


- **Close items will collide with high probability**
- **Distant items will have very little chance to collide**

Efficient Cuckoo-driven LSH

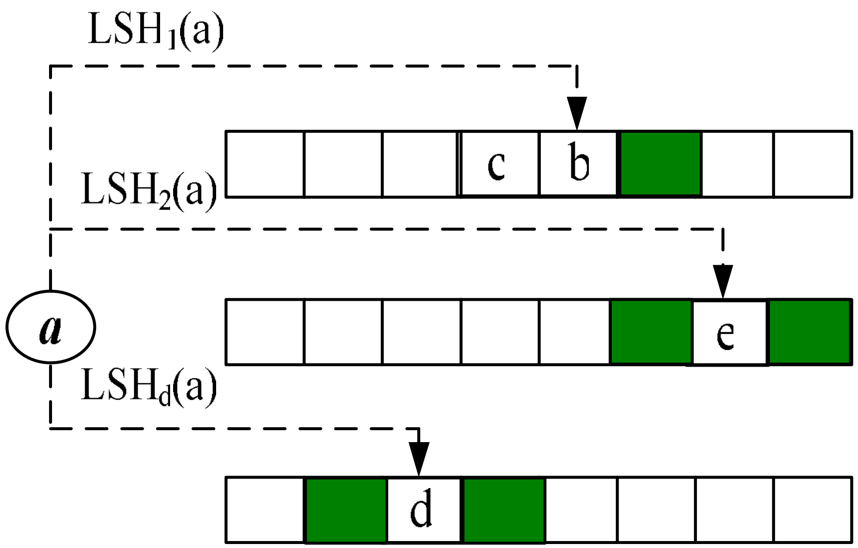
(1) Use Cuckoo Driven LSH to reduce search time when collision occurs
(2) Use neighbor buckets to further reduce the possibility of kickout
(3) Space efficiency due to neighboring probe and data locality

Blue: hit position by LSH computation
Green: Neighbor bucket has data correlation



A multi-choice LSH

If all $LSH_i(a)$ are full, can choose adjacent empty bucket



Available locations for item *a*

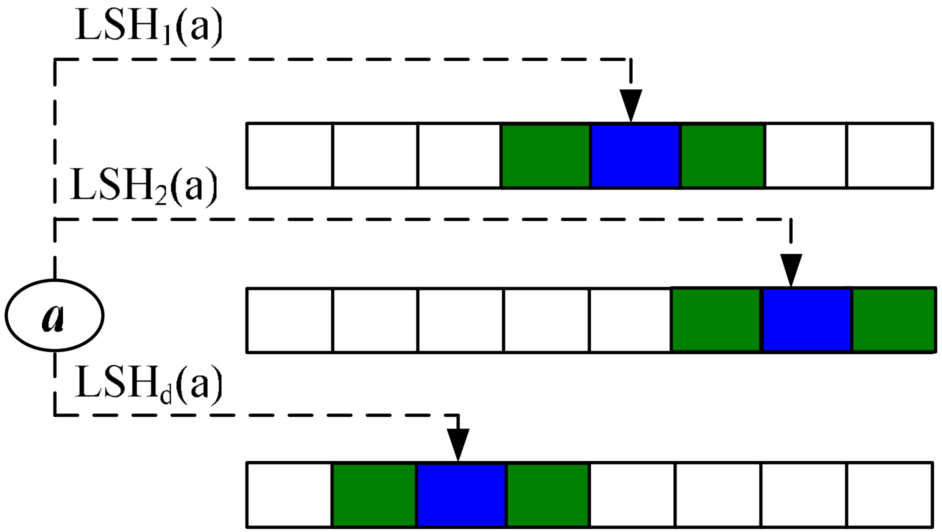
Probing adjacent neighbors: the probability of endless "kicking out" is much more smaller than ordinary cuckoo hashing

