

HYDRAsstor: a Scalable Secondary Storage

7th USENIX Conference on File and Storage Technologies
(FAST '09)

February 26th 2009



**NEC Laboratories
America**
Relentless passion for innovation

C. Dubnicki, L. Gryz, L. Heldt,
M. Kaczmarczyk, W. Kilian,
P. Strzelczak, J. Szczepkowski,
M. Welnicki

C. Ungureanu

Scalable secondary storage

Characteristics	Requirements
Huge amount of data	<ul style="list-style-type: none">- Scalability (dynamic)- Low cost per TB
Small backup windows	<ul style="list-style-type: none">- Very high write performance
Duplication between backup streams	<ul style="list-style-type: none">- Global deduplication
Reliable, on-line retrieval	<ul style="list-style-type: none">- Failure tolerance- High restore performance
Varying value of data	<ul style="list-style-type: none">- Adjust resilience overhead- Data deletion

Scalable secondary storage

Characteristics	Requirements
Huge amount of data	<ul style="list-style-type: none">- Scalability (dynamic)- Low cost per TB
Small backup windows	<ul style="list-style-type: none">- Very high write performance
Duplication between backup streams	<ul style="list-style-type: none">- Global deduplication
Reliable, on-line retrieval	<ul style="list-style-type: none">- Failure tolerance- High restore performance
Varying value of data	<ul style="list-style-type: none">- Adjust resilience overhead- Data deletion

Scalable secondary storage

Characteristics	Requirements
Huge amount of data	<ul style="list-style-type: none">- Scalability (dynamic)- Low cost per TB
Small backup windows	<ul style="list-style-type: none">- Very high write performance
Duplication between backup streams	<ul style="list-style-type: none">- Global deduplication
Reliable, on-line retrieval	<ul style="list-style-type: none">- Failure tolerance- High restore performance
Varying value of data	<ul style="list-style-type: none">- Adjust resilience overhead- Data deletion

Scalable secondary storage

Characteristics	Requirements
Huge amount of data	<ul style="list-style-type: none">- Scalability (dynamic)- Low cost per TB
Small backup windows	<ul style="list-style-type: none">- Very high write performance
Duplication between backup streams	<ul style="list-style-type: none">- Global deduplication
Reliable, on-line retrieval	<ul style="list-style-type: none">- Failure tolerance- High restore performance
Varying value of data	<ul style="list-style-type: none">- Adjust resilience overhead- Data deletion

Scalable secondary storage

Characteristics	Requirements
Huge amount of data	<ul style="list-style-type: none">- Scalability (dynamic)- Low cost per TB
Small backup windows	<ul style="list-style-type: none">- Very high write performance
Duplication between backup streams	<ul style="list-style-type: none">- Global deduplication
Reliable, on-line retrieval	<ul style="list-style-type: none">- Failure tolerance- High restore performance
Varying value of data	<ul style="list-style-type: none">- Adjust resilience overhead- Data deletion

Challenges

- High-performance, decentralized
 global deduplication
 ... in a **dynamic**, distributed system
 ... with deletion and **failures**
- Combination introduces complexity
- **Tension** between:
 - Deduplication and dynamic scalability
 - Deduplication and on-demand deletion
 - Failure tolerance and deletion



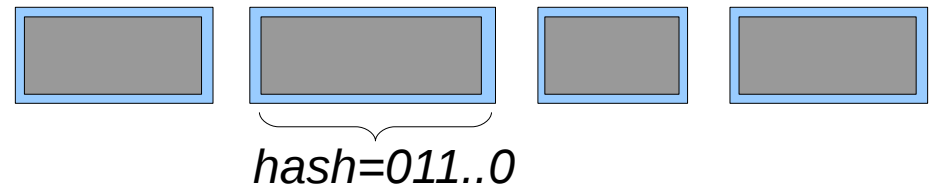
- Satisfies **Scalable secondary storage** requirements
- Started as a research project at *NEC Laboratories America*, in Princeton, NJ
- Successfully commercialized
 - Today: real-world, commercial system
 - Sold by NEC in the US and Japan
- Development of back-end continues at *9LivesData, LLC* in Warsaw, Poland
 - Spinoff from NEC Laboratories

HYDRAsstor functionality

- Content addressable storage (CAS)
- Vast data repository
 - Storing and extracting streams of blocks
 - Single system image built of independent nodes
- Support for standard access methods
 - Filesystem, VTL
- Dynamic capacity sharing
- Self-recovery from failures
- On-demand deletion

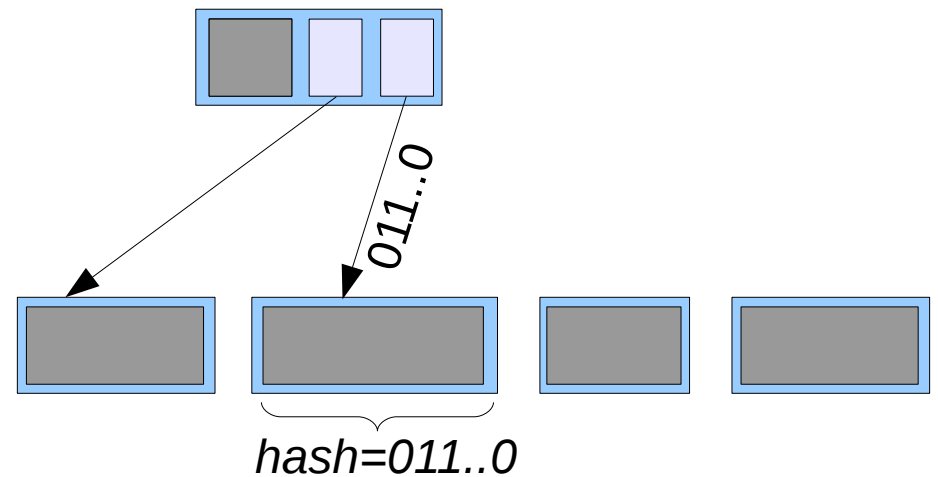
Programming Model

- Repository of blocks
 - Content-addressed
 - Immutable
 - Variable-sized



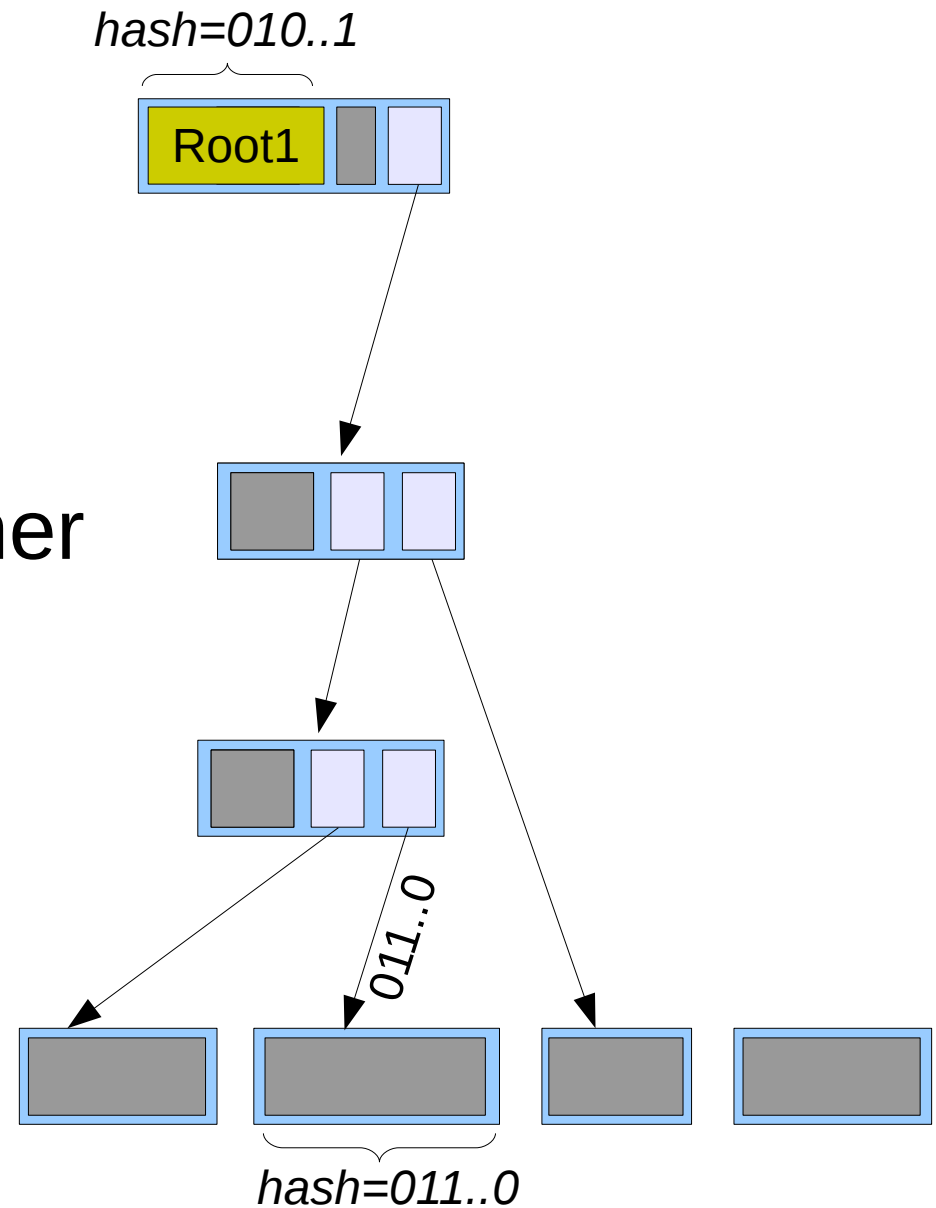
Programming Model

- Repository of blocks
 - Content-addressed
 - Immutable
 - Variable-sized
- Exposed pointers to other blocks



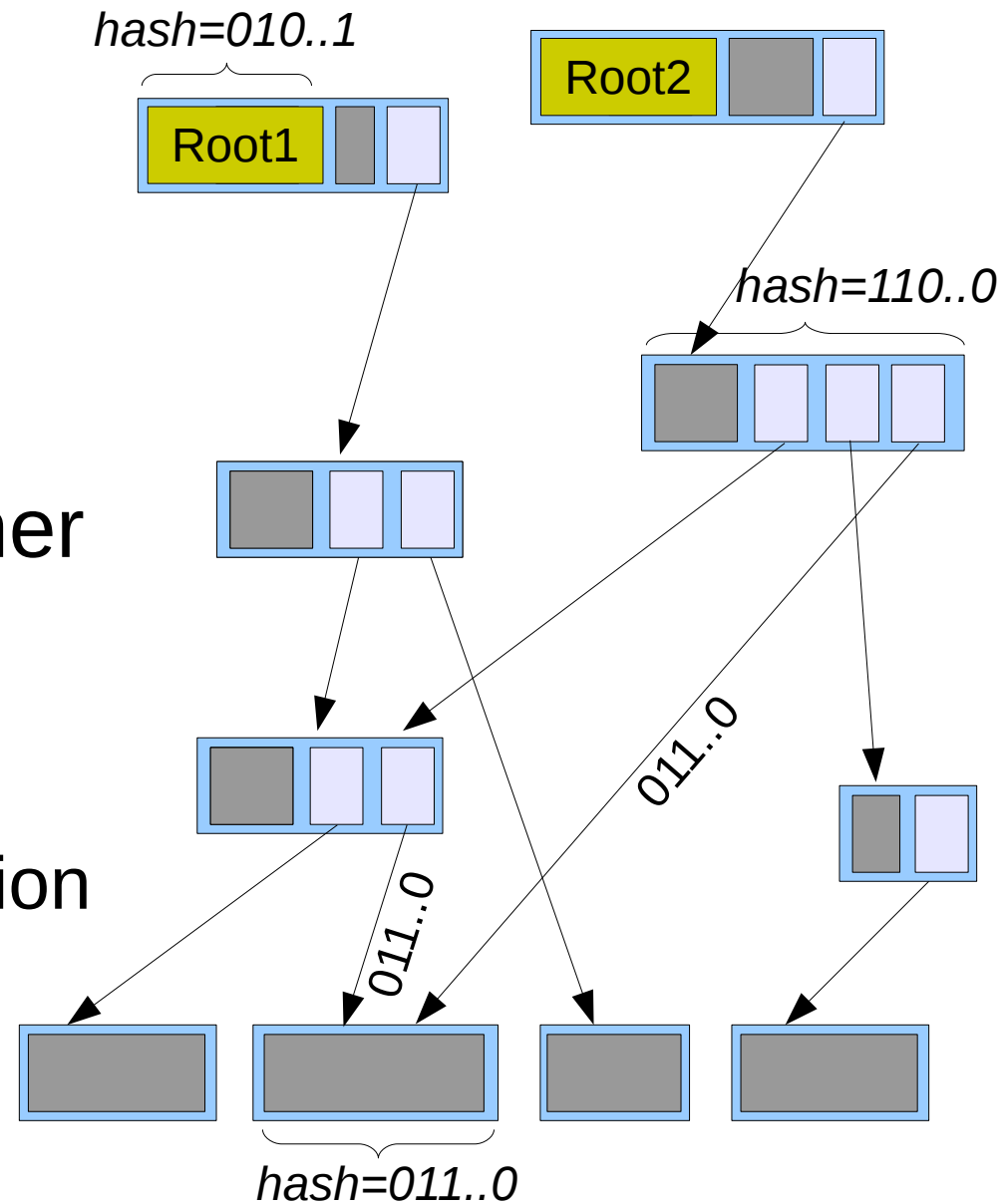
Programming Model

- Repository of blocks
 - Content-addressed
 - Immutable
 - Variable-sized
- Exposed pointers to other blocks
- Trees of blocks



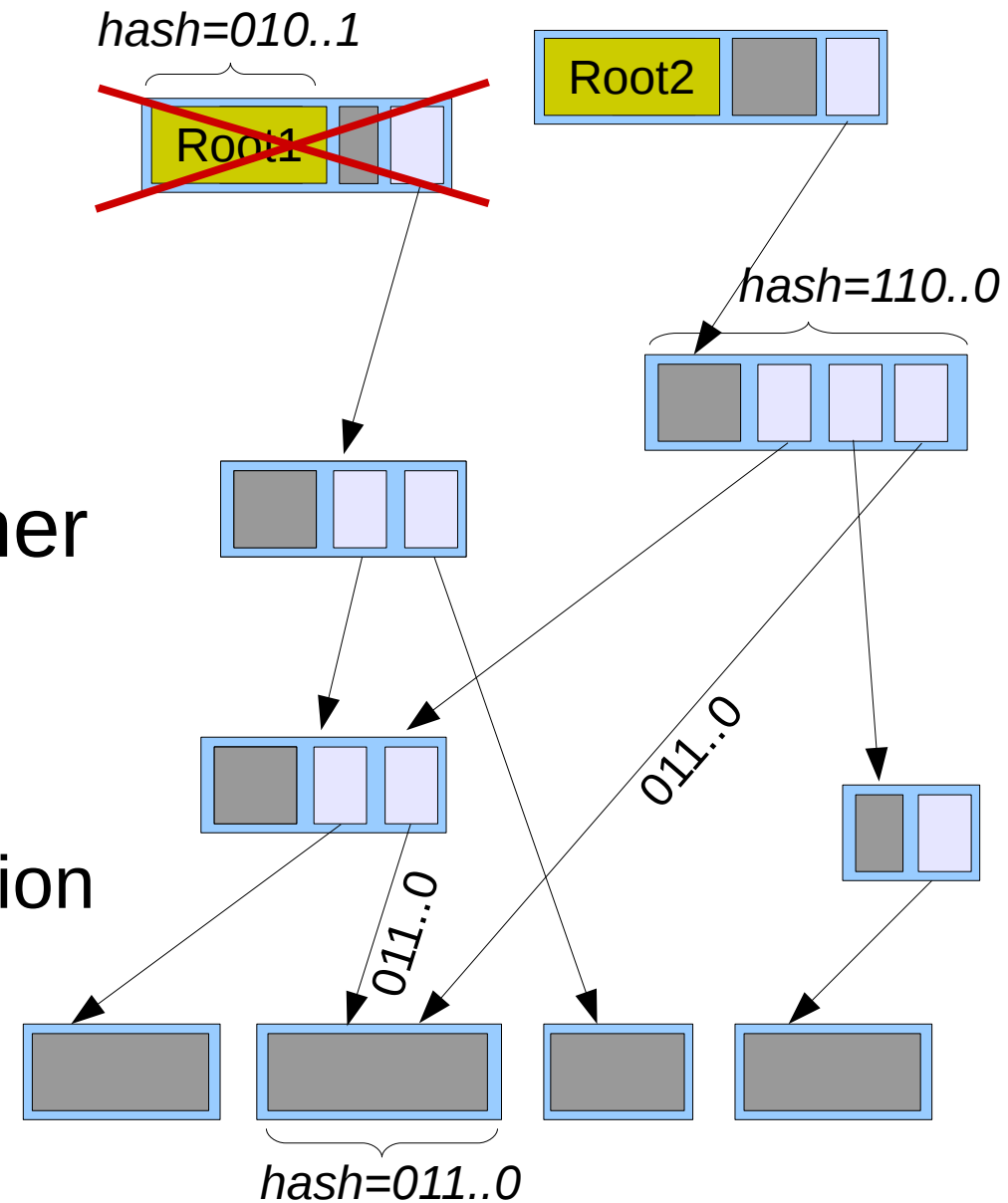
Programming Model

- Repository of blocks
 - Content-addressed
 - Immutable
 - Variable-sized
- Exposed pointers to other blocks
- Trees of blocks
 - DAGs due to deduplication
 - No cycles possible



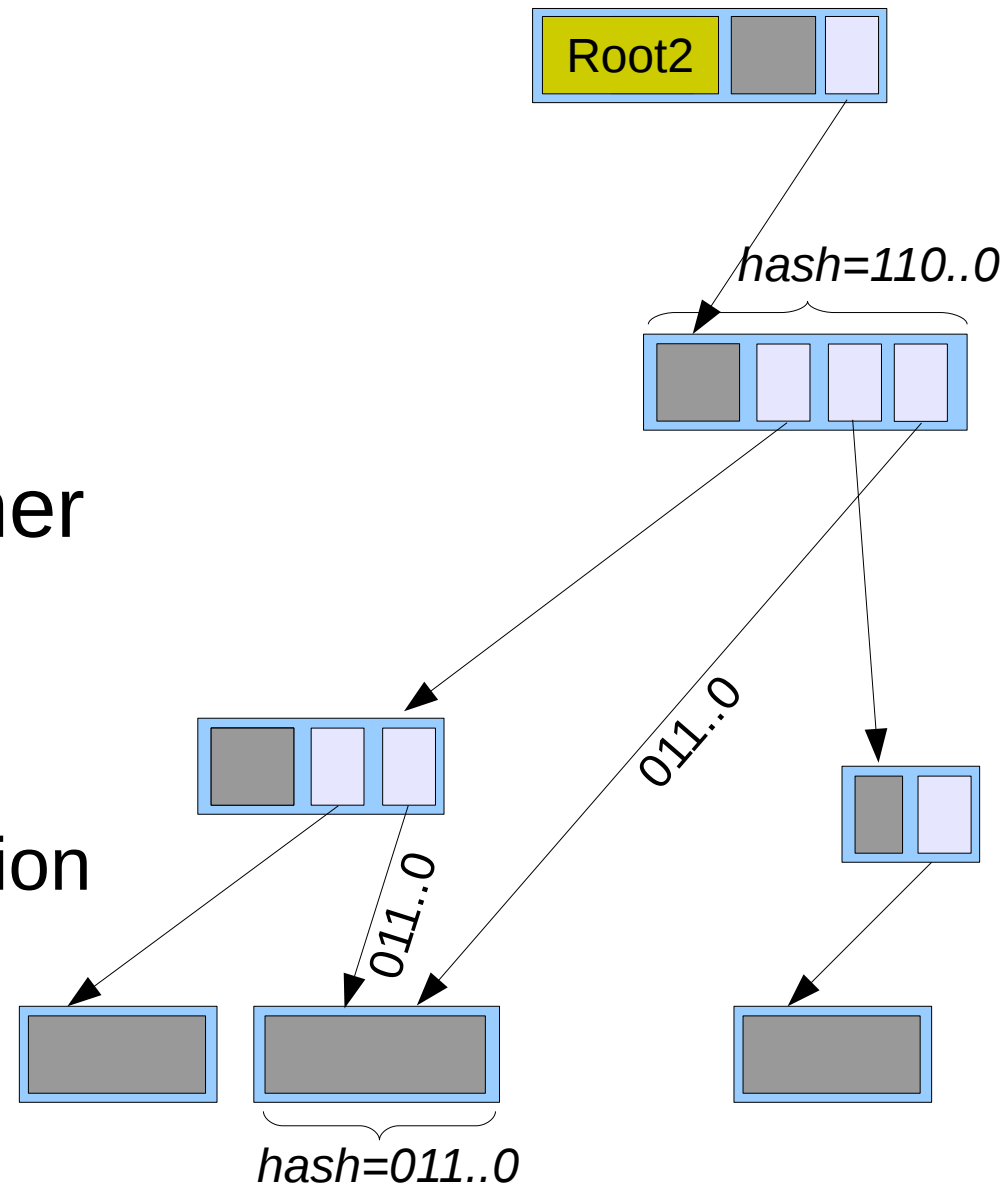
Programming Model

- Repository of blocks
 - Content-addressed
 - Immutable
 - Variable-sized
- Exposed pointers to other blocks
- Trees of blocks
 - DAGs due to deduplication
 - No cycles possible
- Deletion of whole trees



