JG|U

JOHANNES GUTENBERG
UNIVERSITÄT MAINZ

Design of an Exact Data Deduplication Cluster

Jürgen Kaiser (JGU)

Dirk Meister (JGU)

Andre Brinkmann (JGU)

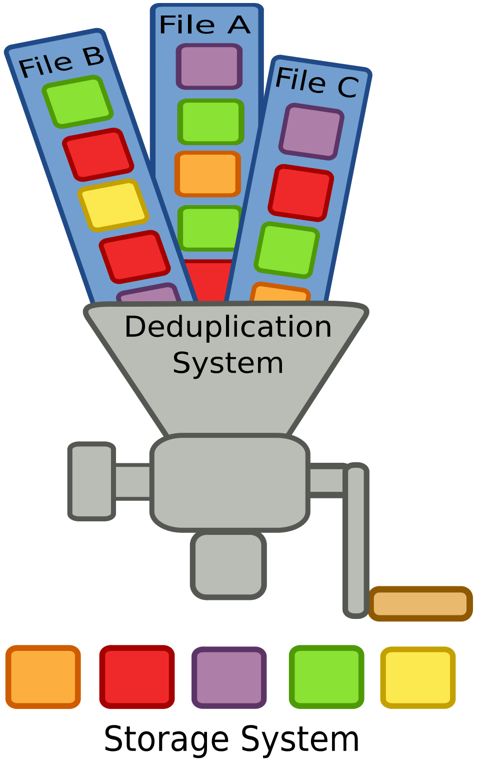
Sascha Effert (christmann)

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

JG|U

Outline

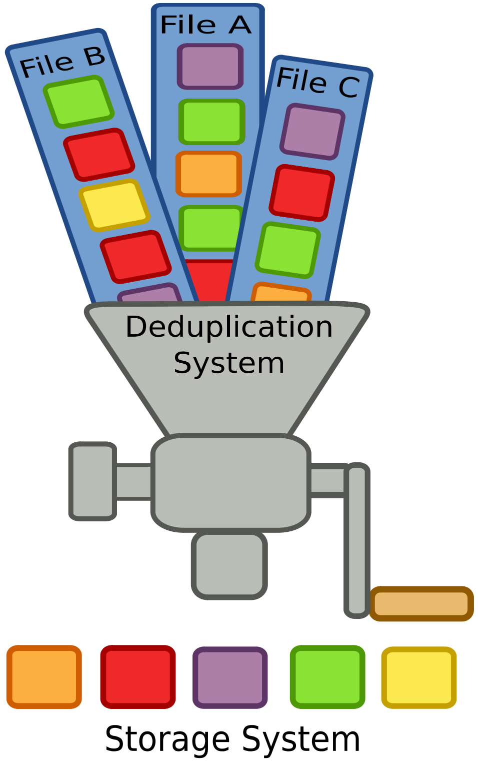
- Deduplication (short)
- System Overview
- Fault Tolerance
- Inter-node Communication
- Evaluation
- Conclusion



Deduplication (short)

- Storage savings approaches
- Remove course-grained redundancy

- Process overview
  1. Split data into chunks
  2. Fingerprint chunks with cryptographic hash
  3. Check if fingerprint is already stored in index (Chunk Index)
  4. Store new chunk data (Storage)
  5. Store block-chunk-mapping (Block Index)



Clustered Deduplication

- Single-node deduplication systems
  - Limited scaling

- Just a Bunch of Deduplication Systems
  - Lots of independent deduplication systems
  - Complex load balancing, migration, and management
  - Cross data set sharing