NEST: Efficient Cuckoo-driven LSH

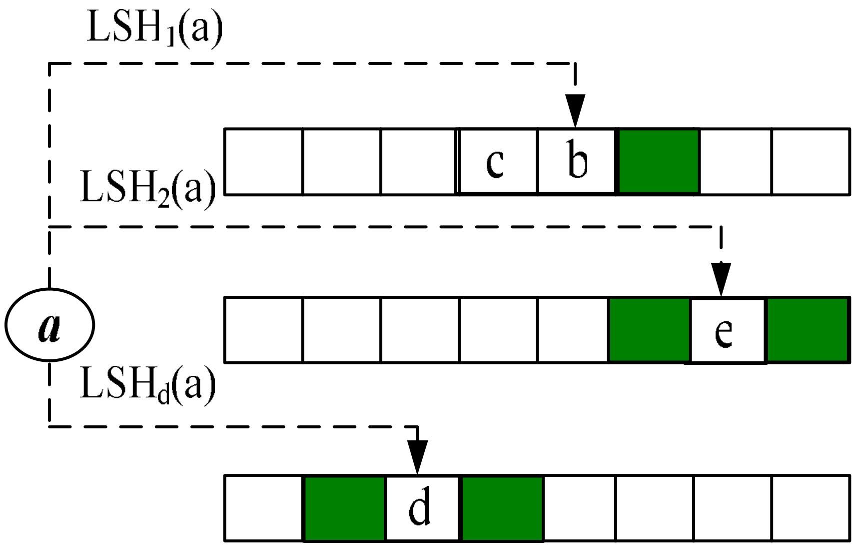
(1) Use Cuckoo Driven LSH to reduce search time when collision occurs
(2) Use neighbor buckets to further reduce the possibility of kickout
(3) Space efficiency due to neighboring probe and data locality

Blue: hit position by LSH computation
Green: Neighbor bucket has data correlation

If all $LSH_i(a)$ are full, can choose adjacent empty bucket



A multi-choice LSH

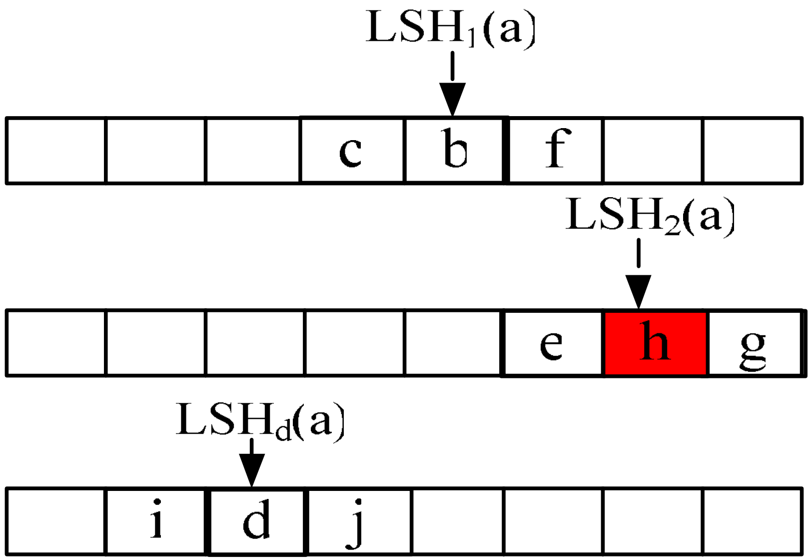


Available locations for item *a*

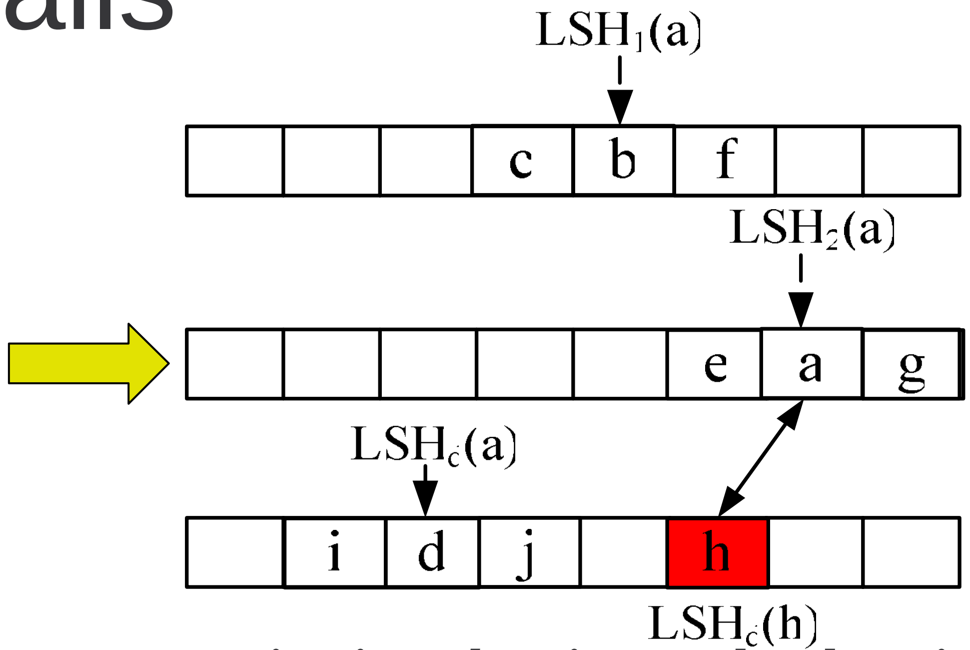
Probing adjacent neighbors: the probability of endless "kicking out" in NEST is much more smaller than ordinary cuckoo hashing