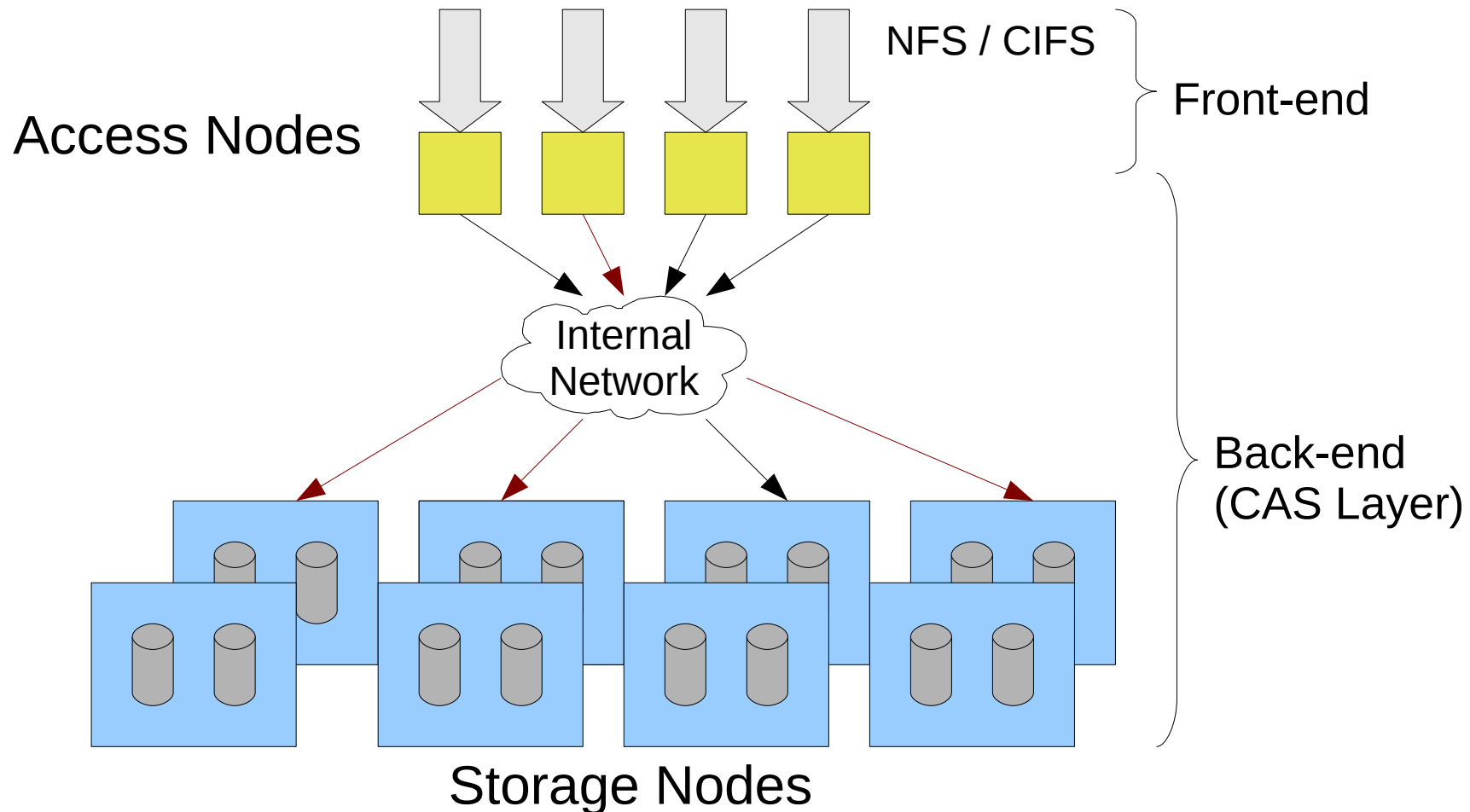
Programming Model

- Repository of blocks
 - Content-addressed
 - Immutable
 - Variable-sized
- Exposed pointers to other blocks
- Trees of blocks
 - DAGs due to deduplication
 - No cycles possible
- Deletion of whole trees



Architecture overview

- Standard server-grade hardware running Linux
- Scalability on data-center level



Data organization: selected requirements

Requirements on scalable storage	Required internal data services
Failure tolerance	<ul style="list-style-type: none">• Identify data resilience reduction• Fast data rebuilding
High performance	<ul style="list-style-type: none">• Preserve locality of data streams• Prefetching
Dynamic scalability	<ul style="list-style-type: none">• Decentralized data management• Load balancing• Fast data transfer to new location
Deduplication	<ul style="list-style-type: none">• Location of potential duplicates• Availability & resiliency verification
On-demand deletion	<ul style="list-style-type: none">• Failure-tolerant, distributed deletion

Data organization: selected requirements

Requirements on scalable storage	Required internal data services
Failure tolerance	<ul style="list-style-type: none">• Identify data resilience reduction• Fast data rebuilding
High performance	<ul style="list-style-type: none">• Preserve locality of data streams• Prefetching
Dynamic scalability	<ul style="list-style-type: none">• Decentralized data management• Load balancing• Fast data transfer to new location
Deduplication	<ul style="list-style-type: none">• Location of potential duplicates• Availability & resiliency verification
On-demand deletion	<ul style="list-style-type: none">• Failure-tolerant, distributed deletion

Data organization: selected requirements

Requirements on scalable storage	Required internal data services
Failure tolerance	<ul style="list-style-type: none">• Identify data resilience reduction• Fast data rebuilding
High performance	<ul style="list-style-type: none">• Preserve locality of data streams• Prefetching
Dynamic scalability	<ul style="list-style-type: none">• Decentralized data management• Load balancing• Fast data transfer to new location
Deduplication	<ul style="list-style-type: none">• Location of potential duplicates• Availability & resiliency verification
On-demand deletion	<ul style="list-style-type: none">• Failure-tolerant, distributed deletion

Data organization: selected requirements

Requirements on scalable storage	Required internal data services
Failure tolerance	<ul style="list-style-type: none">• Identify data resilience reduction• Fast data rebuilding
High performance	<ul style="list-style-type: none">• Preserve locality of data streams• Prefetching
Dynamic scalability	<ul style="list-style-type: none">• Decentralized data management• Load balancing• Fast data transfer to new location
Deduplication	<ul style="list-style-type: none">• Location of potential duplicates• Availability & resiliency verification
On-demand deletion	<ul style="list-style-type: none">• Failure-tolerant, distributed deletion

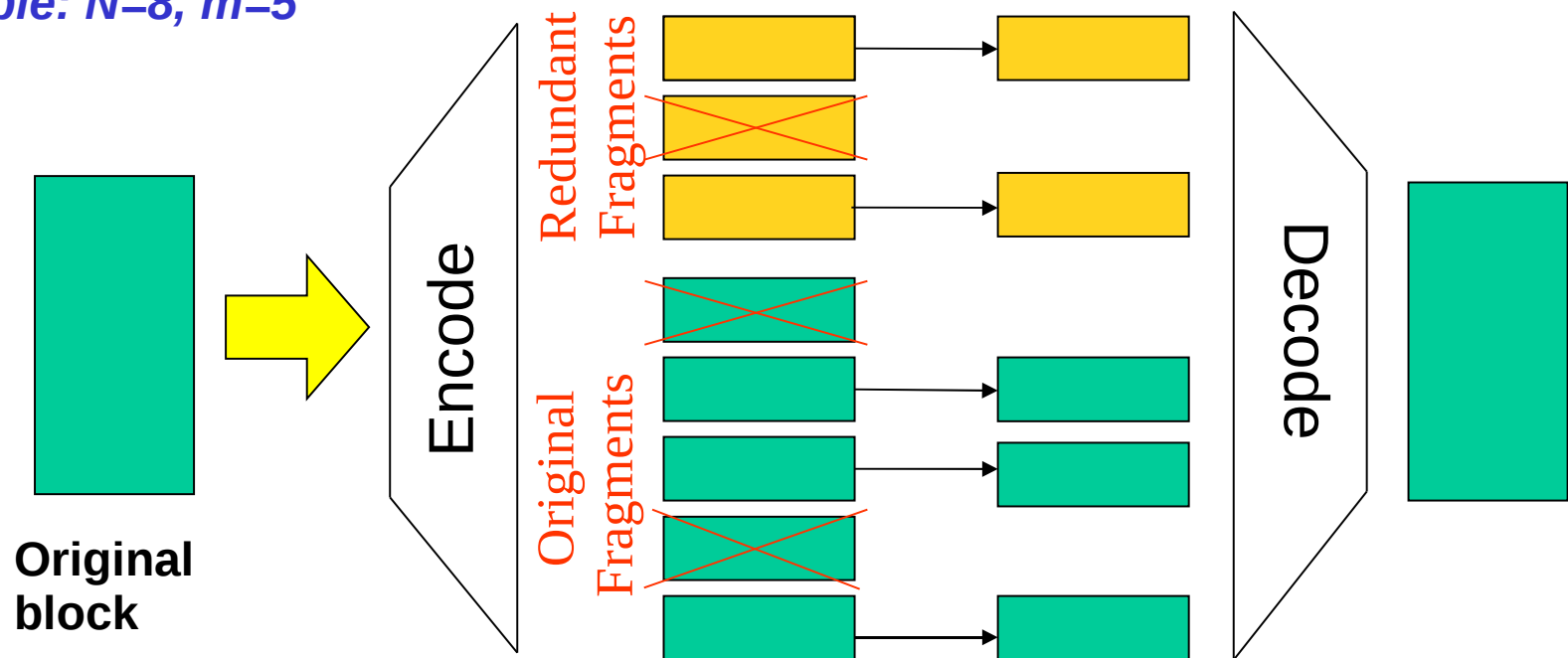
Data organization: selected requirements

Requirements on scalable storage	Required internal data services
Failure tolerance	<ul style="list-style-type: none">• Identify data resilience reduction• Fast data rebuilding
High performance	<ul style="list-style-type: none">• Preserve locality of data streams• Prefetching
Dynamic scalability	<ul style="list-style-type: none">• Decentralized data management• Load balancing• Fast data transfer to new location
Deduplication	<ul style="list-style-type: none">• Location of potential duplicates• Availability & resiliency verification
On-demand deletion	<ul style="list-style-type: none">• Failure-tolerant, distributed deletion