- → Clustering promising to scale deduplication

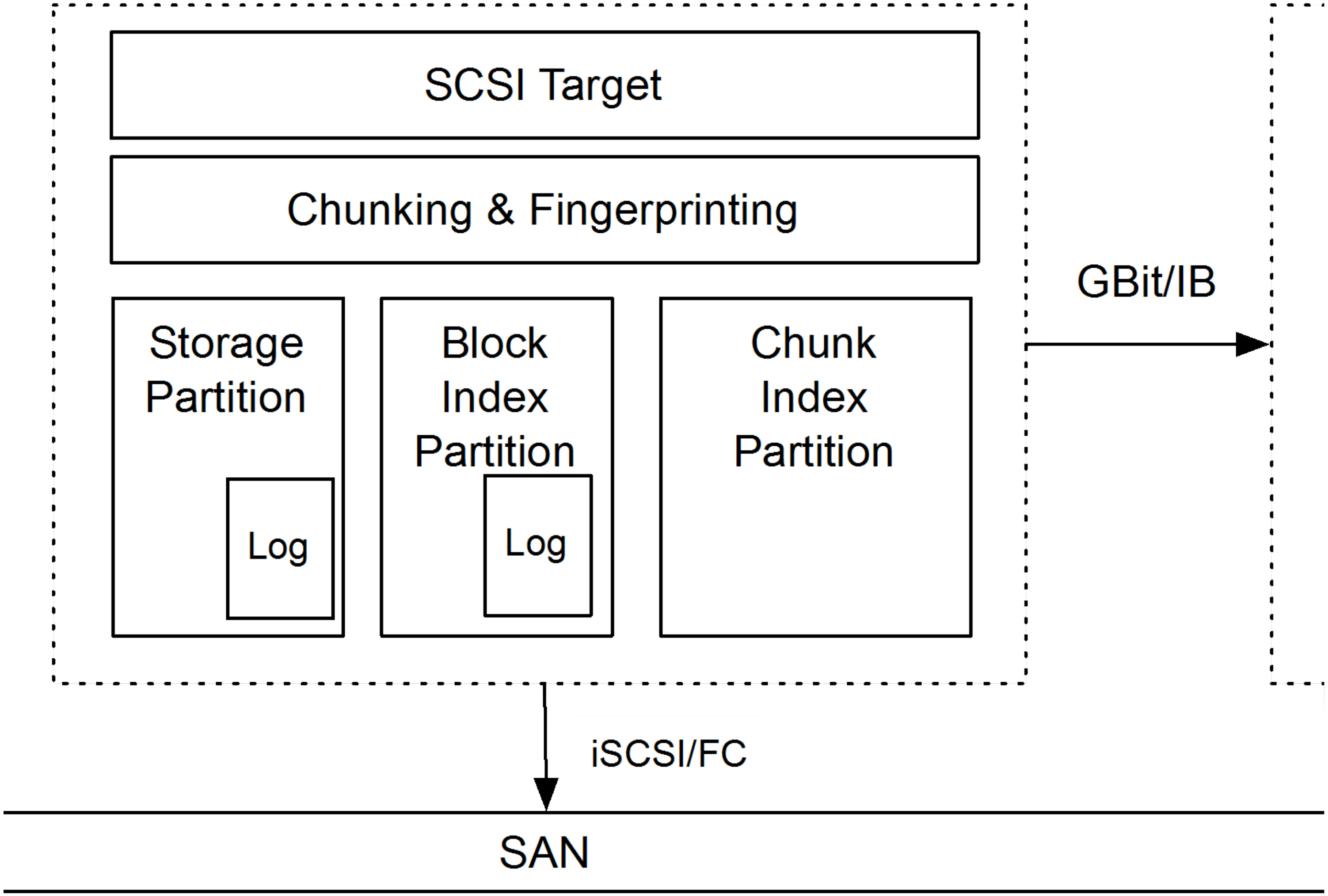
Exact deduplication

- Term: “Exact deduplication”
  - Detect all duplicate chunks
  - Same deduplication as single-node system
    - But larger and faster

- Goal:
  - Scalable, exact deduplication
  - Small chunk sizes (8 – 16 KB)

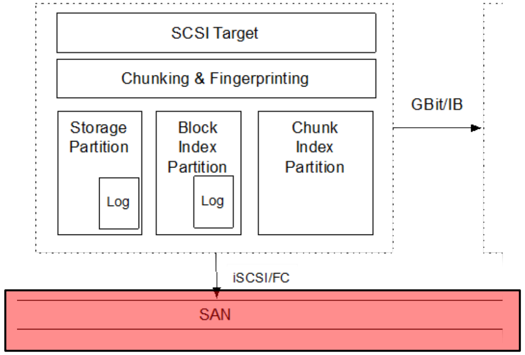
- Contribution
  - Architecture combining these properties
  - Evaluation using prototype
  - Exploring limitations