"NEST: Locality-aware Approximate Query Service for Cloud Computing", Proceedings of INFOCOM, April 2013, pages: 1327-1335

NEST: Resolve Collision: if still fails

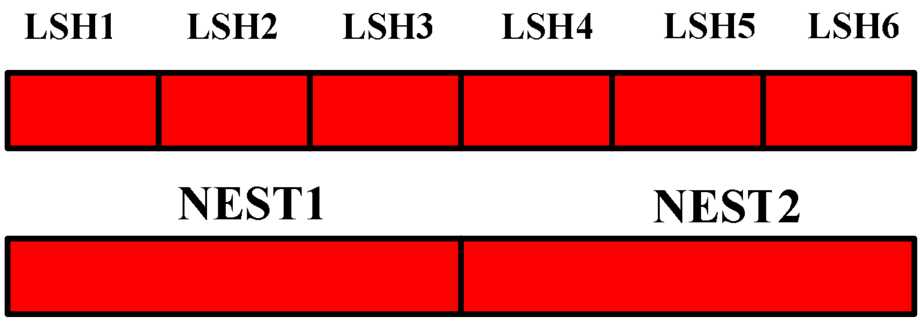


Hashing collisions for inserting item *a*

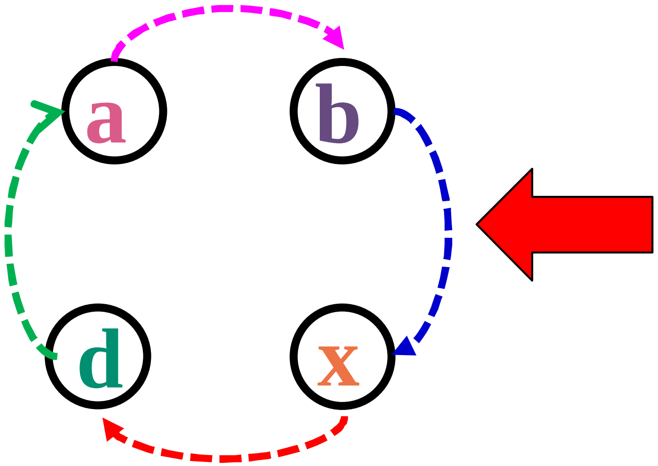


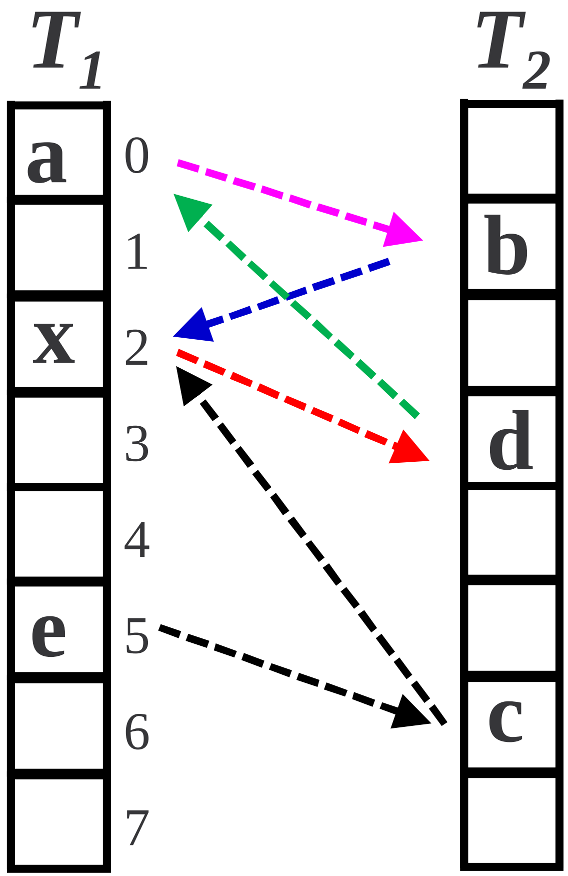
Moving item *h* to its another location

Note: Adjacent probing significantly reduce or even avoid hash failing (FAST INDEX)