Failure tolerance: erasure coding

- Block erasure-coded into N fragments
- Storage overhead tunable

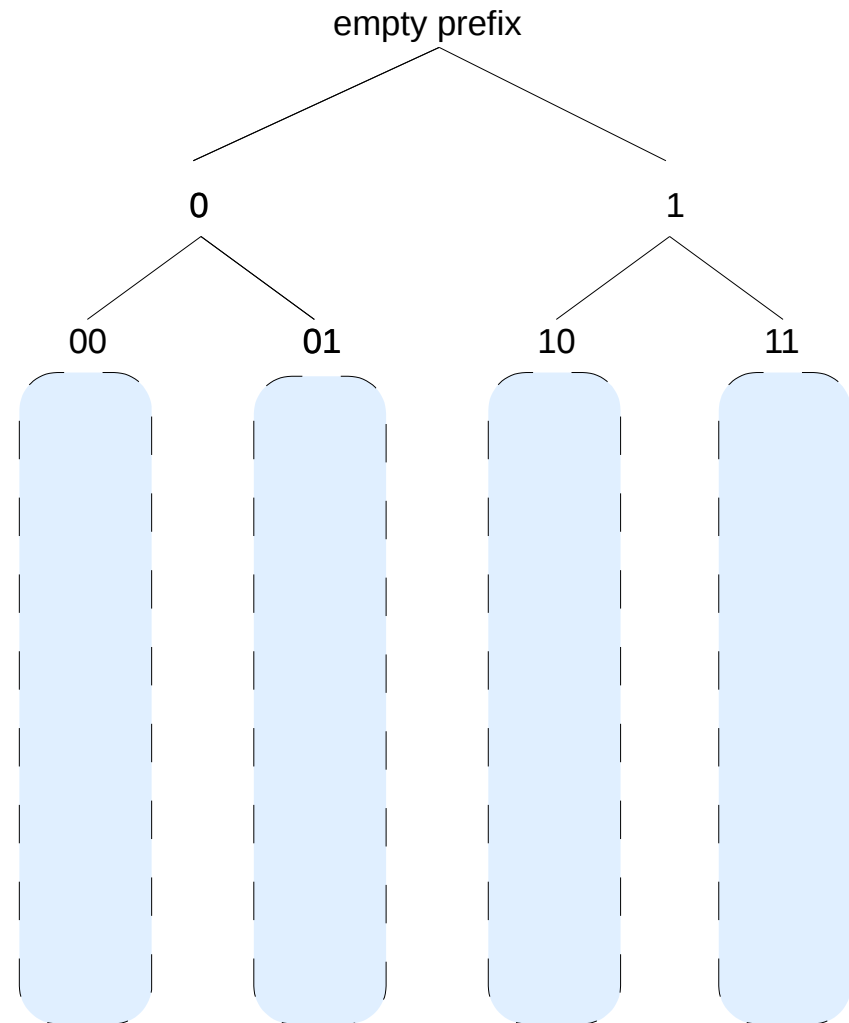
Example: N=8, m=5



Any 3 fragments can be lost

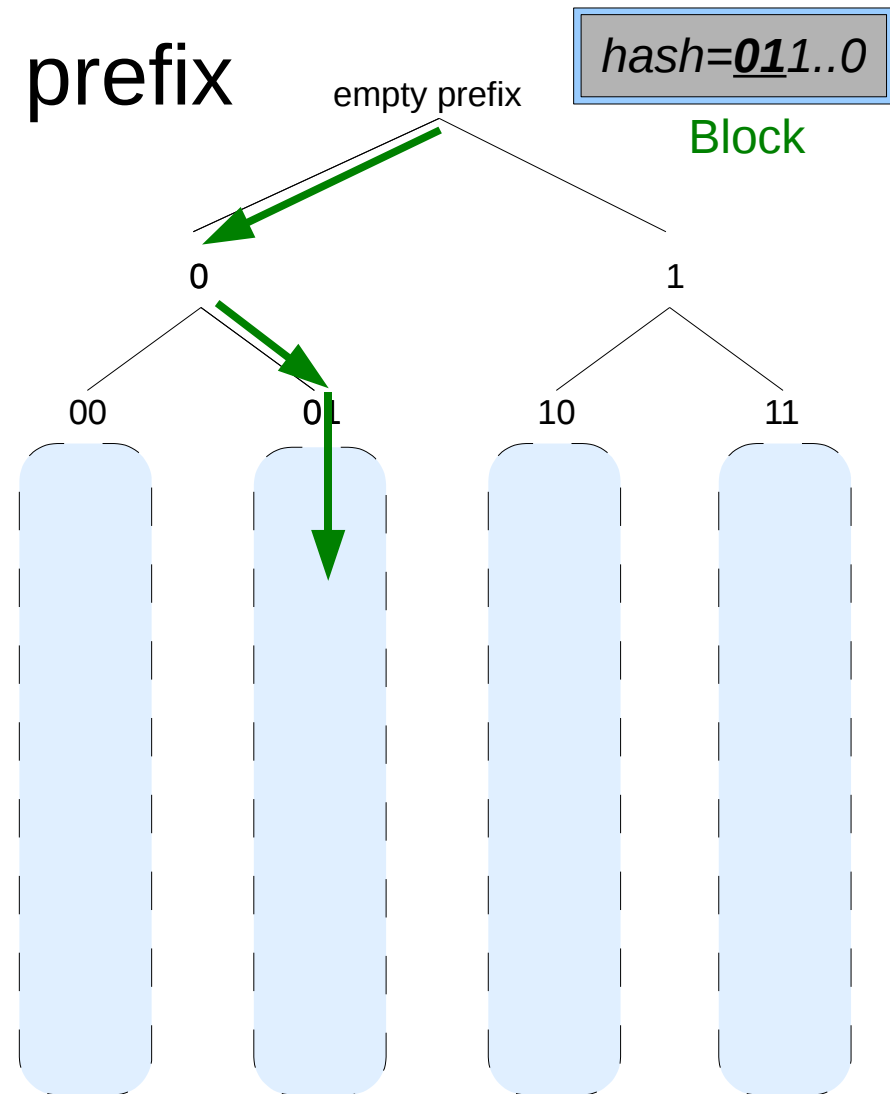
Scalability with DHT: data placement

- Block location: DHT with prefix routing



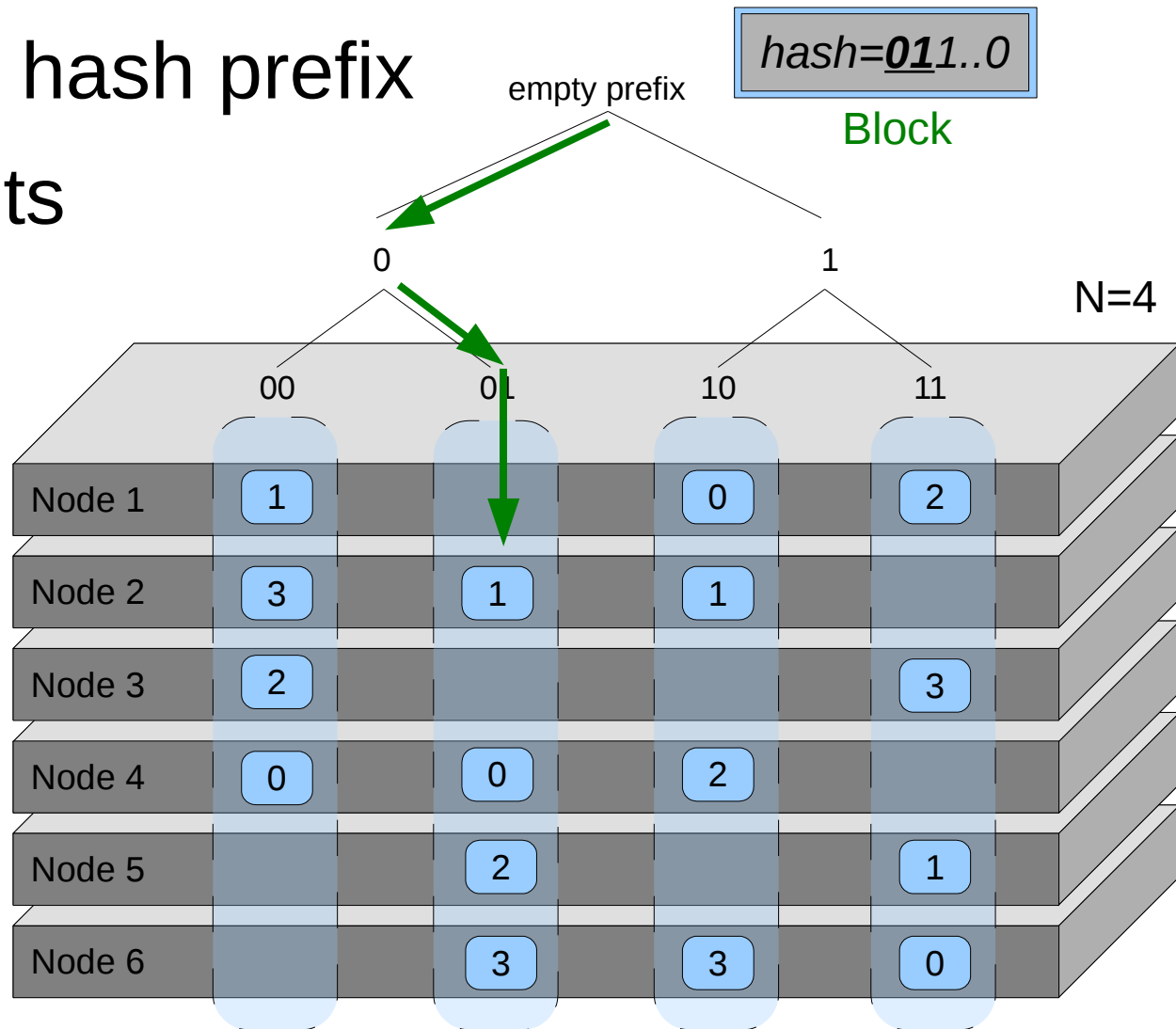
Scalability with DHT: data placement

- Block location: DHT with prefix routing
- Block mapped to hash prefix



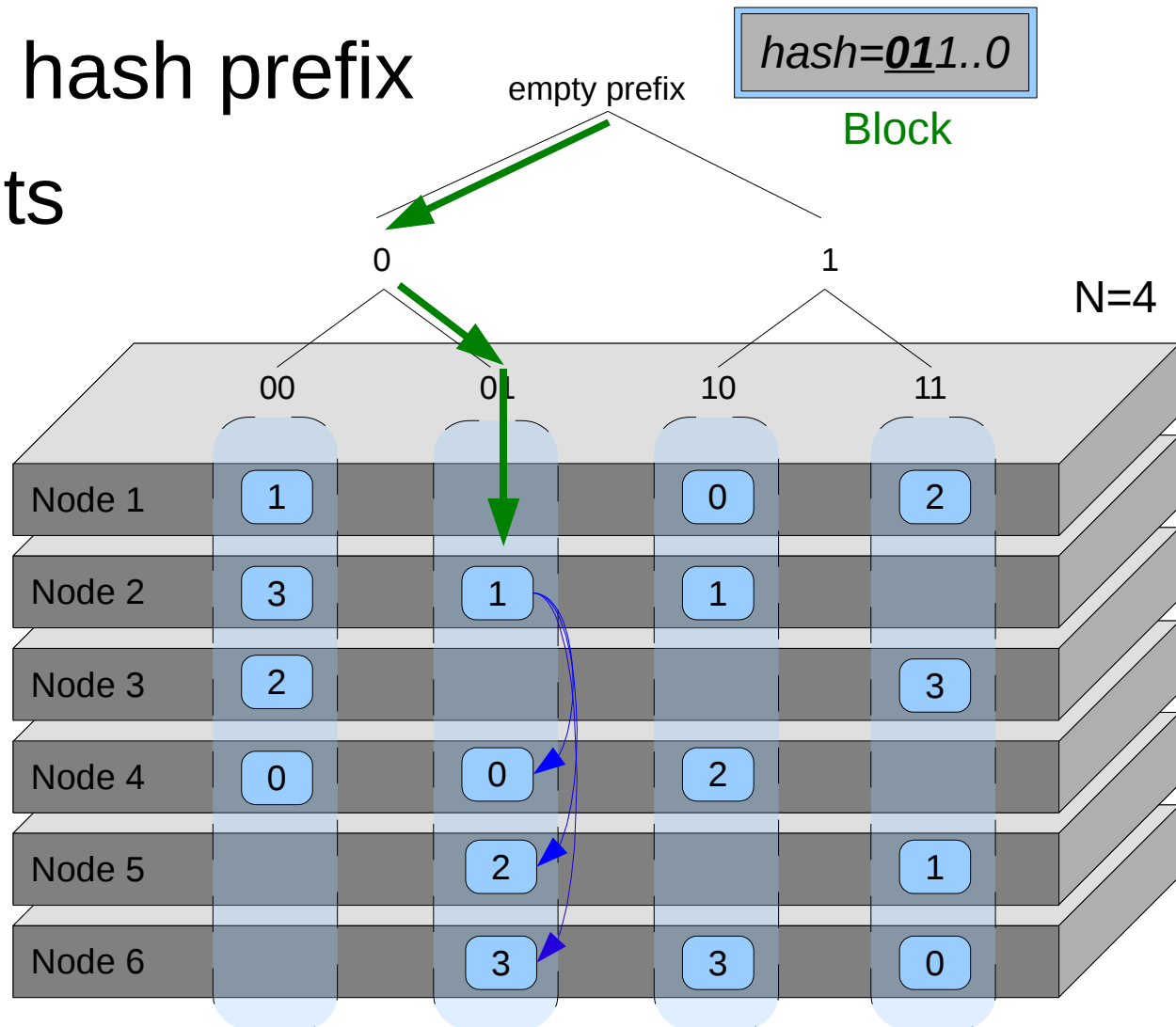
Scalability with DHT: data placement

- Block location: DHT with prefix routing
- Block mapped to hash prefix
- Prefix components
 - Hosted on SNs
 - N components per prefix



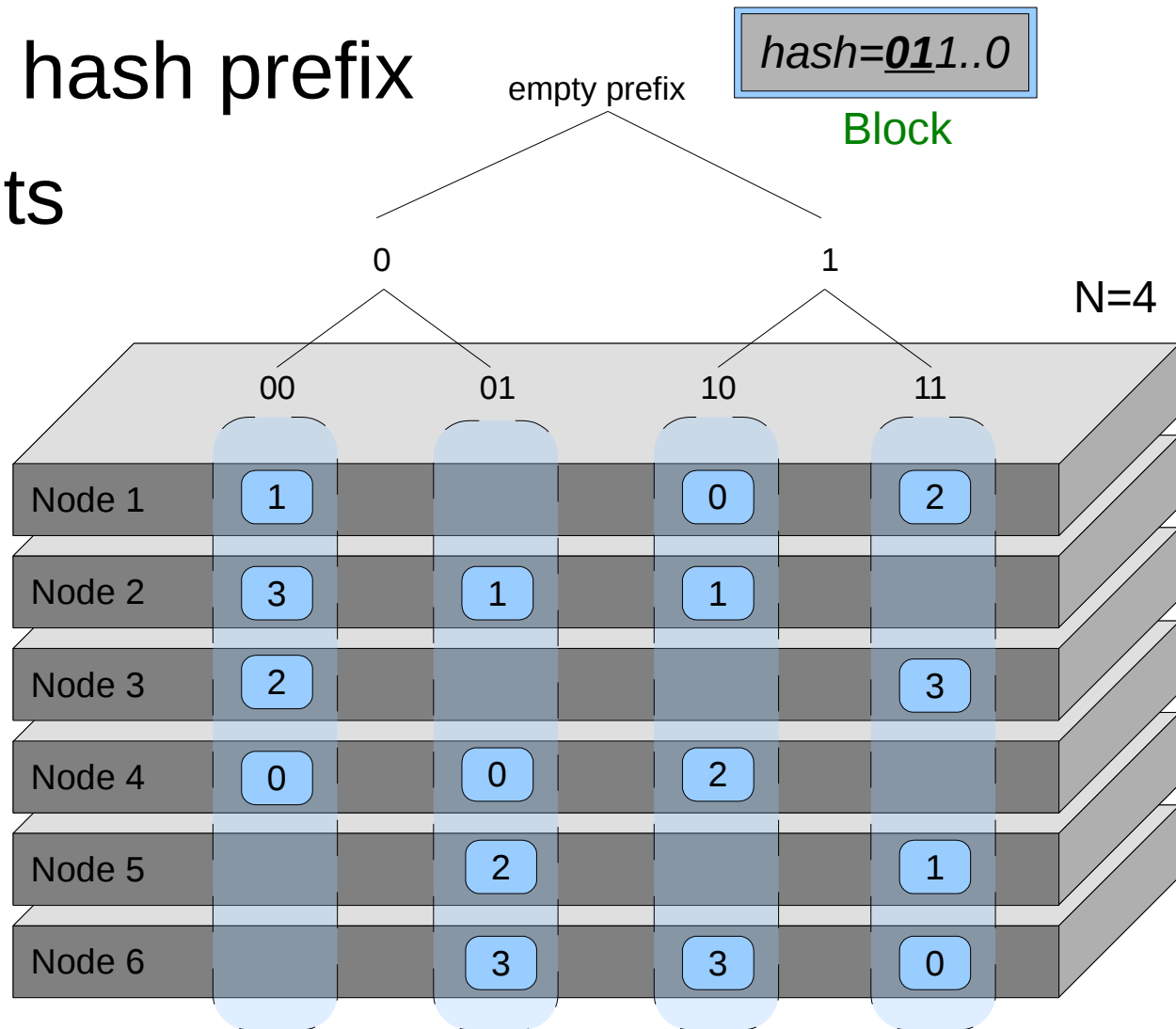
Scalability with DHT: data placement

- Block location: DHT with prefix routing
- Block mapped to hash prefix
- Prefix components
 - Hosted on SNs
 - N components per prefix
 - Store fragments



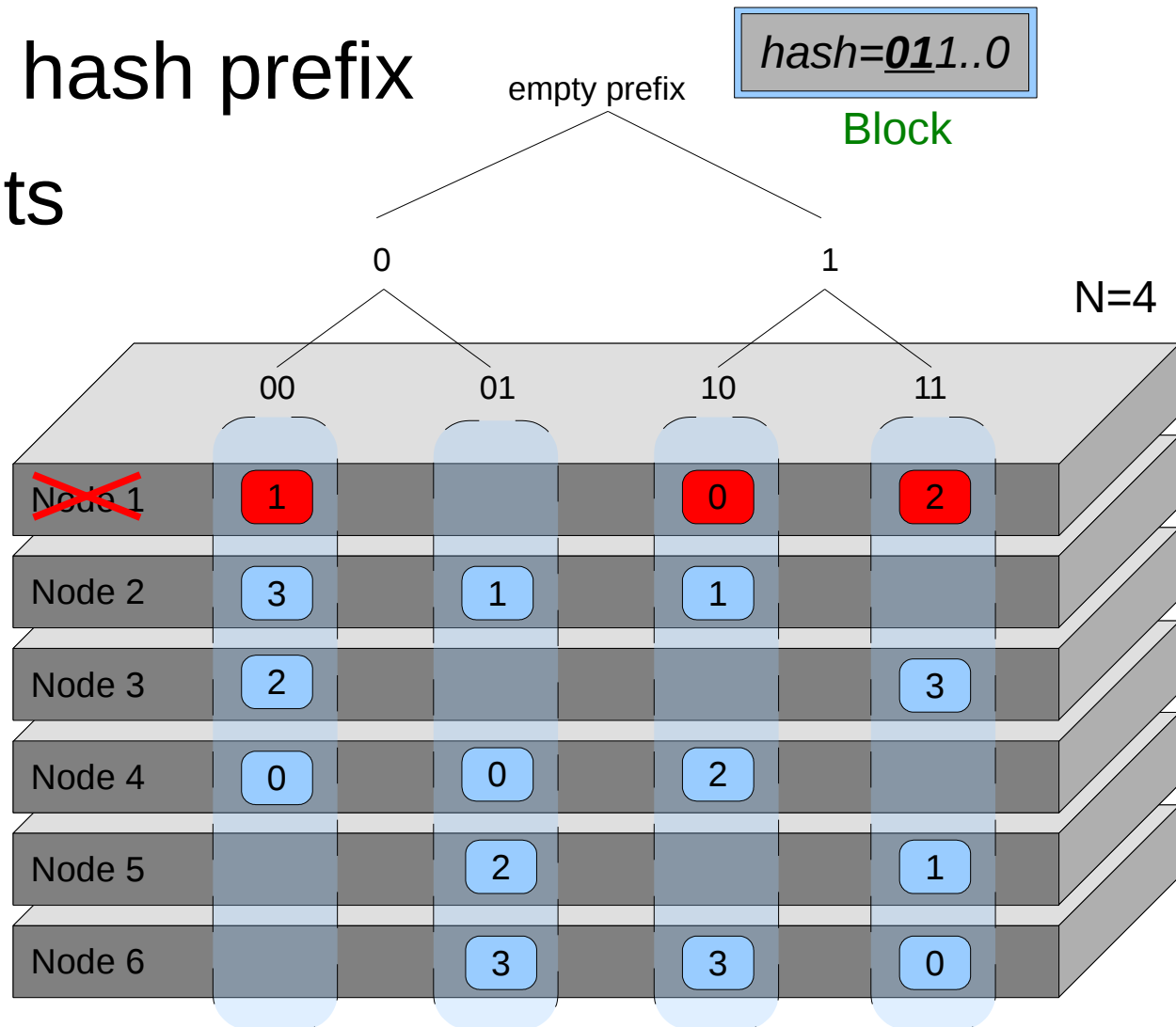
Scalability with DHT: data placement

- Block location: DHT with prefix routing
- Block mapped to hash prefix
- Prefix components
 - Hosted on SNs
 - N components per prefix
 - Store fragments
- Distributed consensus



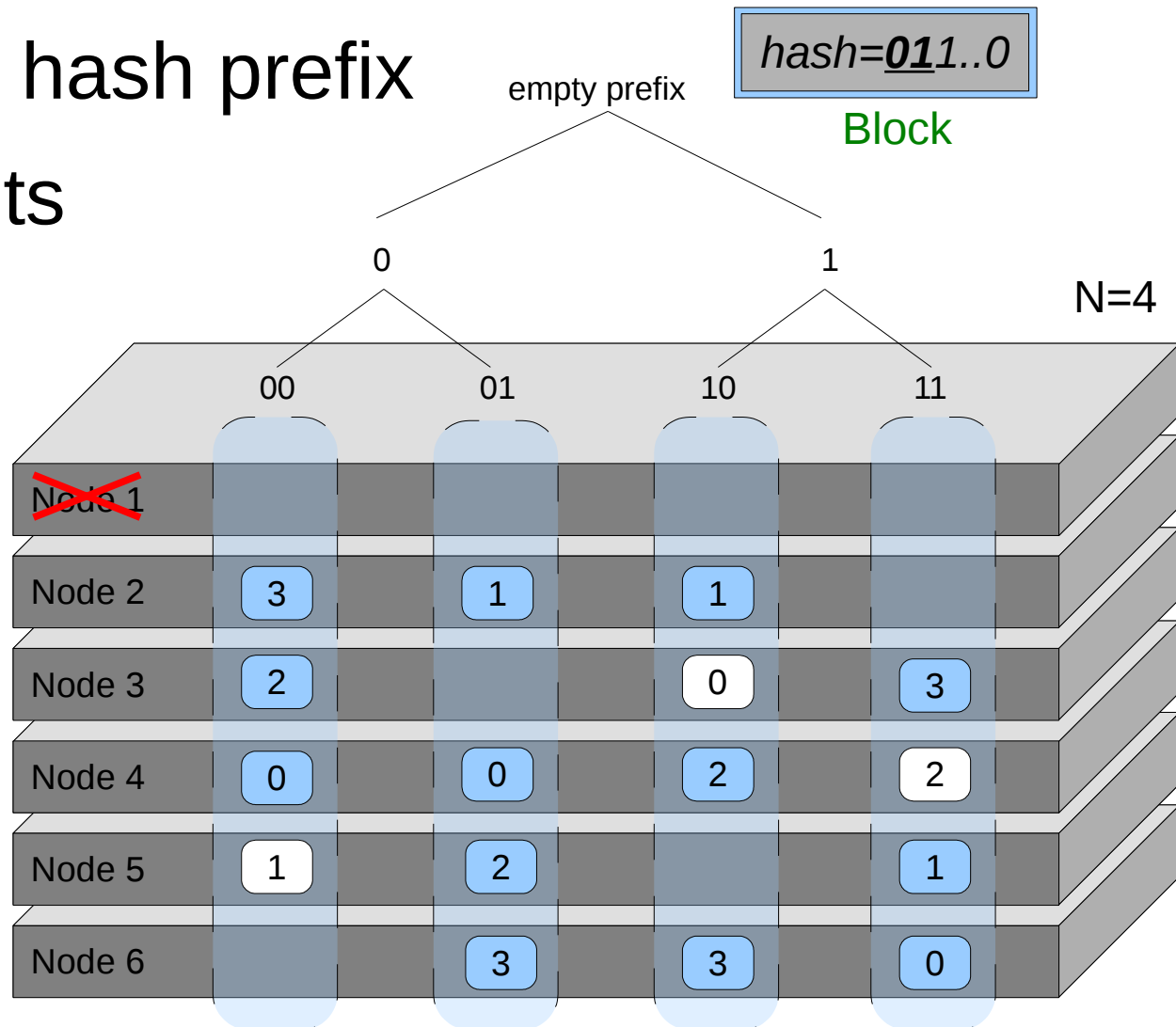
Scalability with DHT: data placement

- Block location: DHT with prefix routing
- Block mapped to hash prefix
- Prefix components
 - Hosted on SNs
 - N components per prefix
 - Store fragments
- Distributed consensus



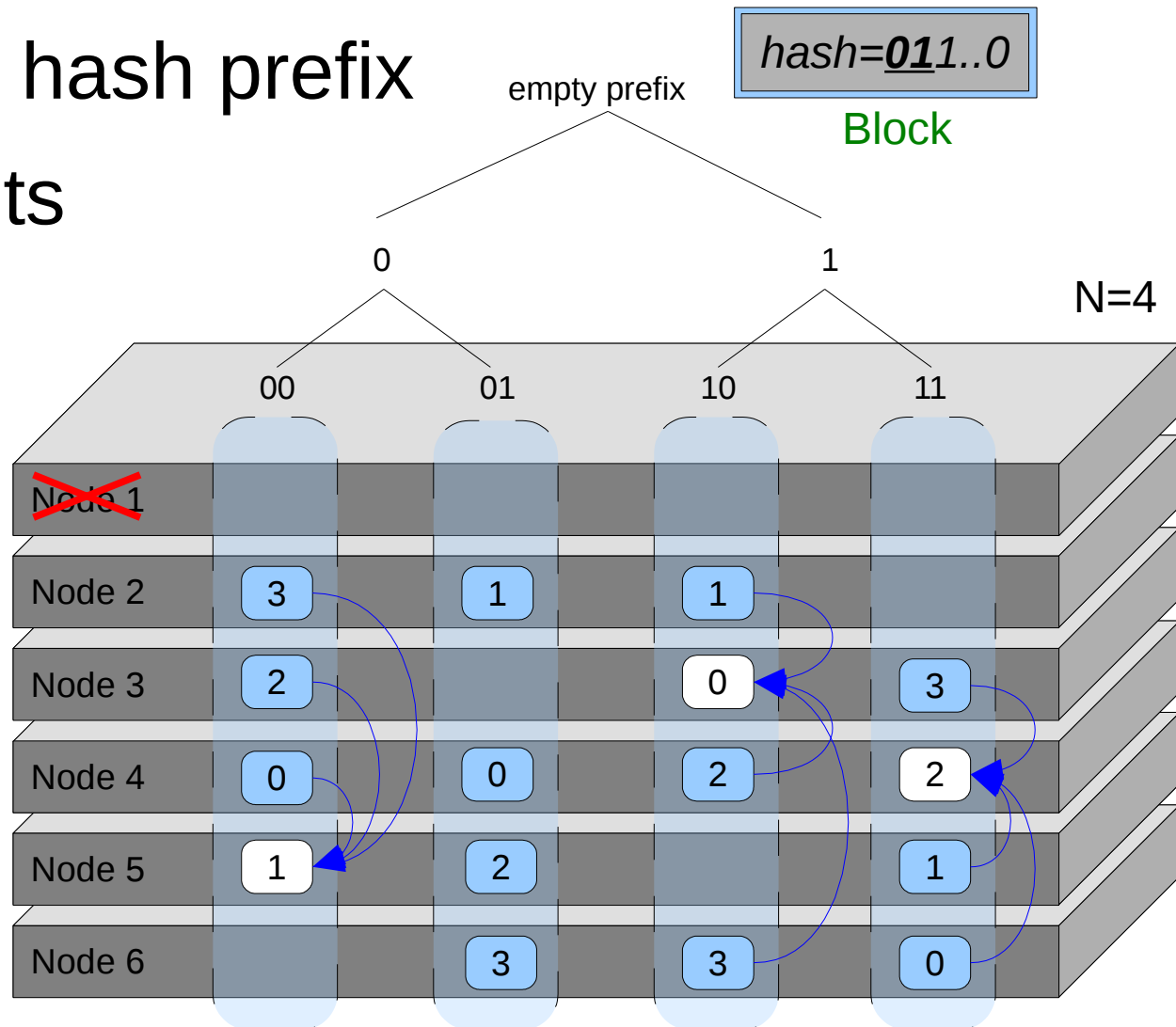
Scalability with DHT: data placement

- Block location: DHT with prefix routing
- Block mapped to hash prefix
- Prefix components
 - Hosted on SNs
 - N components per prefix
 - Store fragments
- Distributed consensus



Scalability with DHT: data placement

- Block location: DHT with prefix routing
- Block mapped to hash prefix
- Prefix components
 - Hosted on SNs
 - N components per prefix
 - Store fragments
- Distributed consensus