System Design Overview



Storage Organization

- Shared Nothing (Direct Attached)
  - – Replication/erasure coding
  - – Chunk data over network

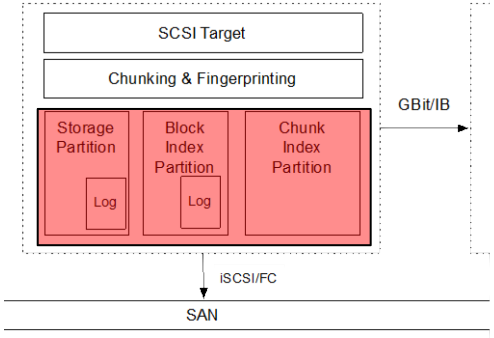


- Shared Storage (SAN)
  - – Complex locking schemes
  - – Scaling issues

- One partition is only accessed by single node
  - – No short-term locking
  - – Sharing is used for
    - Fault tolerance
    - Load balancing

Partition types

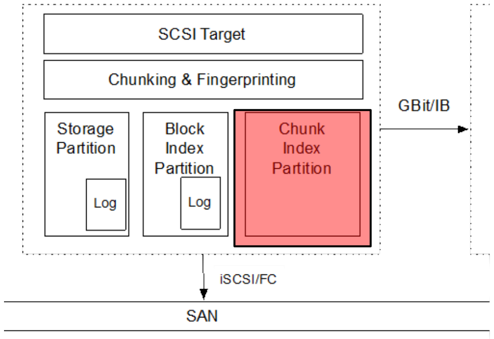
- Chunk Index Partitions
- Block Index Partitions
- Container Partitions
- Container Metadata Partitions



- Partition types are handled differently
  - Fault tolerance
  - Load balancing
  - Data assignment

Chunk Index Partition

- Contains parts of distributed chunk index
- Chunk assigned by SHA prefix



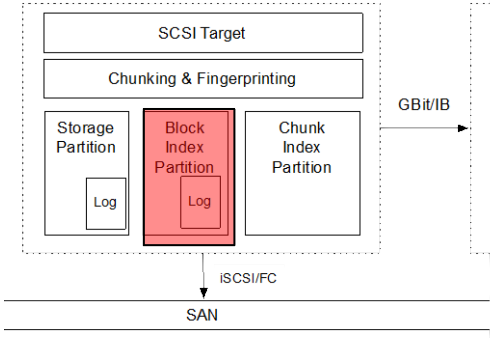
- Performance sensitive
  - → SSD Storage
  - 100,000s requests/s during writes
  - Multiple request issues concurrently
  - Index not updated directly
    - Write-ahead log
    - Dirty uncommitted state in memory

Block Index Partition

- Contains the block index

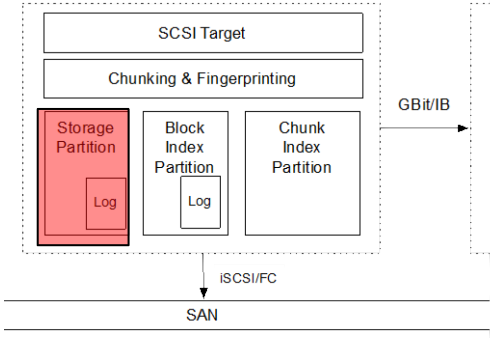
- iSCSI target is assigned to partition
- Only node the partition is assigned to exports iSCSI target
  - Extension like clustered iSCSI could be implemented

- High locality of access
  - → Modest performance requirements
  - Currently on SSD storage



Container Partition

- Stored on HDD storage
- Stored parts of container storage
- Chunk data stored in container



- Assignment
  - Prefer local partitions
  - One local partition for writing
  - → Chunk data never transferred over network

- Container Metadata Partition
  - Metadata, write-ahead log stored on SSD

Fault tolerance

- Storage reliability
  - RAID 6 (or double mirroring)

- Node crash (basic idea):
  - Failure detection using e.g. keep-alive messages
  - Partition remapping by cluster leader
    - New leadership election if cluster leader fails
    - Based on ZooKeeper system
  - Recovery strategy depends on partition type