Example 3: Pseudoforest



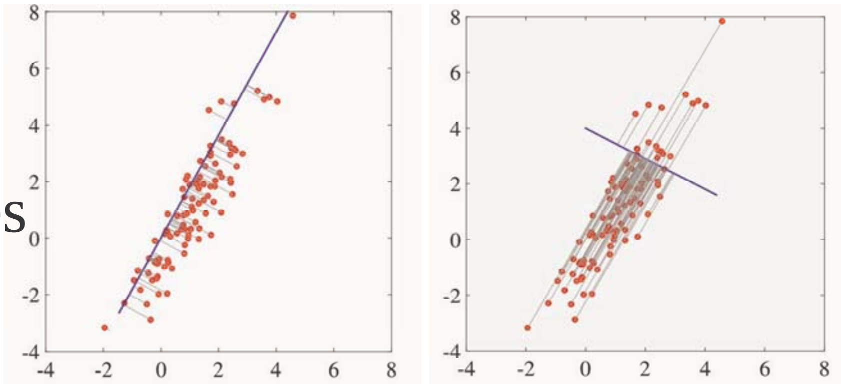


- An **endless loop** is formed.
- **Endless kickouts** for any insertion within the loop.

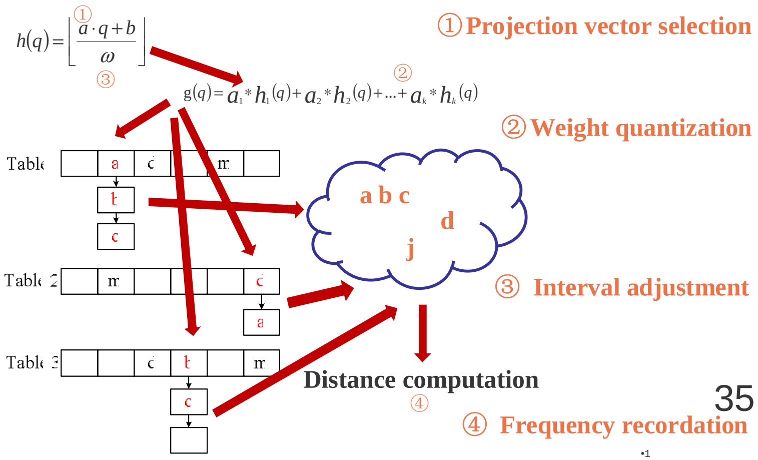
*"SmartCuckoo: A Fast and Cost-Efficient Hashing Index Scheme for Cloud Storage Systems", Proceedings of USENIX Annual Technical Conference (**USENIX ATC**), July 2017, pages: 553-566*

DLSH: A Distribution-aware LSH

- Due to distribution-unaware projection vectors:
  - Multiple hash tables to maintain data locality and guarantee the query accuracy.