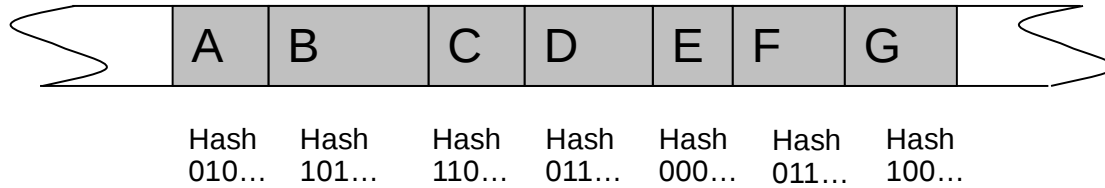


Data organization: synchron chains



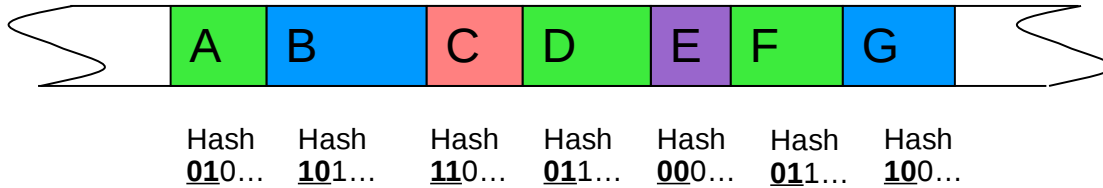
- Data stream split to blocks

Data organization: synchron chains



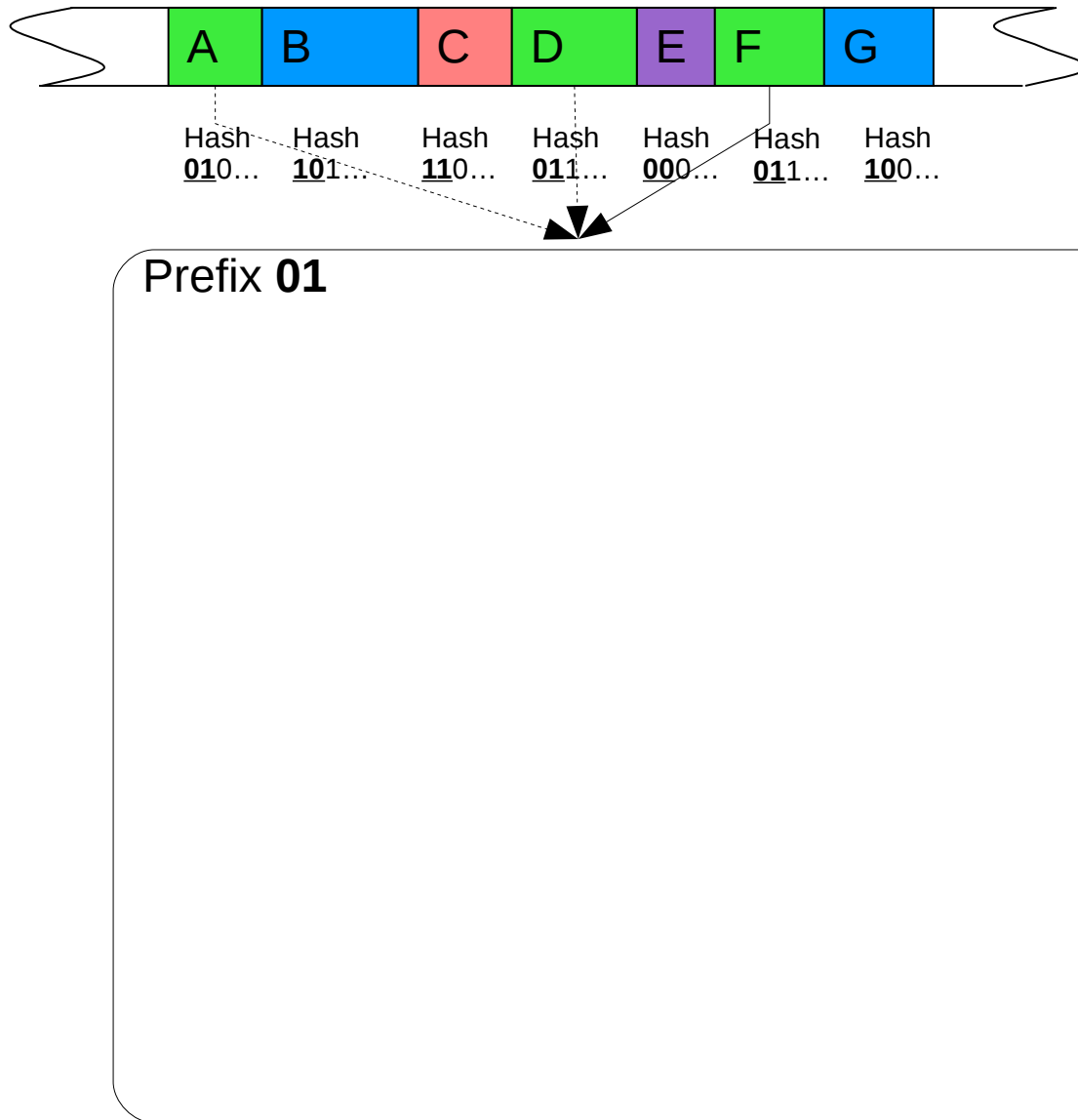
- Data stream split to blocks
- Hashes of blocks computed

Data organization: synchronun chains



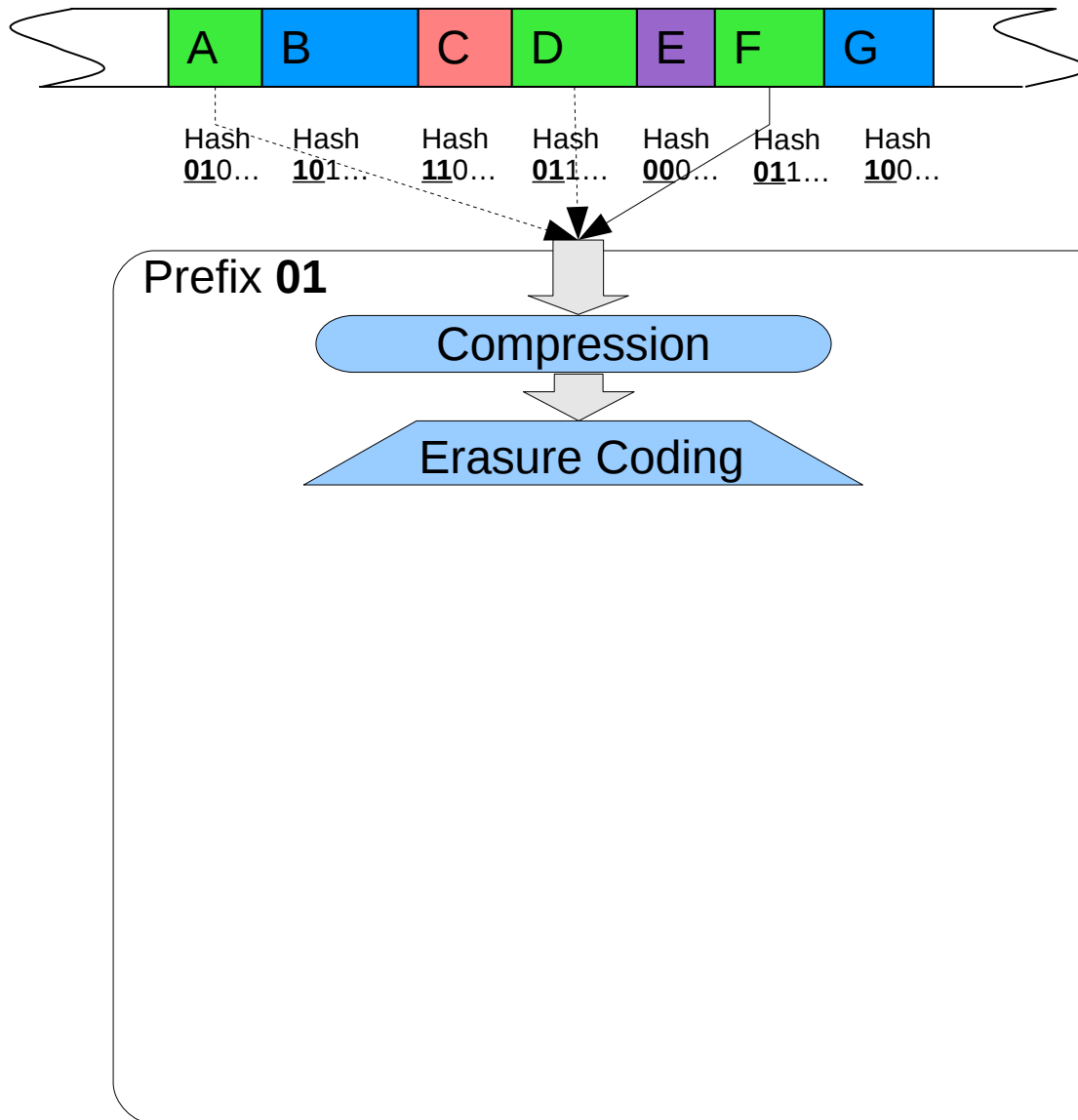
- Data stream split to blocks
- Hashes of blocks computed
- Routing through DHT

Data organization: synchron chains



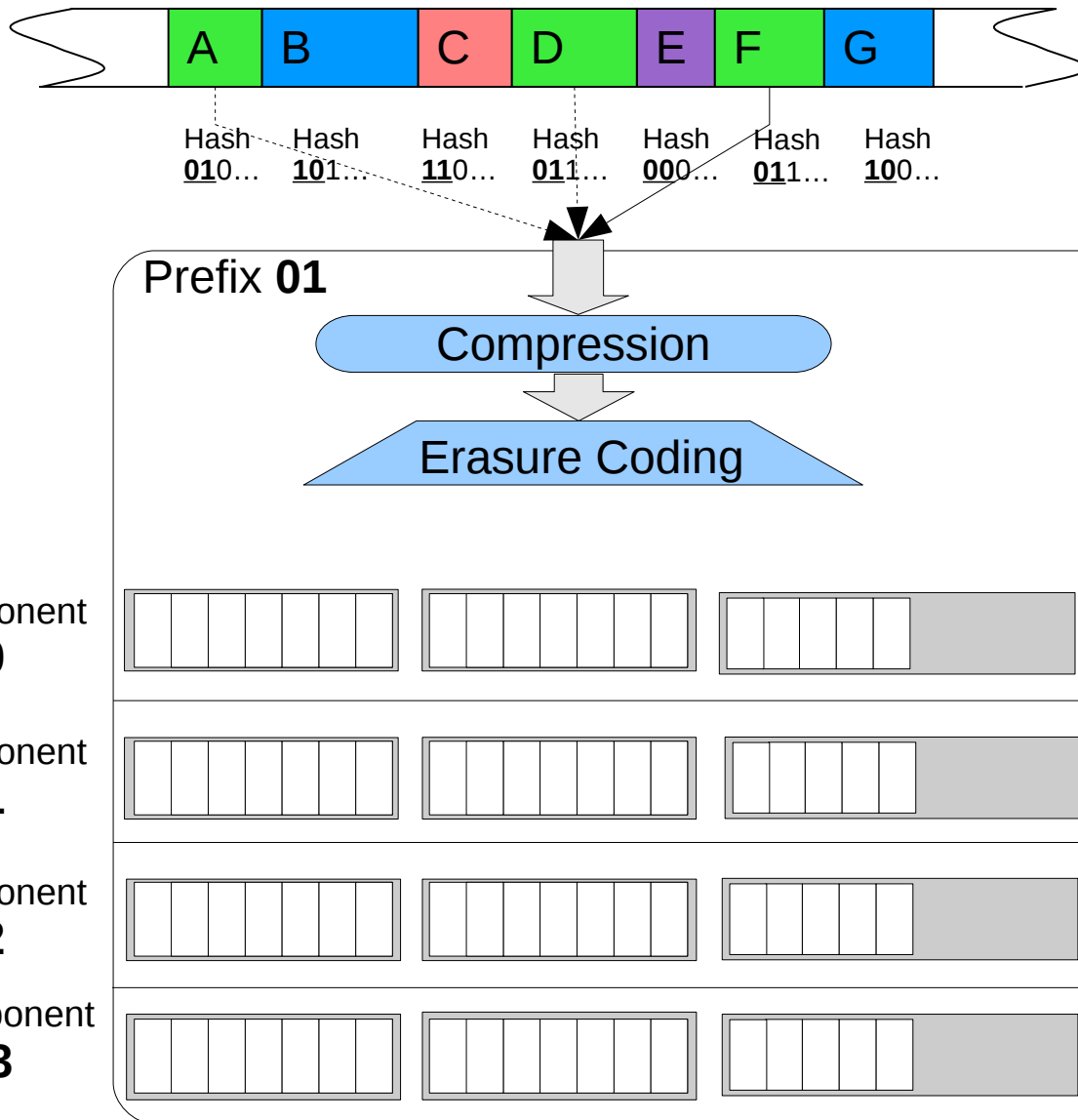
- Data stream split to blocks
- Hashes of blocks computed
- Routing through DHT

Data organization: synchron chains



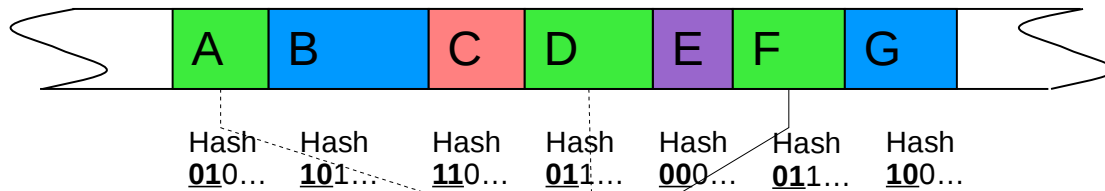
- Data stream split to blocks
- Hashes of blocks computed
- Routing through DHT

Data organization: synchronun chains

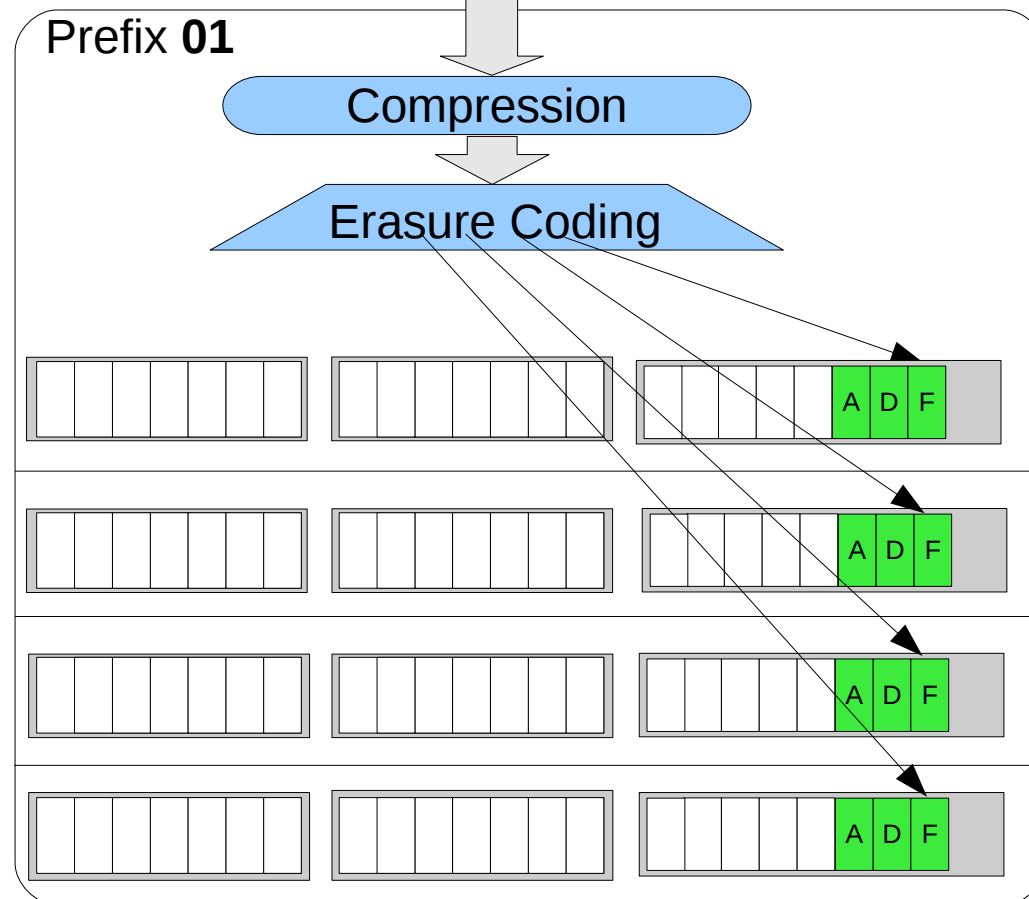


- Data stream split to blocks
- Hashes of blocks computed
- Routing through DHT
- Erasure-coded **fragments** stored by components

Data organization: synchronun chains

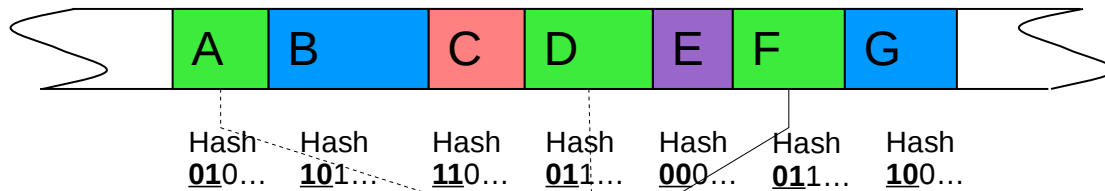


- Data stream split to blocks
- Hashes of blocks computed
- Routing through DHT

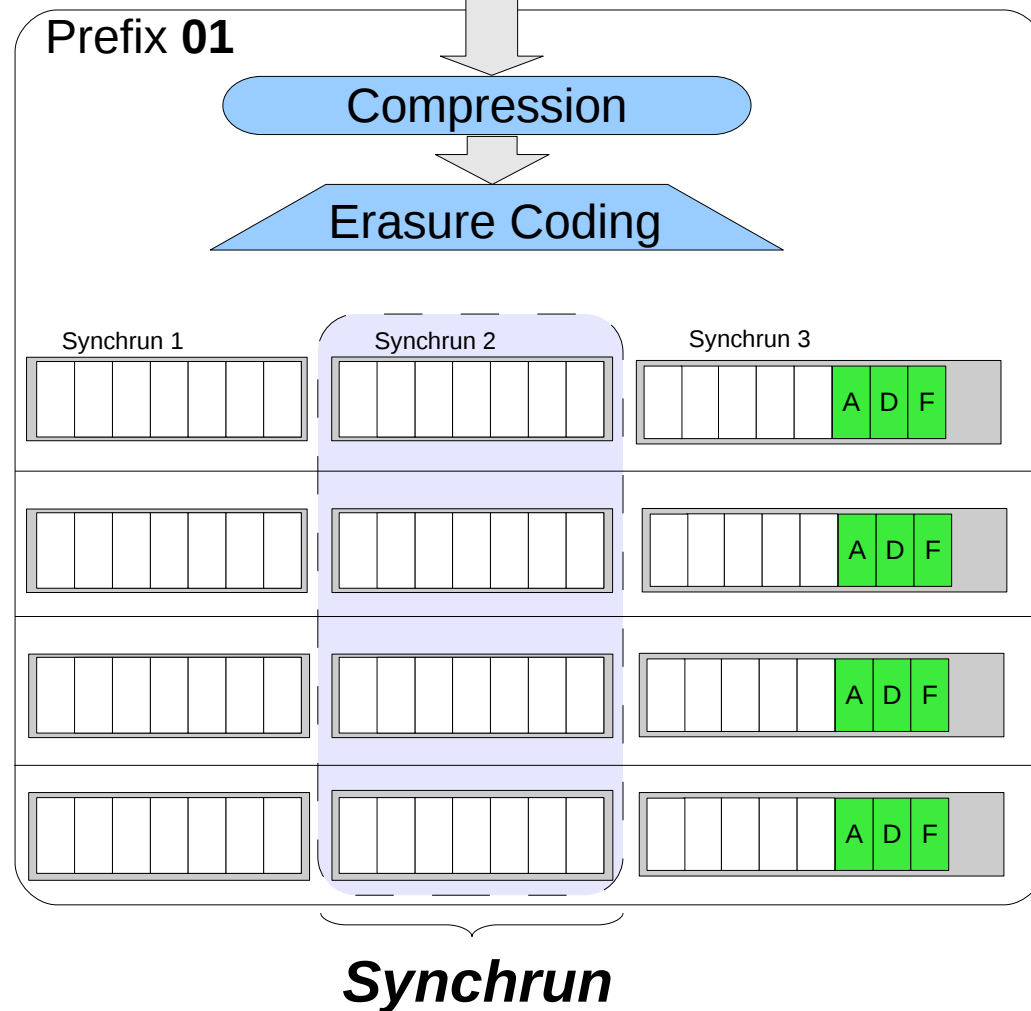


- Erasure-coded **fragments** stored by components

Data organization: synchronun chains

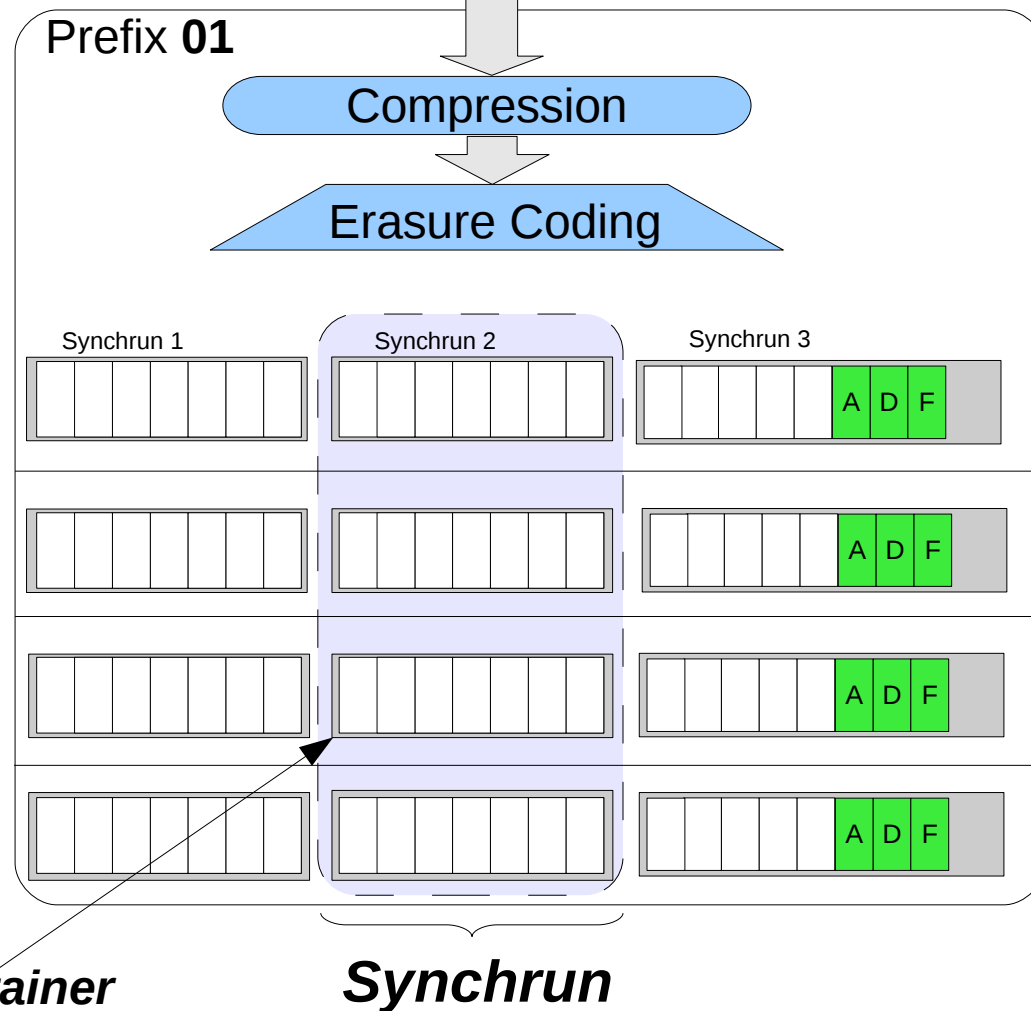
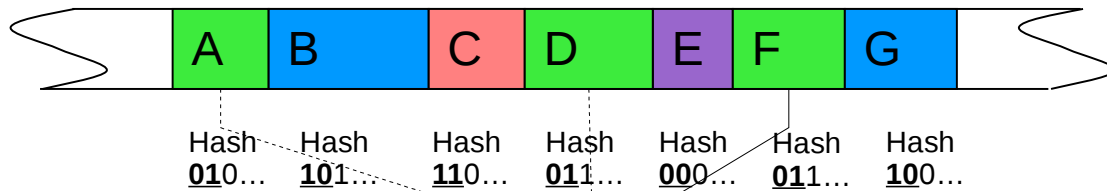