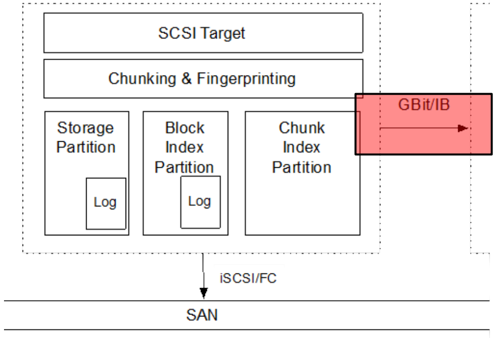
- Load Balancing
  - Strongly related to fault tolerance
  - Details in paper

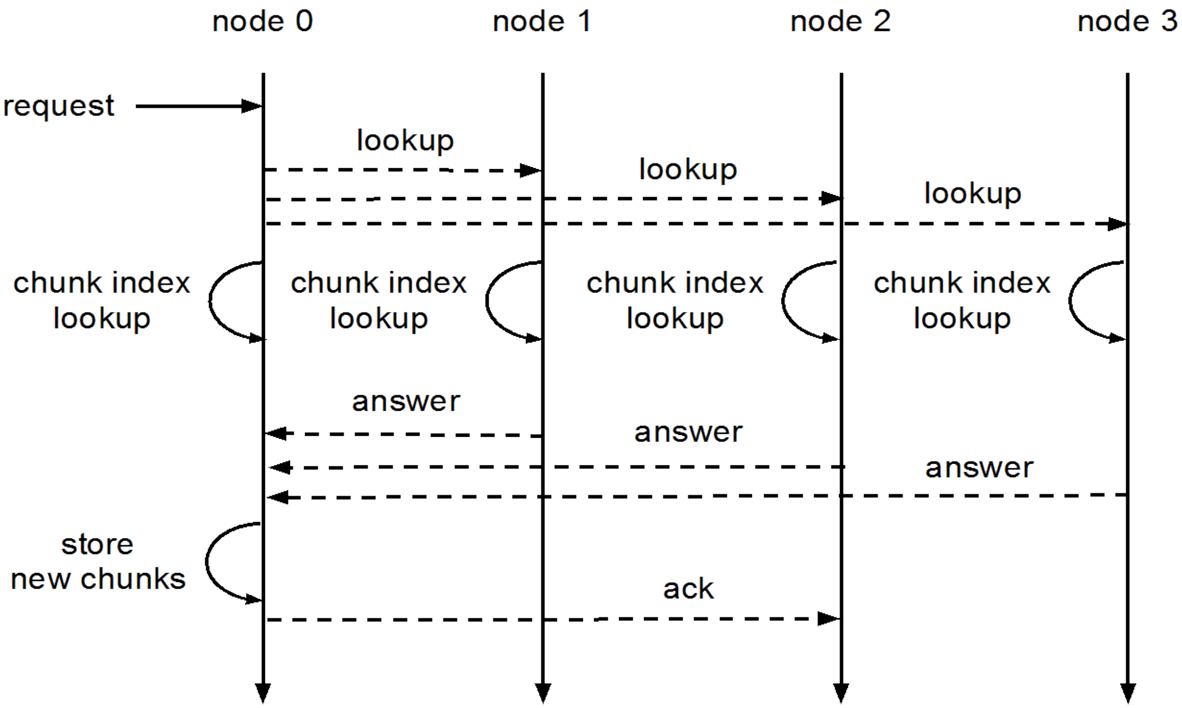
Recovery Strategy

- Chunk Index
  - Needs to be up very quickly → No log replay
  - “False Negatives“
  - State cleaned up during background log replay
- Block Index
  - Out-dated results are not acceptable
  - Replay of write-ahead log to recover latest state
  - Downtime of around a minute
- Container Storage
  - Only small non-committed state → No issue

Inter-node Communication

- Main communication pattern
  - For all chunks in write request:
    - Ask responsible node
    - Requests to same node aggregated
    - Each message is small (20 – 100 bytes)