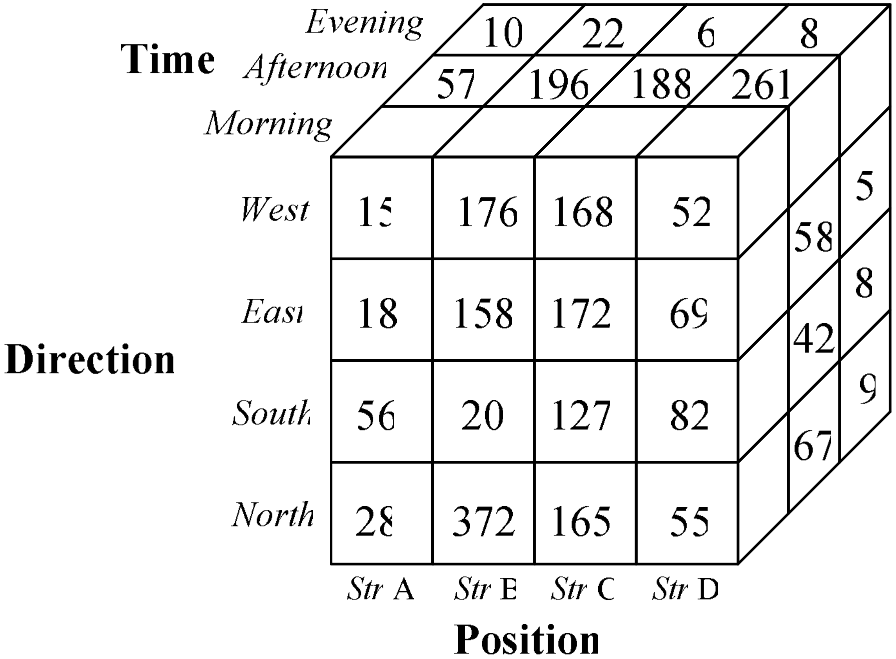


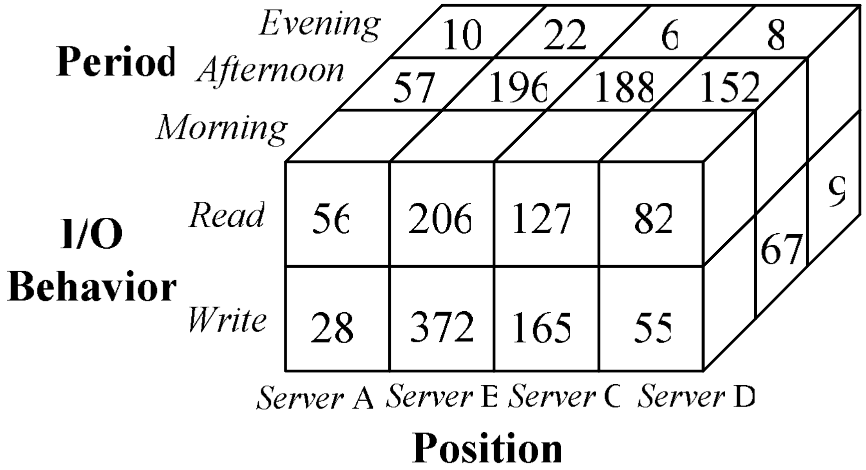
- Design goal:
  - Decrease the number of hash tables
  - Mitigate in-memory consumption
- Approach:
  - Differentiating the aggregated data in a suitable direction;
  - Exhibiting the data locality as well as decreasing the hash collisions.



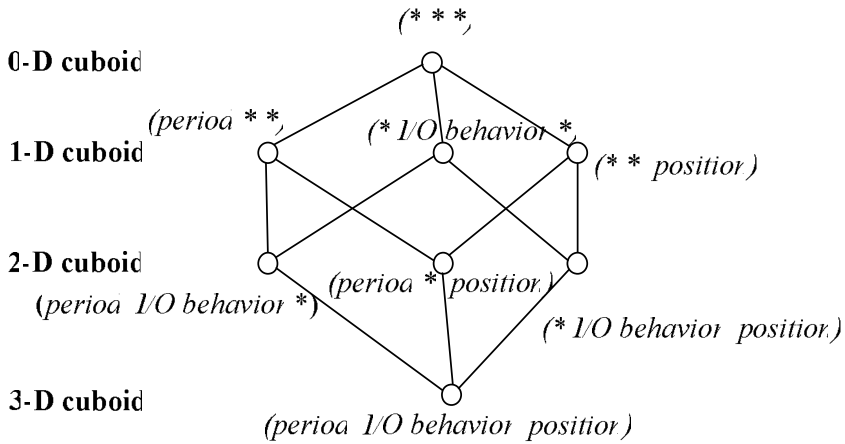
"DLSH: A Distribution-aware LSH Scheme for Approximate Nearest Neighbor Query in Cloud Computing", Proceedings of ACM Symposium on Cloud Computing (SoCC), 2017

Example 4: On-line Precomputation--Data Cube





- Leverage precomputation based data cube to support online cloud services
- Use semantic-aware partial materialization to reduce the operation and space overheads



"ANTELOPE: A Semantic-aware Data Cube Scheme for Cloud Data Center Networks", IEEE Transactions on Computers (TC), Vol.63, No.9, September 2014, pages: 2146-2159.

Open Source Codes (in GitHub)

- ***SmartCuckoo***: *in GitHub. SmartCuckoo is a new cuckoo hashing scheme to support metadata query service.*
- ➢ **https://github.com/syy804123097/SmartCuckoo**
- ***SmartSA*** *(E-STORE)*: *in GitHub to support near-deduplication for image sharing based on the energy availability in Smartphone.*
- ➢ **https://github.com/Pfzuo/SmartSA**
- ***Real-time-Share***: *in GitHub, to support real-time image sharing in the cloud, which is an important component of SmartEye (INFOCOM 2015).*
- ➢ **https://github.com/syy804123097/Real-time-Share**
- ***MinCounter***: *in GitHub. MinCounter is the proposed data structure in the MSST 2015 Paper.*
- ➢ **https://github.com/syy804123097/MinCounter**
- ***NEST***: *in GitHub (Download INFOCOM 2013 Paper, Source Codes, Manual and TraceData).*
- ➢ **https://github.com/syy804123097/NEST**
- ***LSBF*** *(Locality-Sensitive Bloom Filter): in GitHub (Download TC 2012 Paper, Source Codes and Manual).*
- ➢ **https://github.com/syy804123097/LSBF**

Thanks and Questions