- Data stream split to blocks
- Hashes of blocks computed
- Routing through DHT



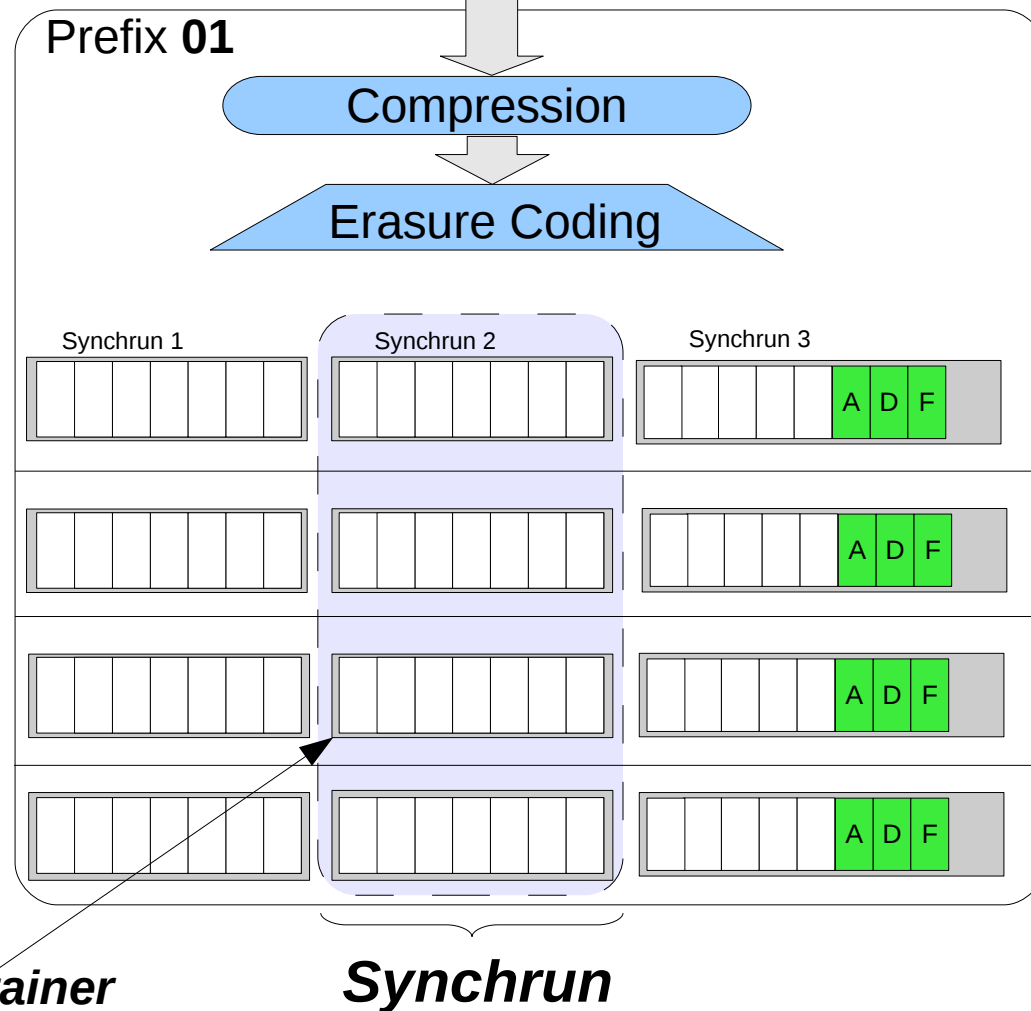
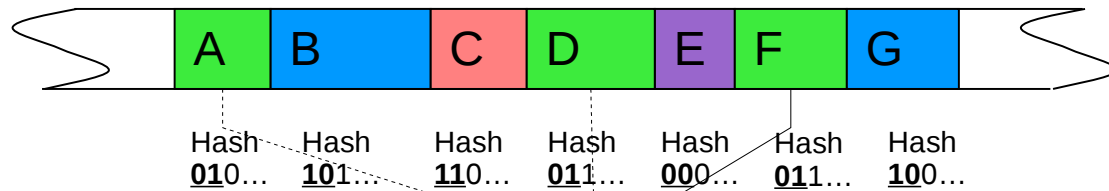
- Erasure-coded **fragments** stored by components
- Grouped into **synchronuns**

Data organization: synchronun chains



- Data stream split to blocks
- Hashes of blocks computed
- Routing through DHT
- Erasure-coded **fragments** stored by components
- Grouped into **synchronuns**
- **Containers** stored on disks
 - Fragment metadata separately from data

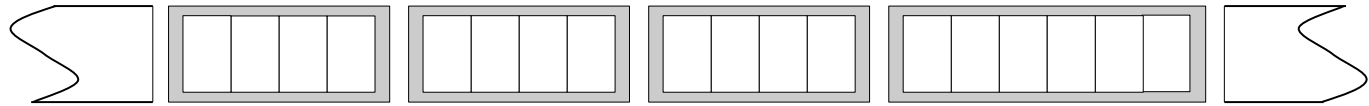
Data organization: synchronun chains



- Data stream split to blocks
- Hashes of blocks computed
- Routing through DHT
- Erasure-coded **fragments** stored by components
- Grouped into **synchronuns**
- **Containers** stored on disks
 - Fragment metadata separately from data
- Ordered synchronun **chains**
 - Preserve order & locality
 - Manageable

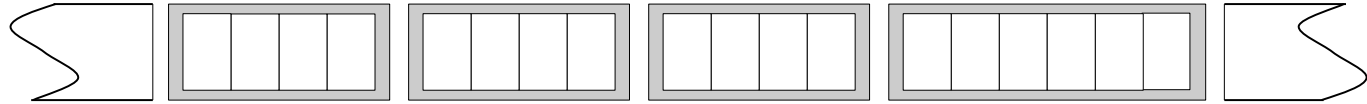
Synchrún chains in a dynamic system

Component
01:1



System growth: split

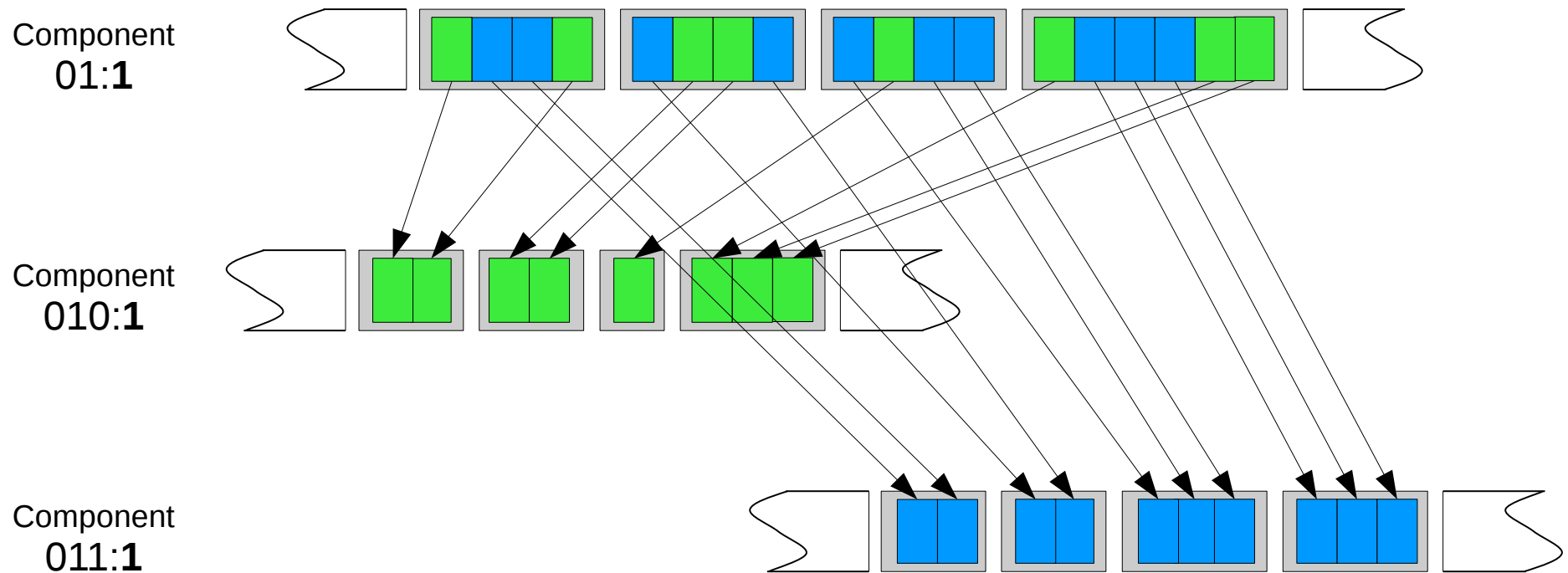
Component
01:1



Component
010:1

Component
011:1

System growth: split

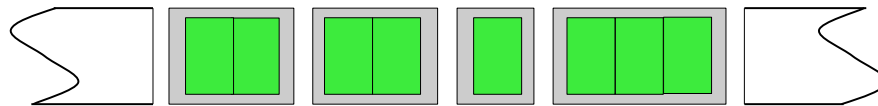


System growth: split

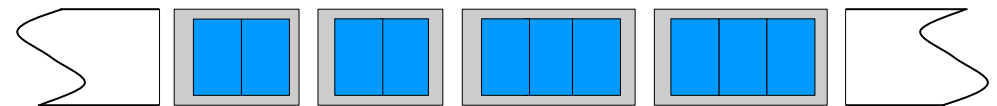
Component
01:1



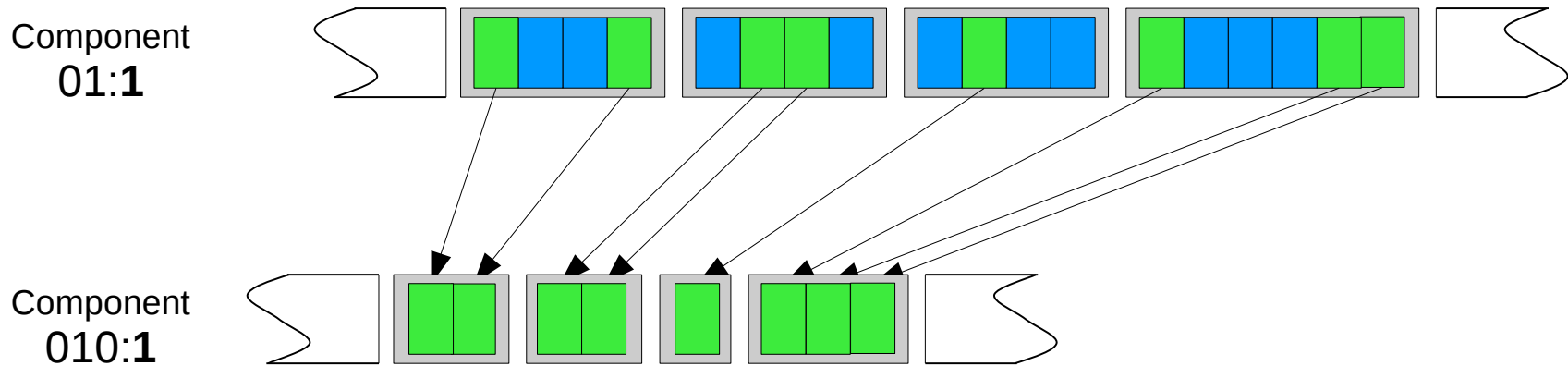
Component
010:1



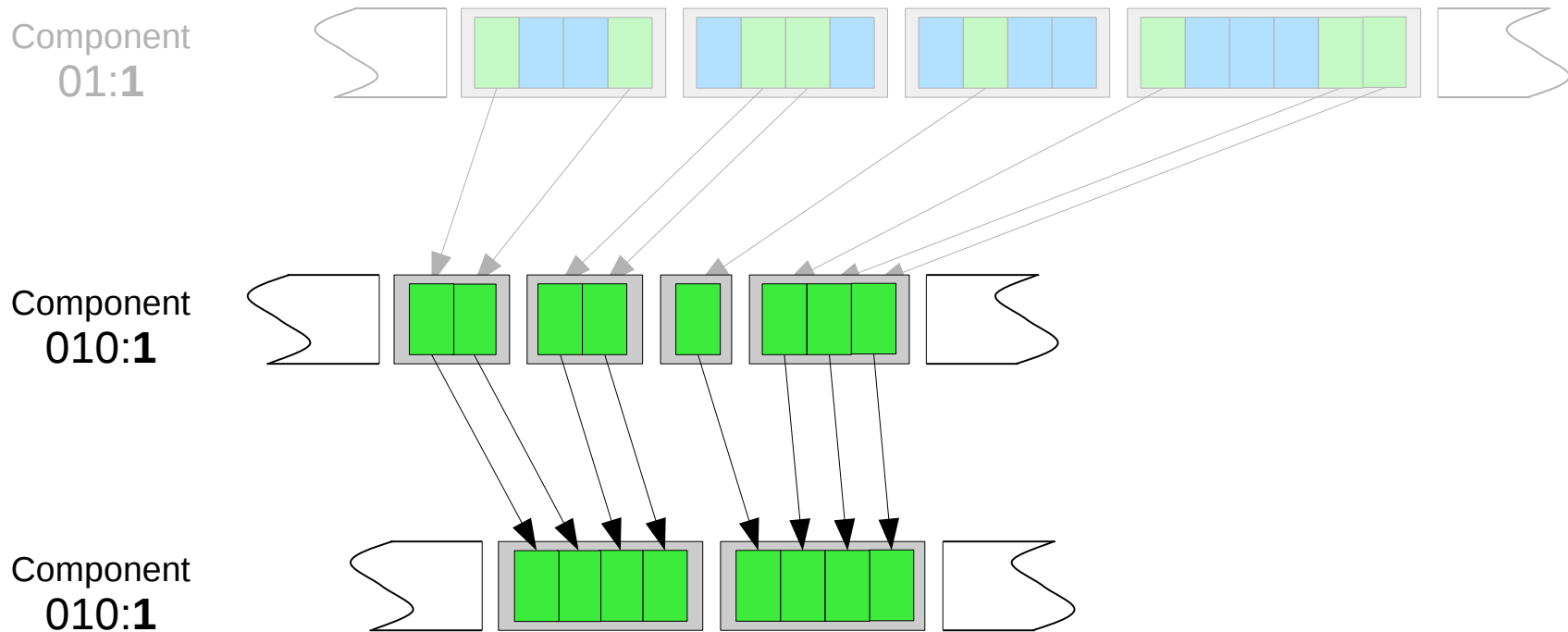
Component
011:1



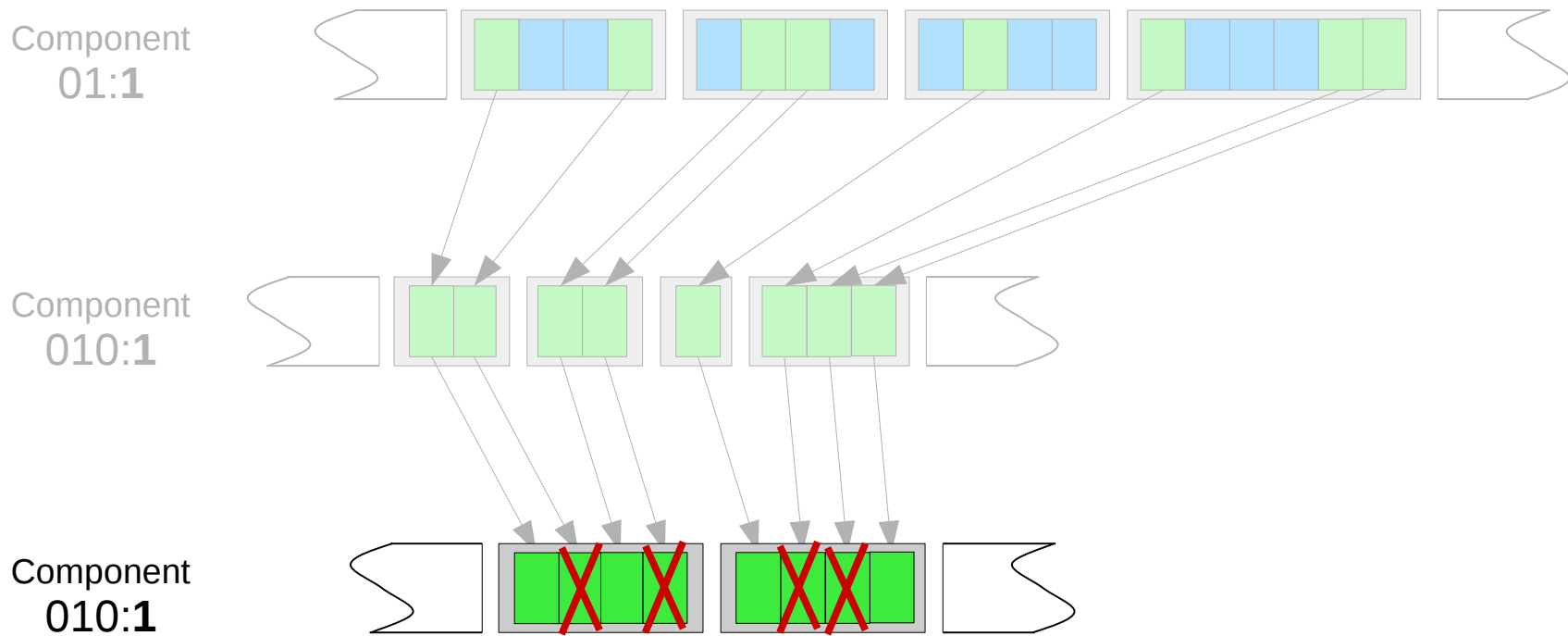
Concatenation



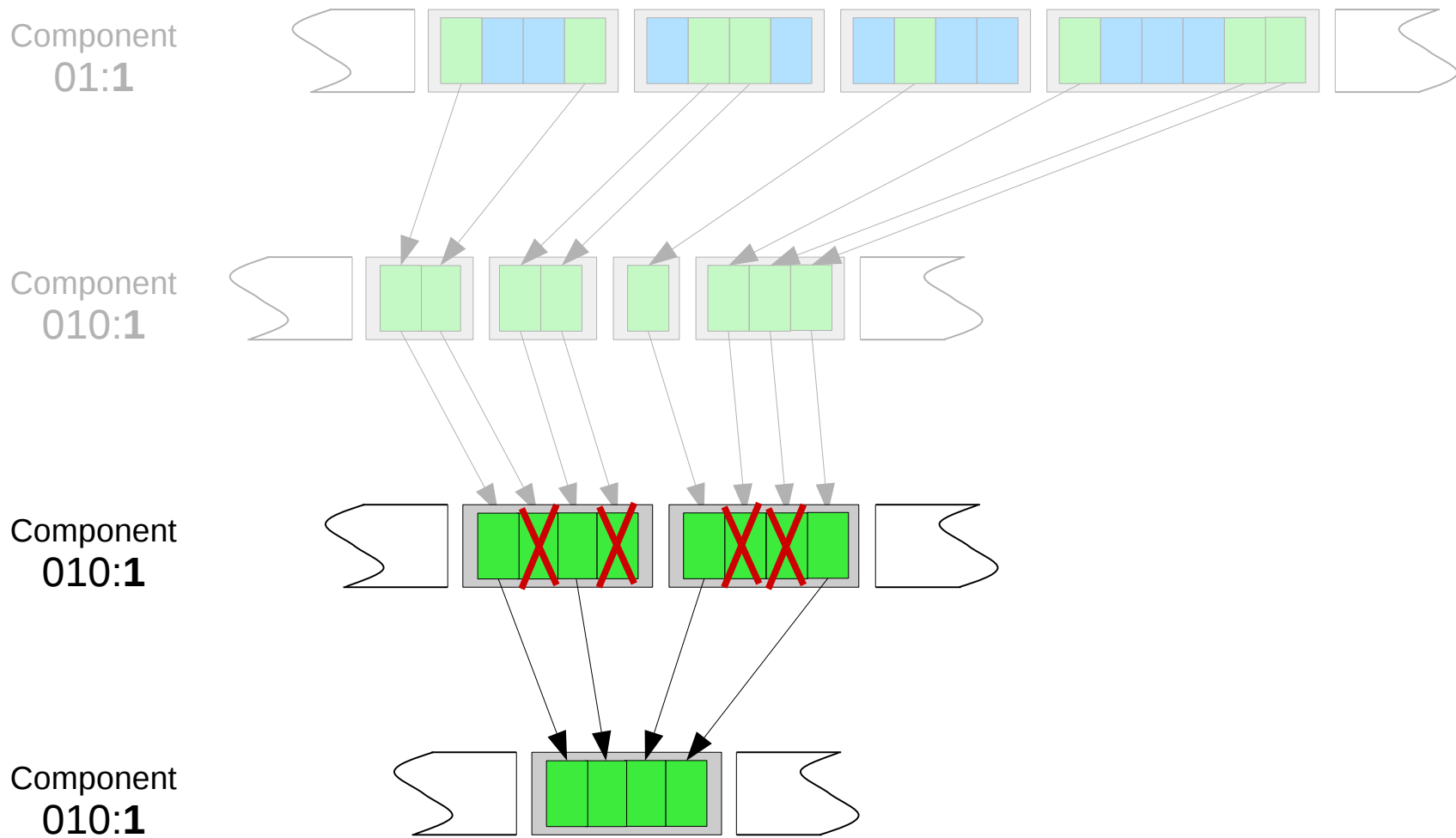
Concatenation



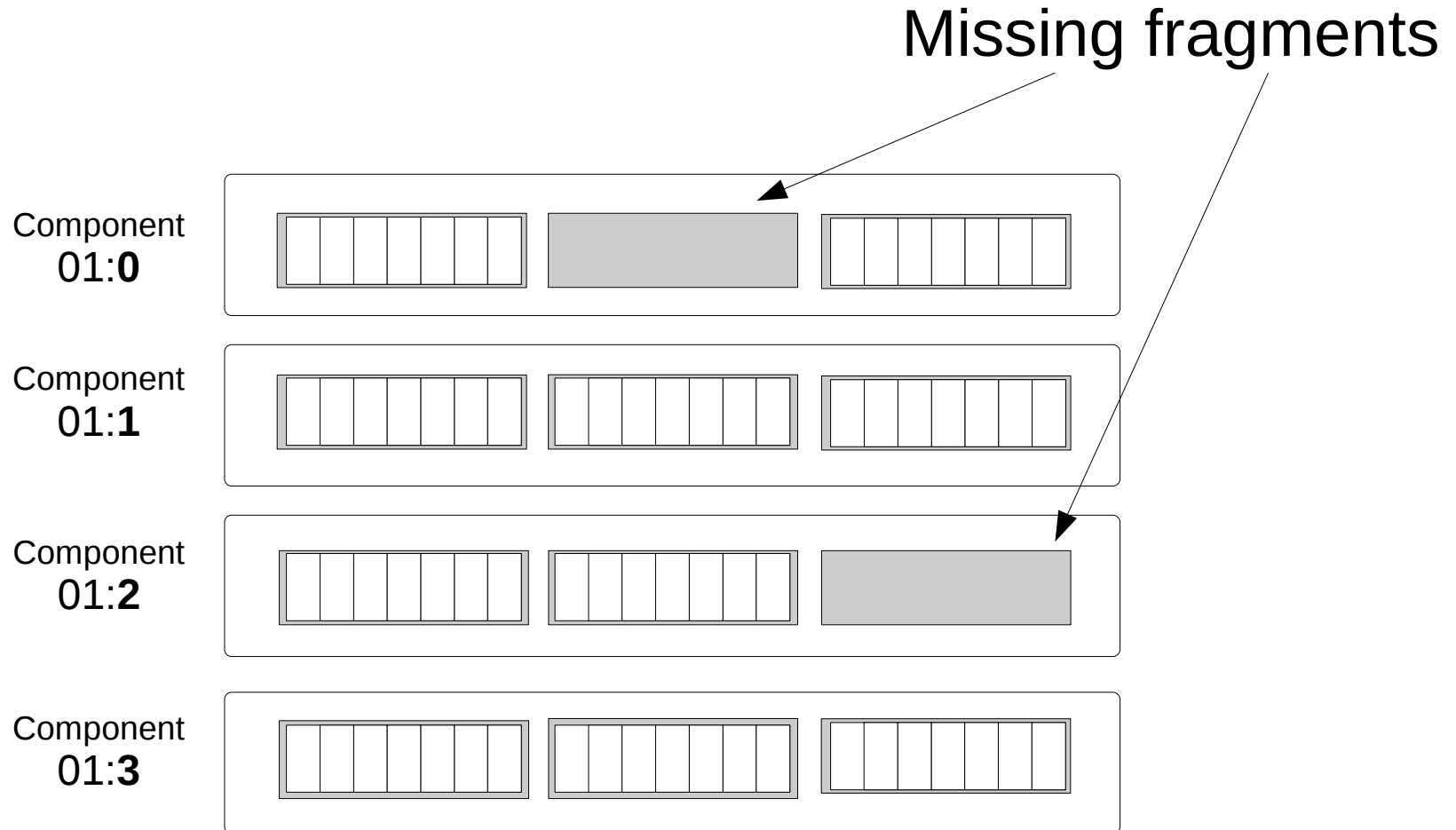
Marking blocks to reclaim



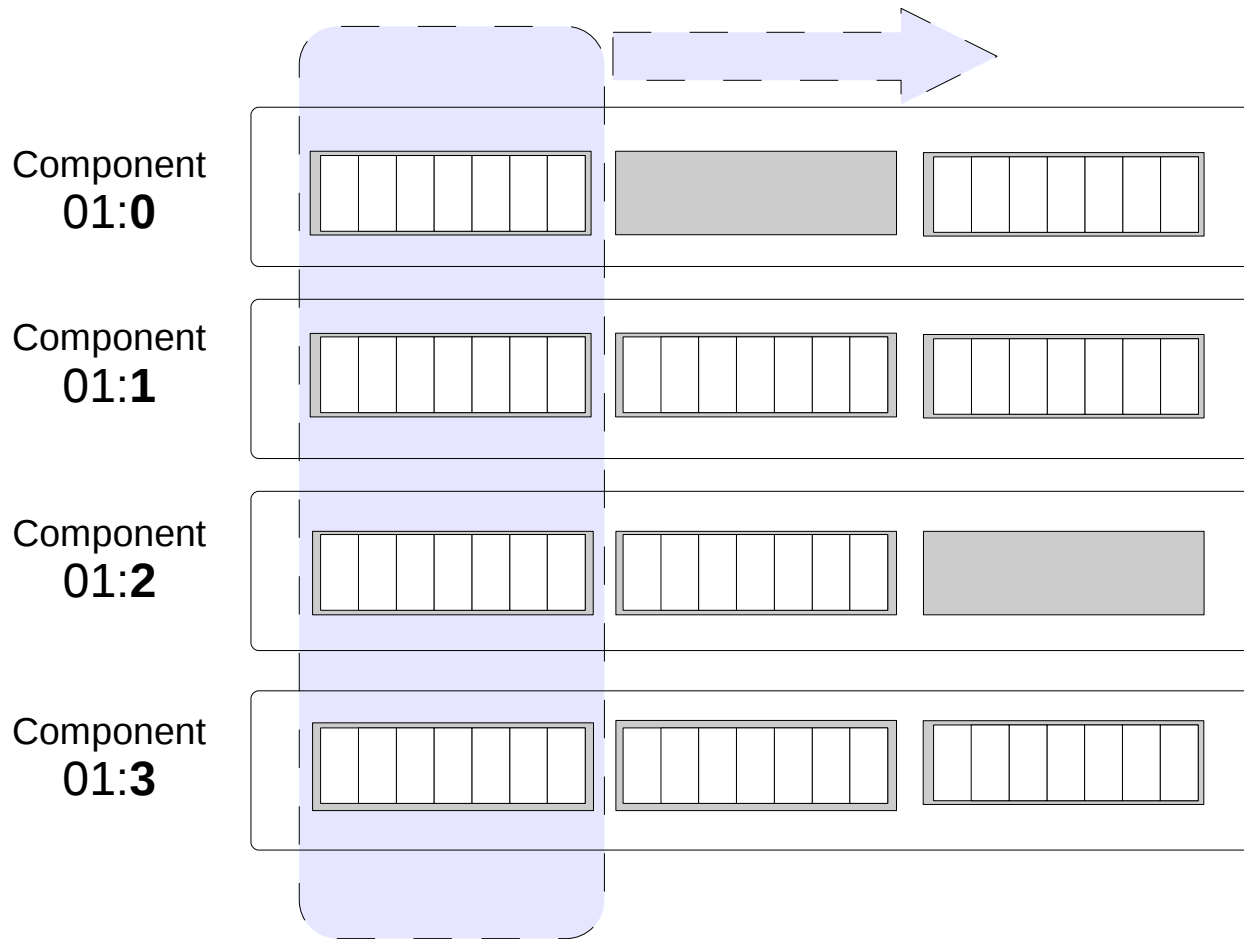
Space reclamation & Concatenation



Data Services: Identification of data resiliency level

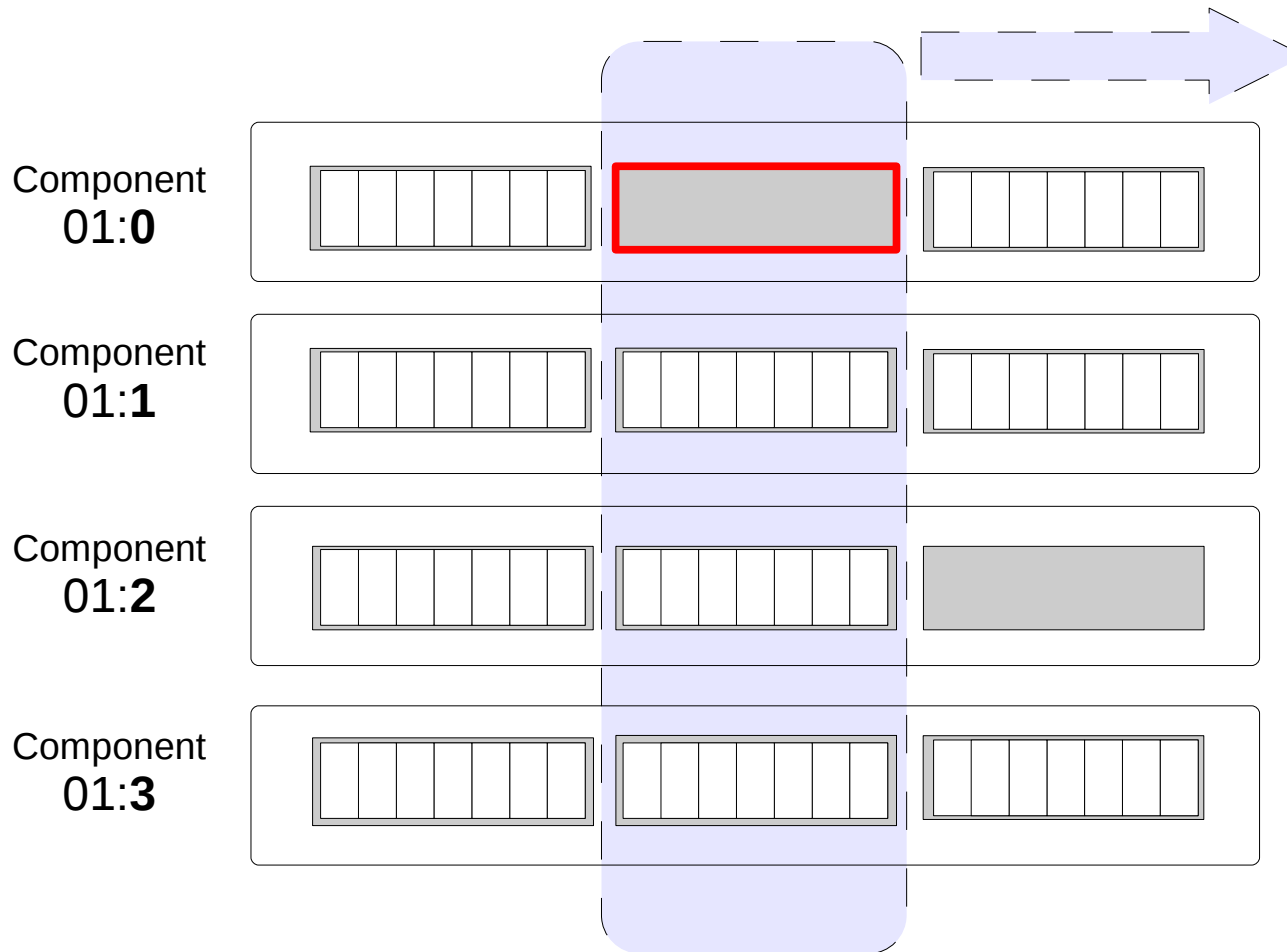


Data Services: Identification of data resiliency level



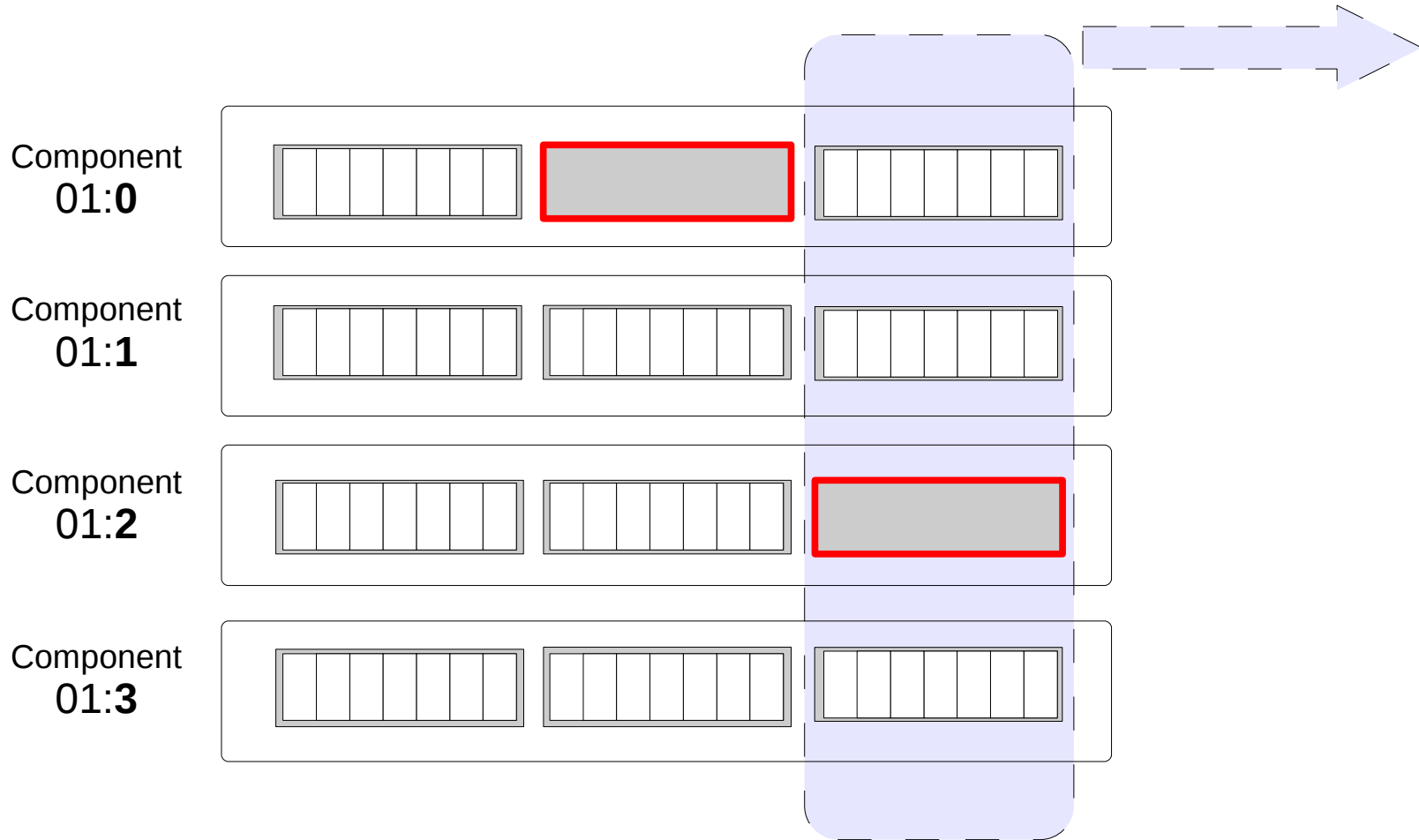
Chain scanning

Data Services: Identification of data resiliency level



Chain scanning

Data Services: Identification of data resiliency level



Chain scanning

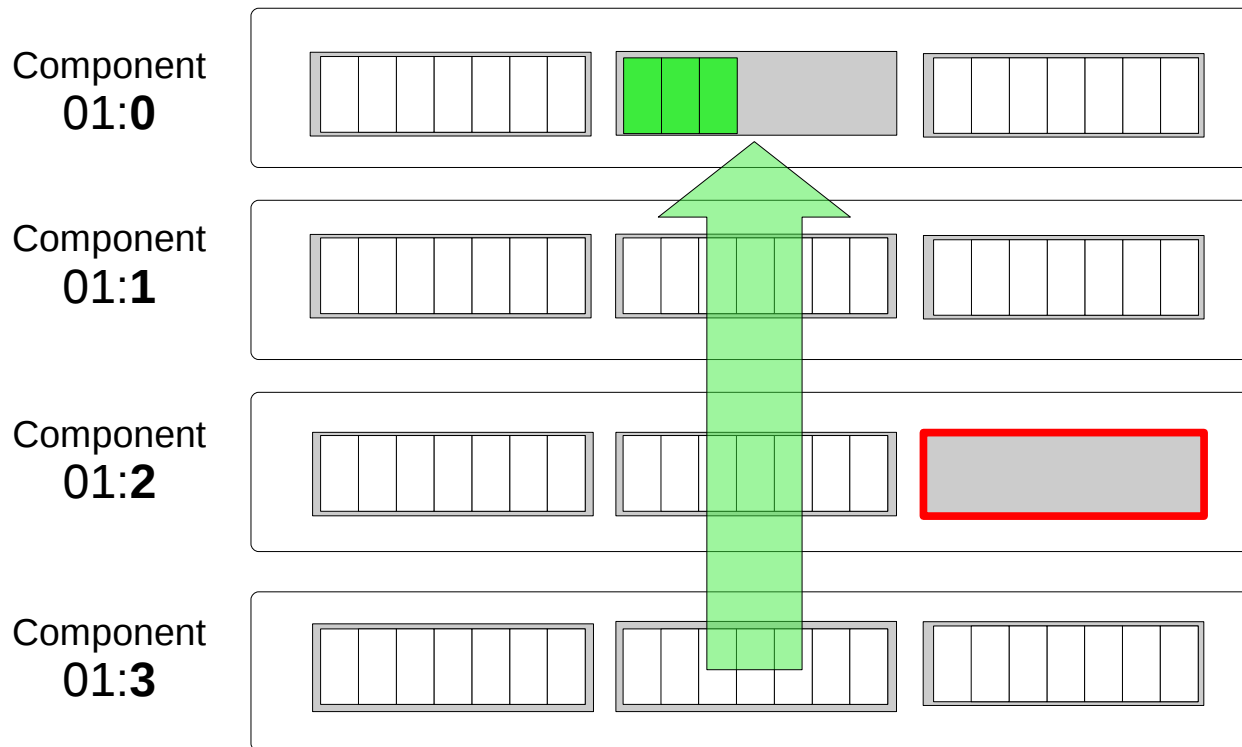
Data Services:

Identification of data resiliency level



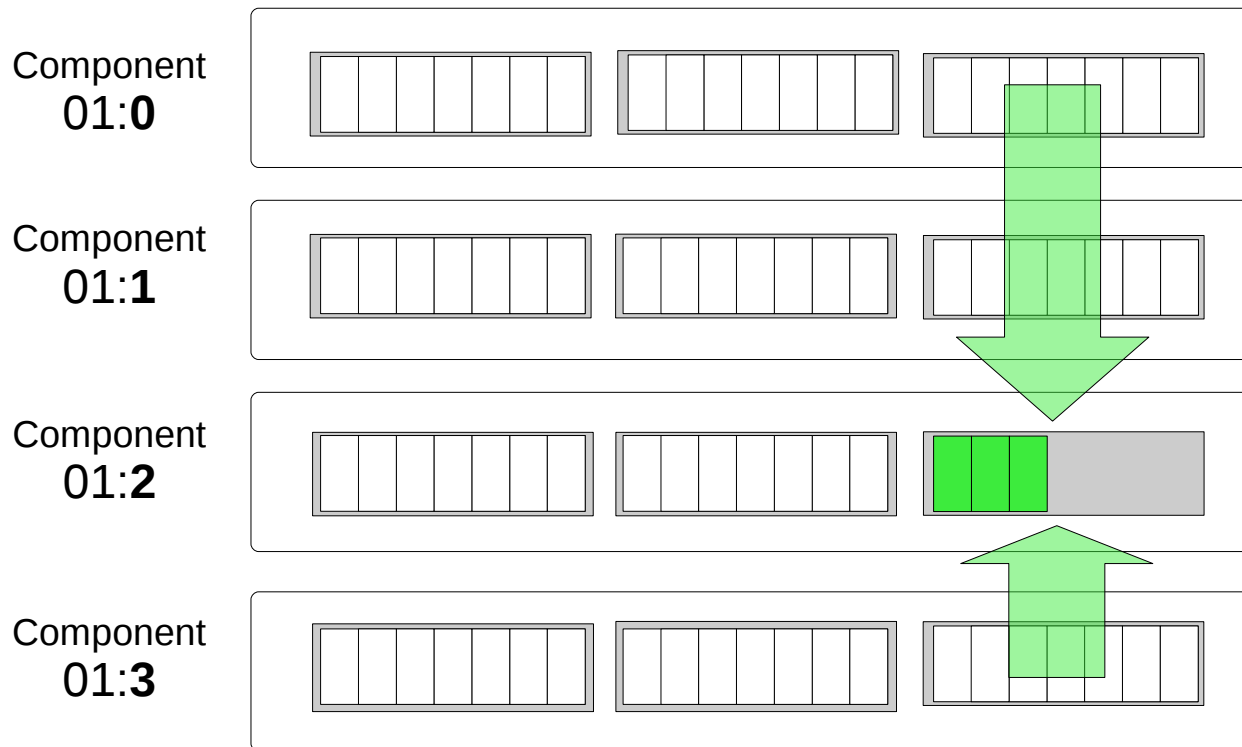
Chain scanning

Data services: reconstruction



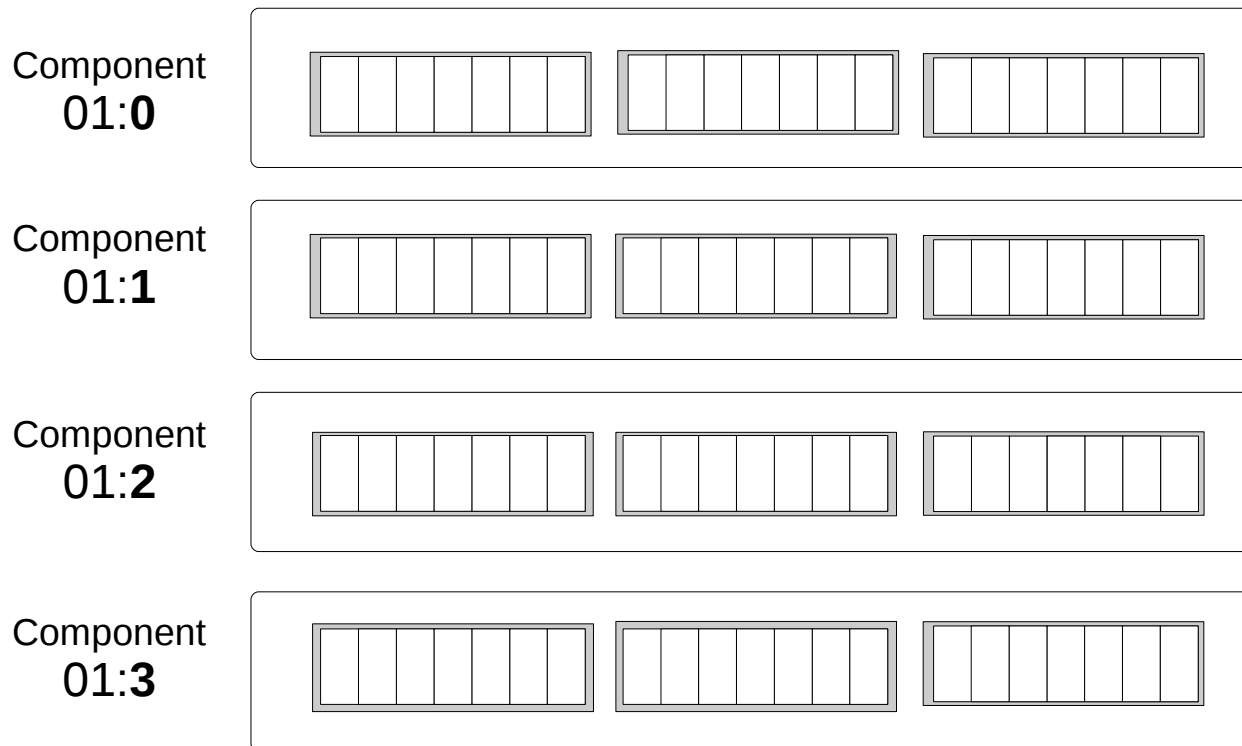
- Sequential read/write of entire Containers
- Erasure decoding and re-encoding

Data services: reconstruction



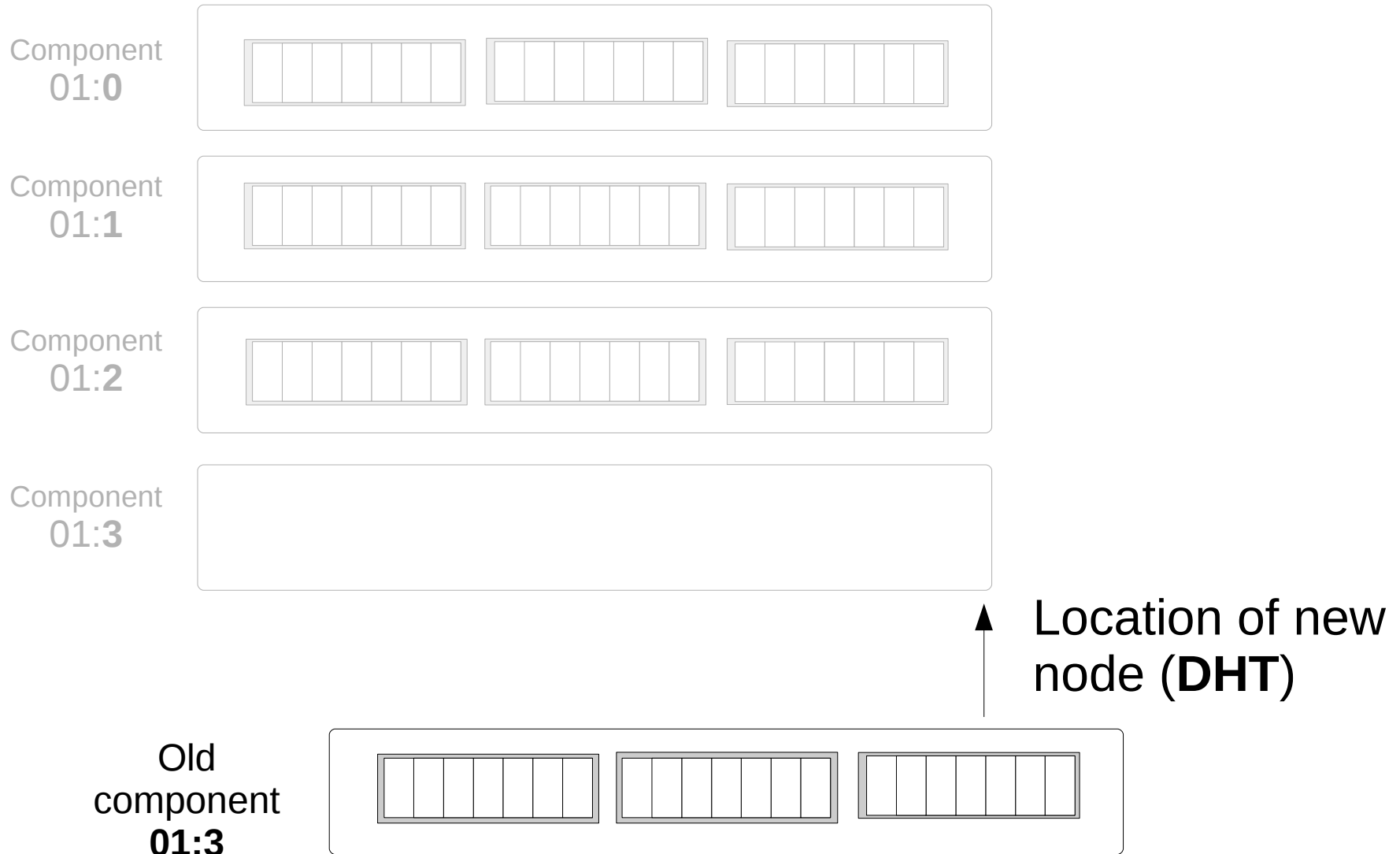
- Sequential read/write of entire Containers
- Erasure decoding and re-encoding

Data services: reconstruction



- Sequential read/write of entire Containers
- Erasure decoding and re-encoding

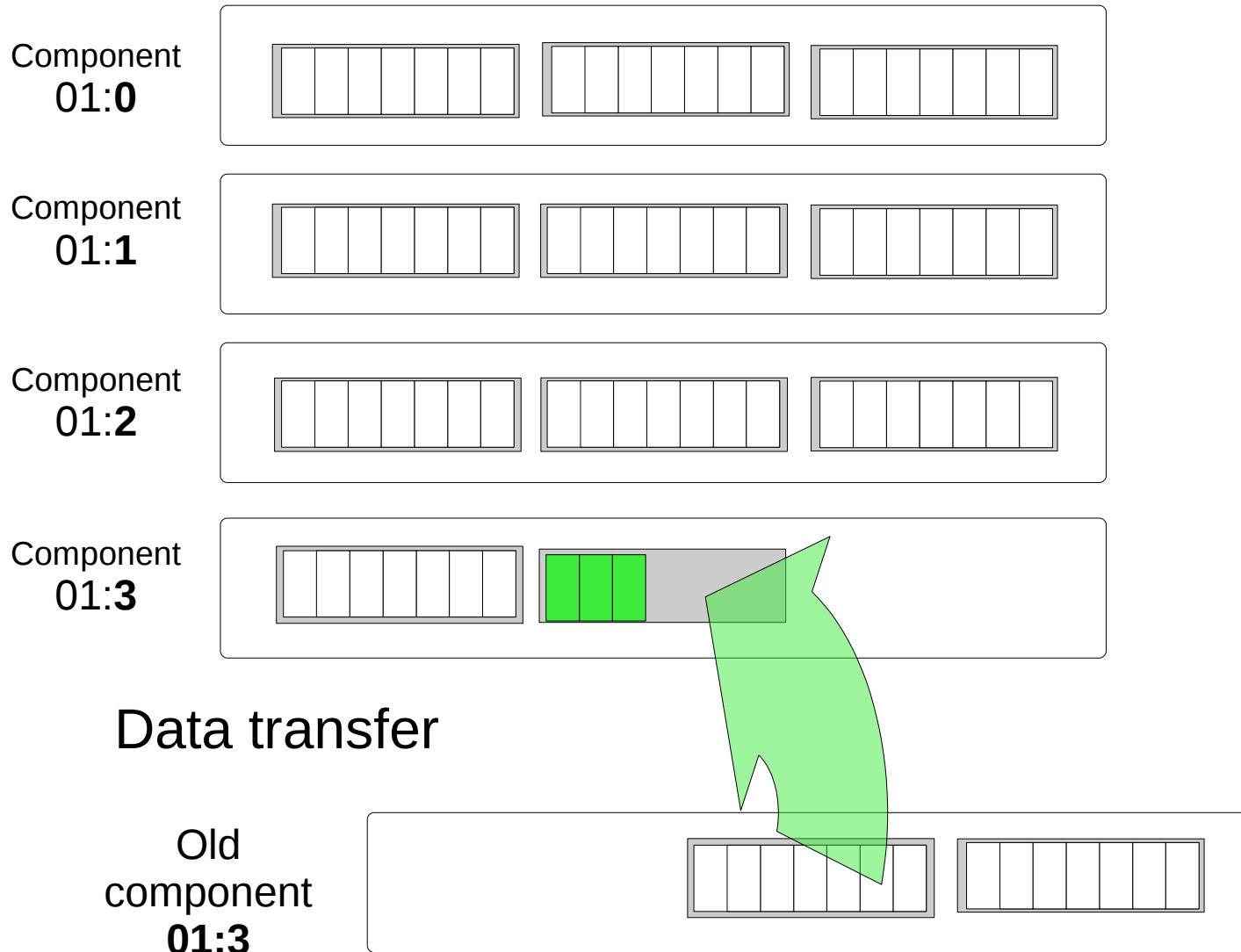
Data services: fast data transfer



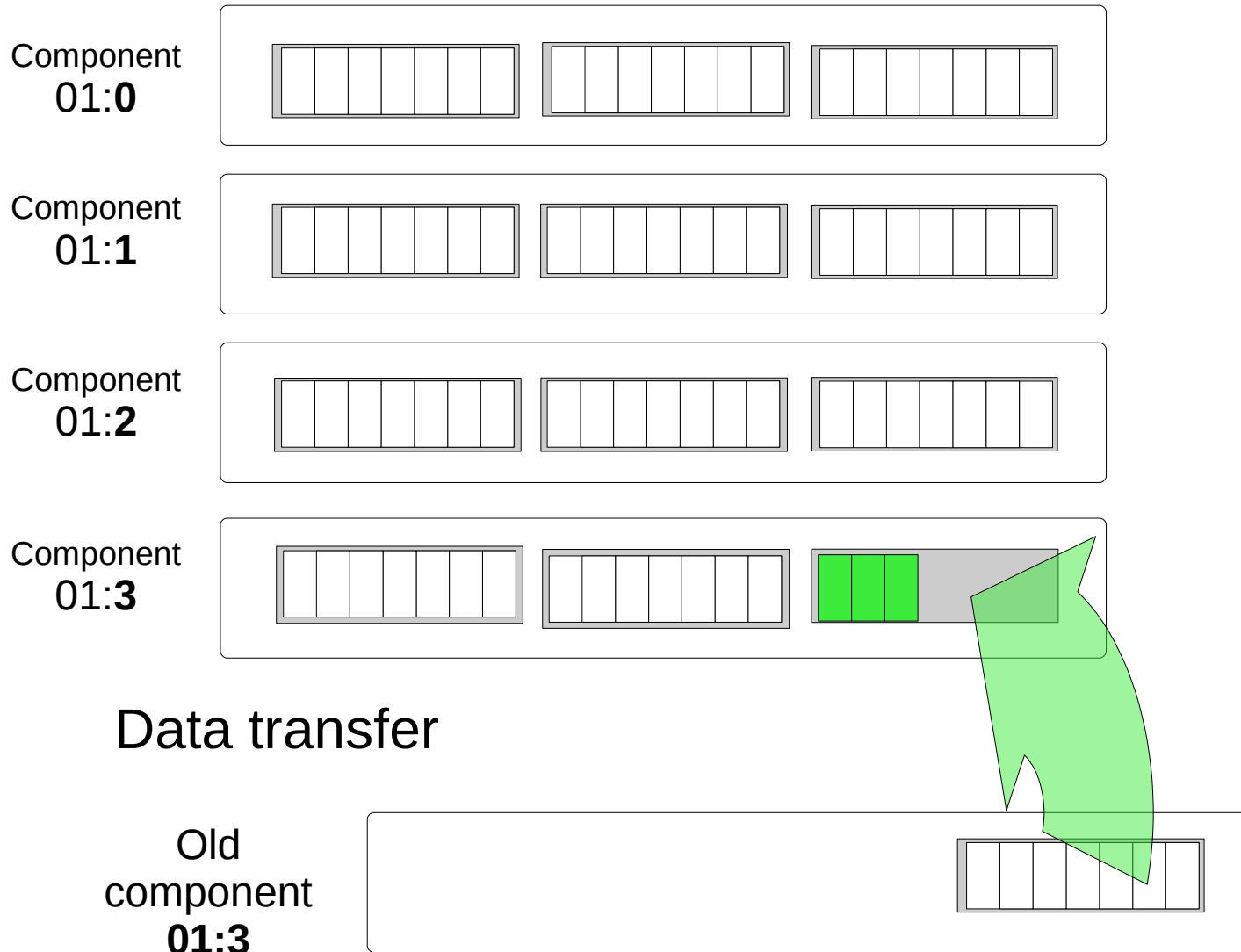
Data services: fast data transfer



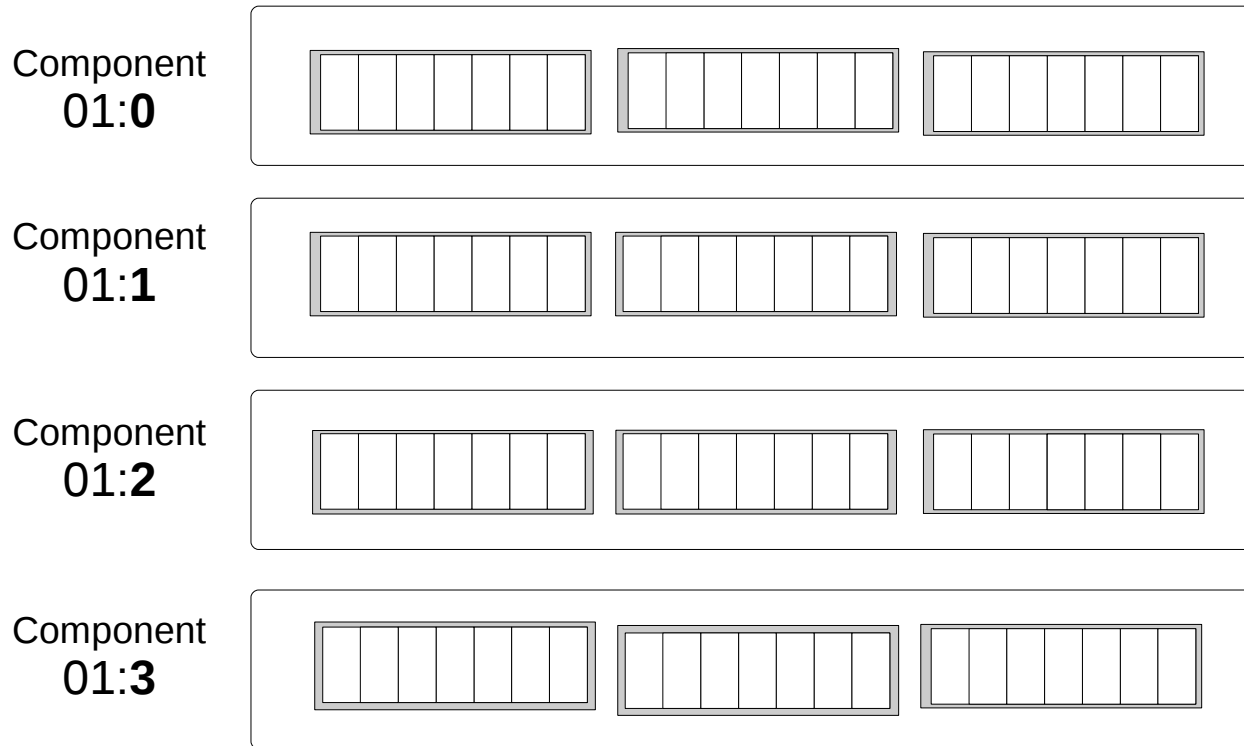
Data services: fast data transfer



Data services: fast data transfer



Data services: fast data transfer



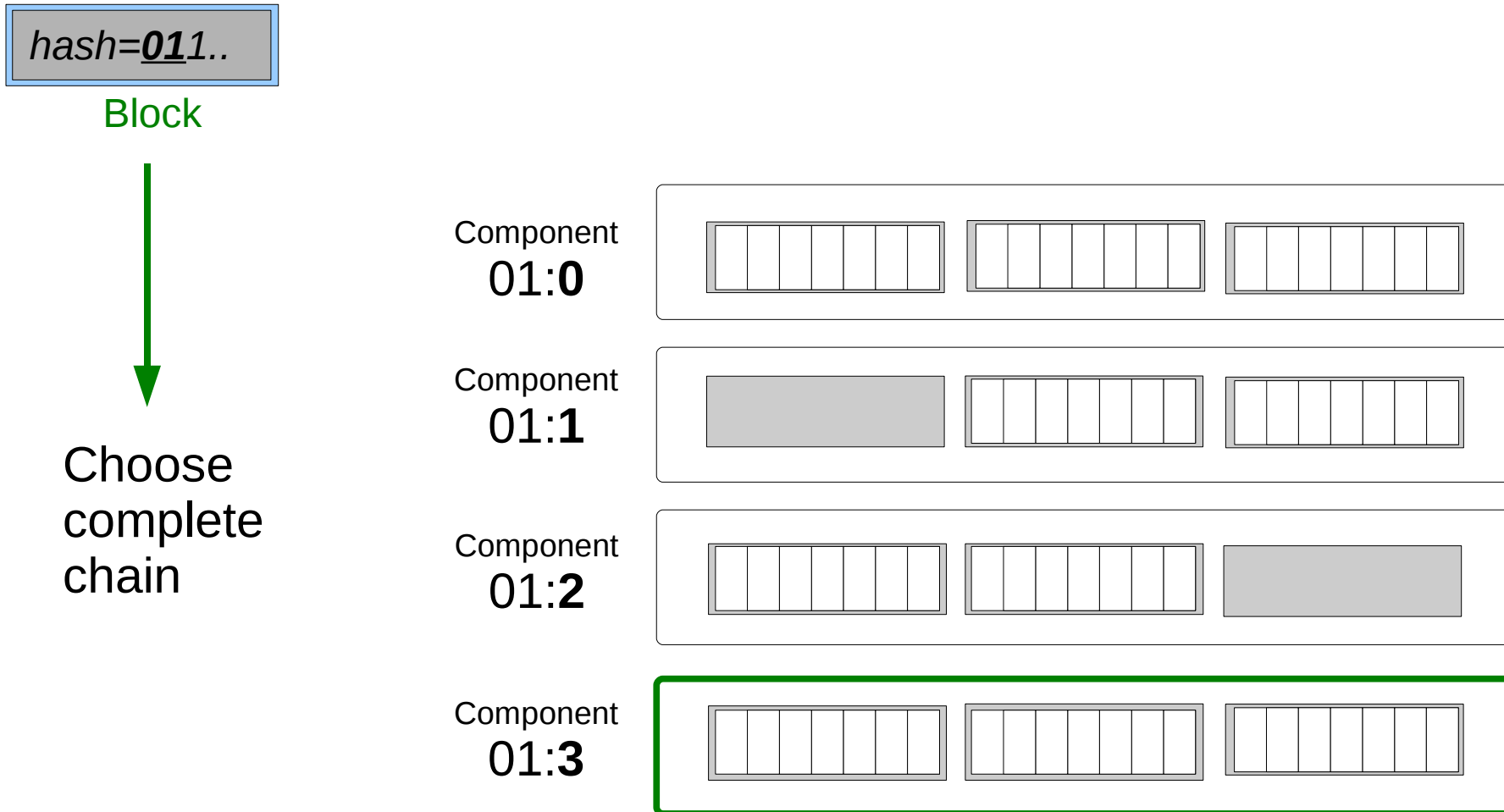
Old
component
01:3



Data services for deduplication

- Global: duplicates detected in entire system
- DHT routing based on content
- Inline deduplication: has to be high-performance
 - Prefetching Containers for streams of duplicates
 - Block hashes stored separately

Data services for deduplication



Completeness: “definitely not a duplicate”

Deletion interaction: wasn't the block scheduled for deletion?

Data services for deduplication

hash=011..

Block

Query

Component
01:0

Component
01:1

Component
01:2

Component
01:3

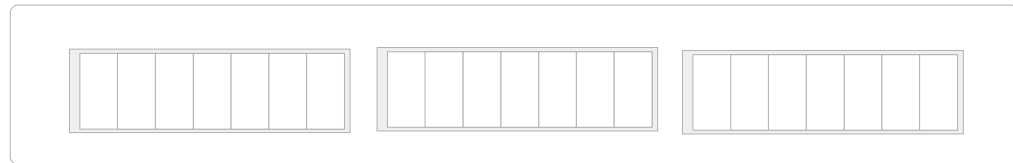


Data services for deduplication

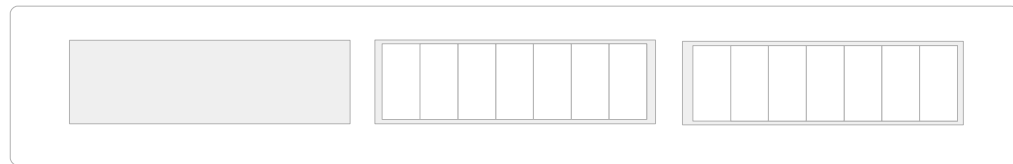
hash=011..

Block

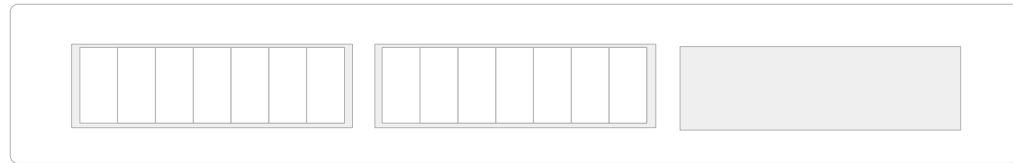
Component
01:0



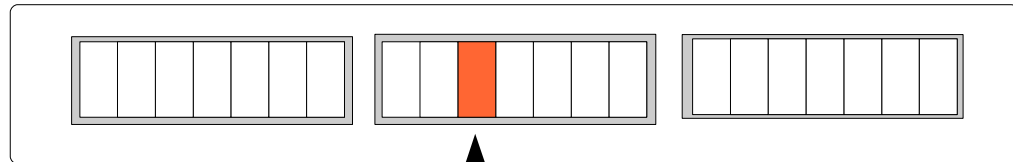
Component
01:1



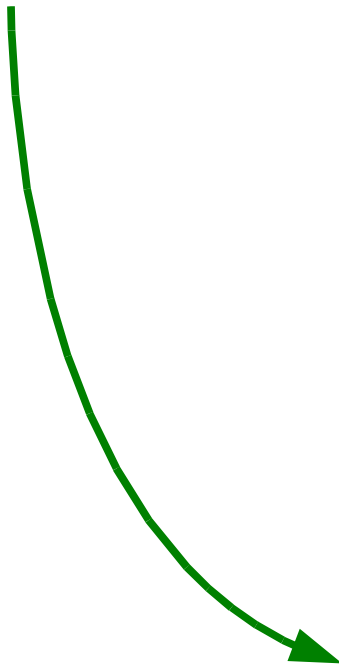
Component
01:2



Component
01:3



Local candidate found

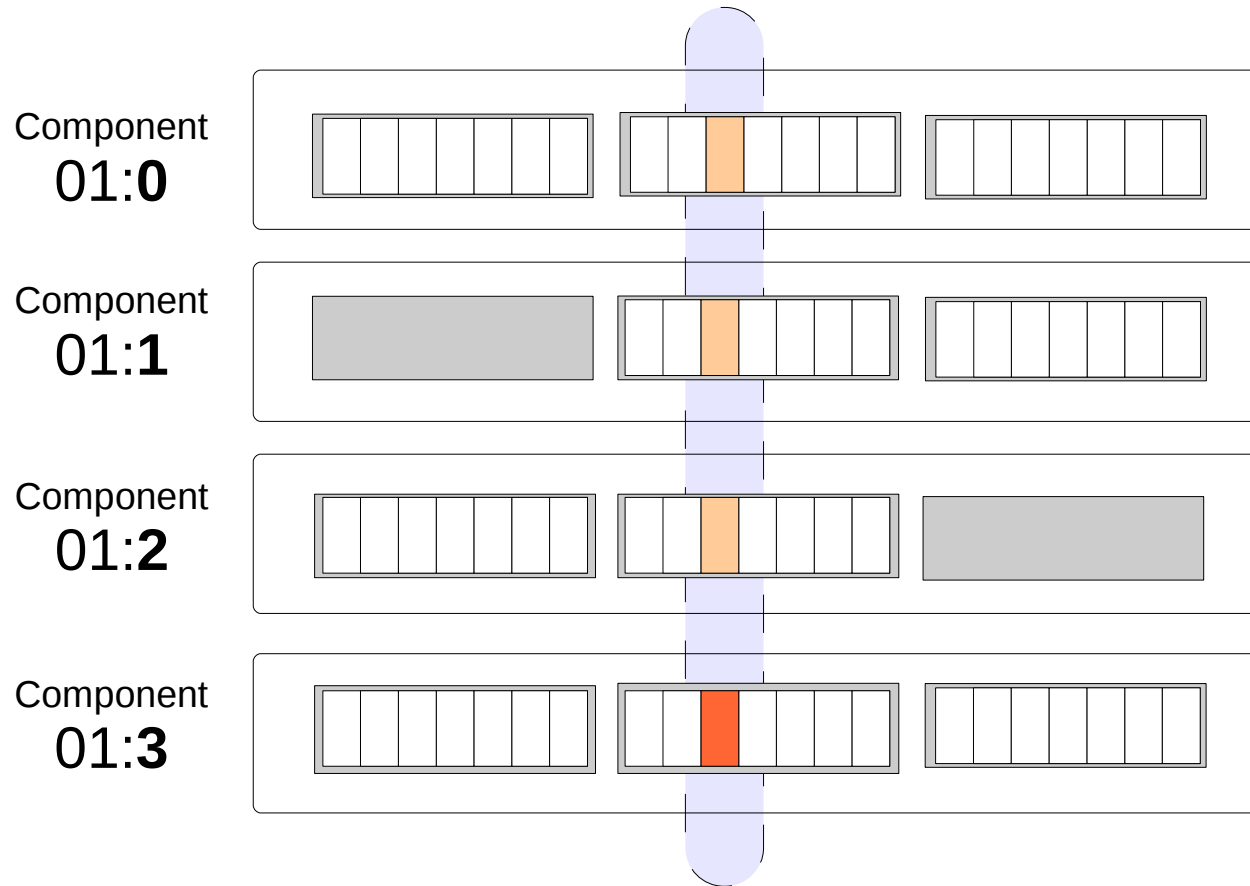


Data services for deduplication

hash=011..

Block

Successful dedup

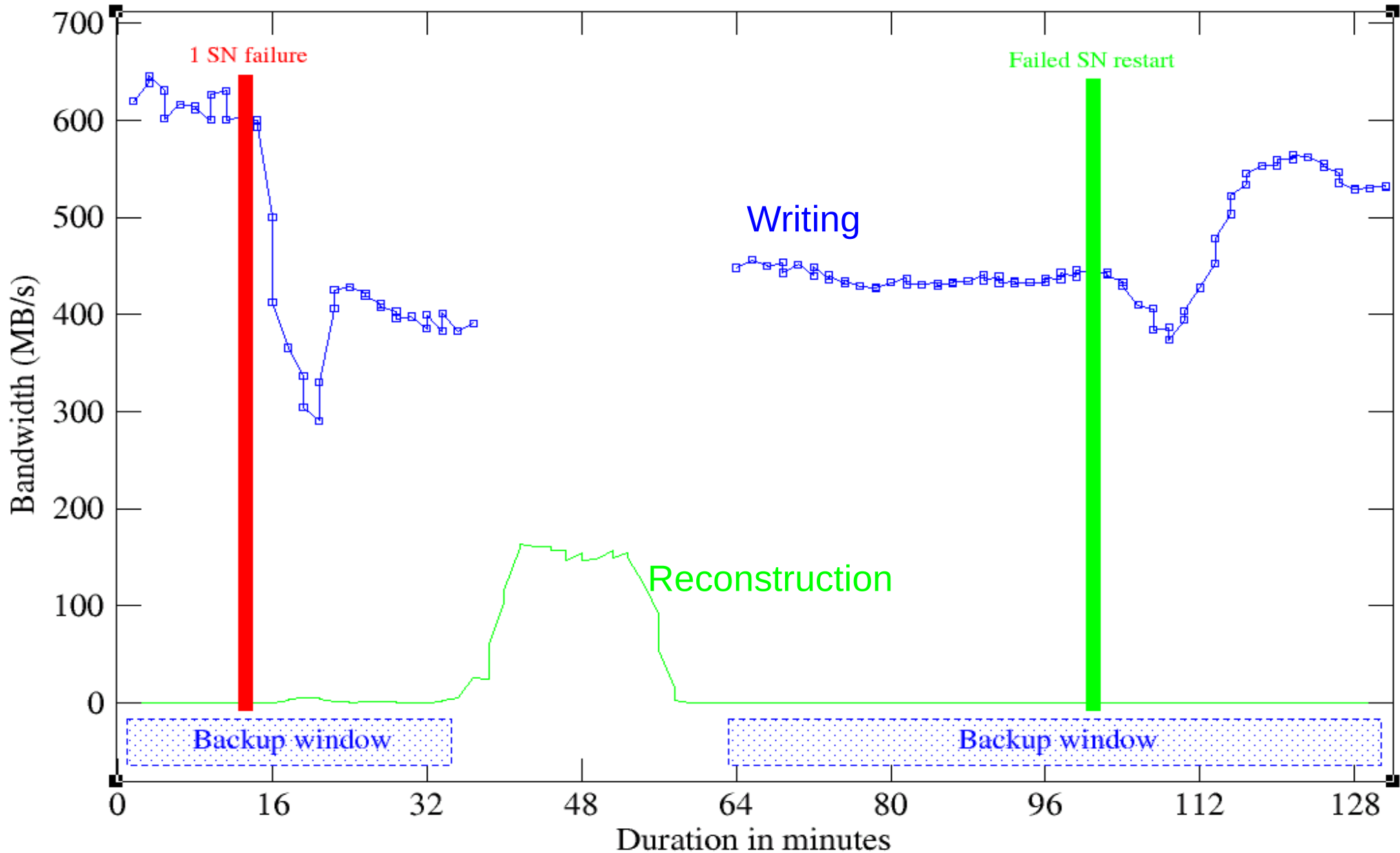


Candidate verification

On-demand data deletion

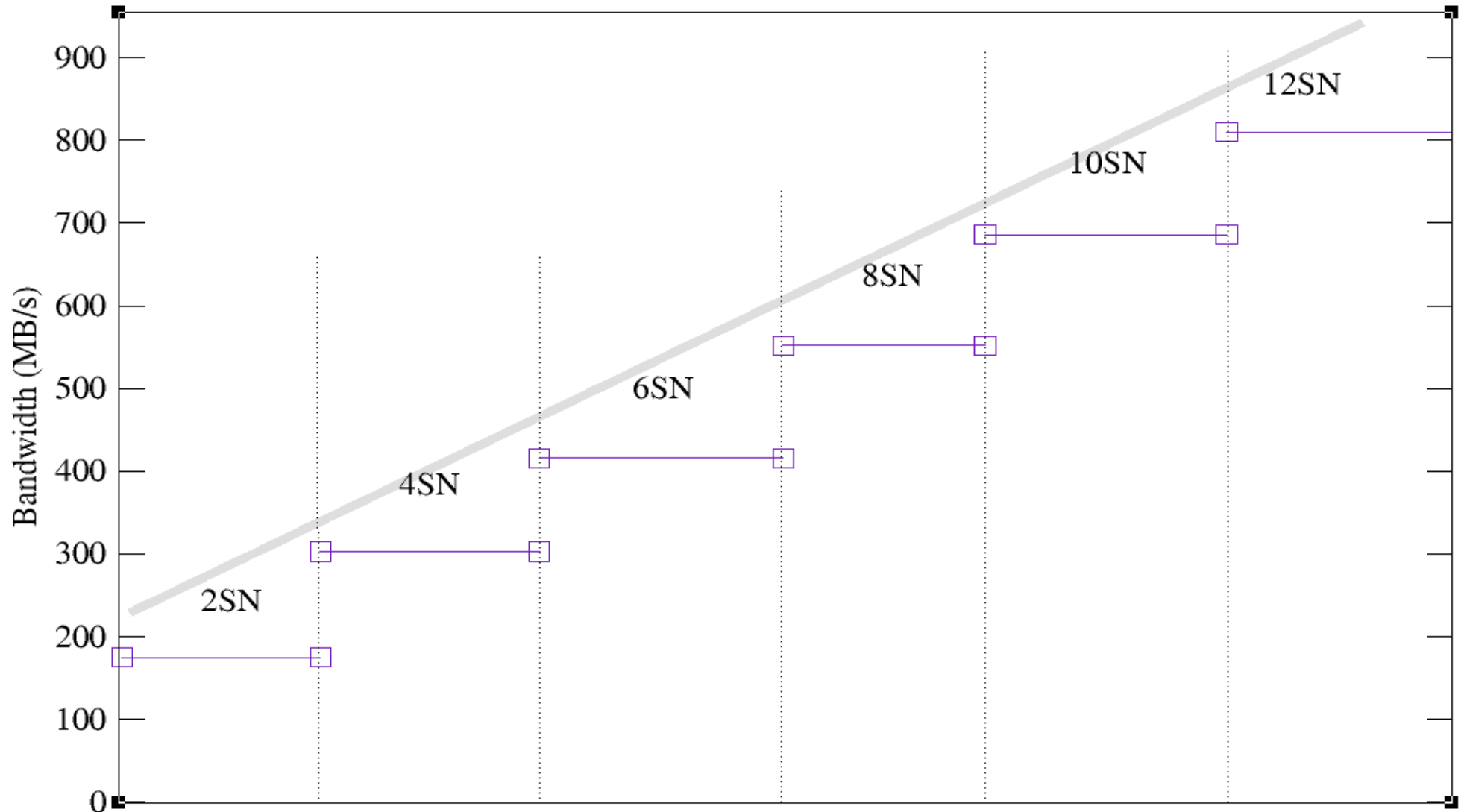
- Distributed garbage collection
- Per-block reference counter stored per-fragment
- Failure-tolerant
 - Block reference counter calculated independently on peer Container chains
- Interference with duplicate elimination:
 - read-only phase for block tree traversal
 - space reclamation in background

Writes during node failure



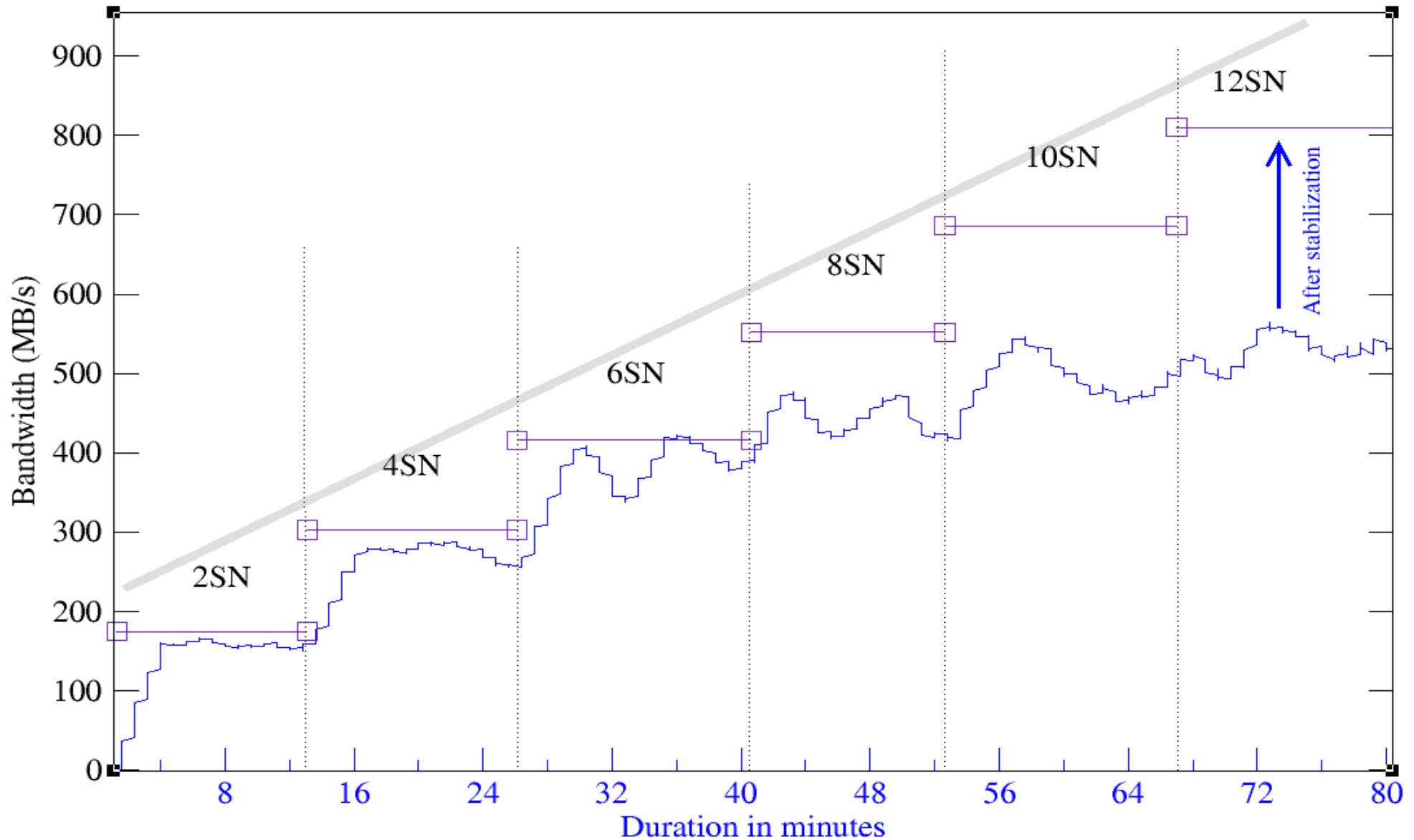
Write Scaling

nodes added while writing



Write Scaling

nodes added while writing



Questions?