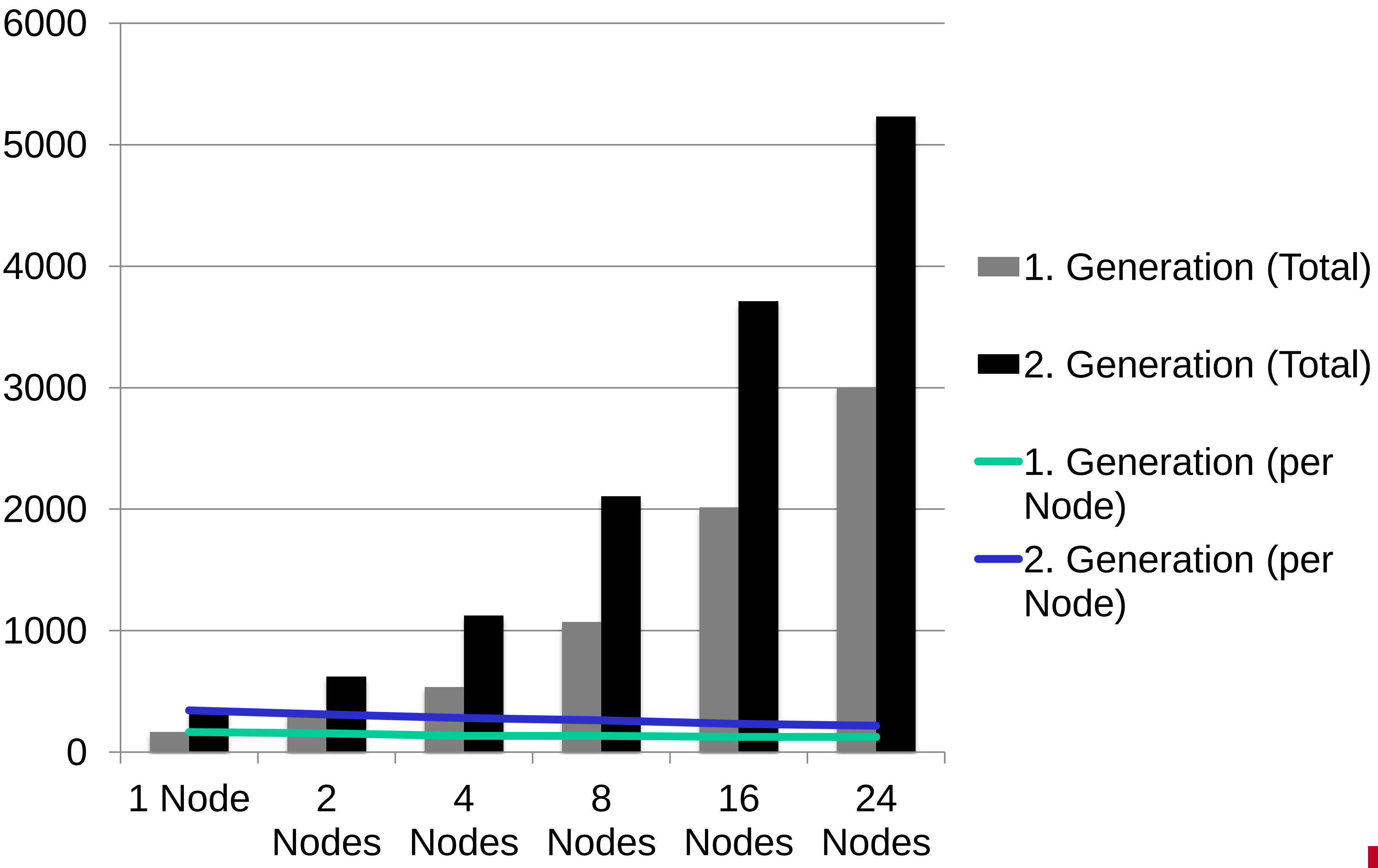


Evaluation

- Prototype implementation
- 60 node cluster
  - 24 as deduplication nodes
  - 24 for workload generation (clients)
  - 12 for shared storage simulation (6 SSD, 6 SAN)
  - Gigabit Ethernet Interconnect
  - IPoIB iSCSI to storage nodes

- Load generation
  - Based on pattern/probability distributions from traces
  - “1. Generation”: Empty deduplication system
  - “2. Generation”: Later backup runs

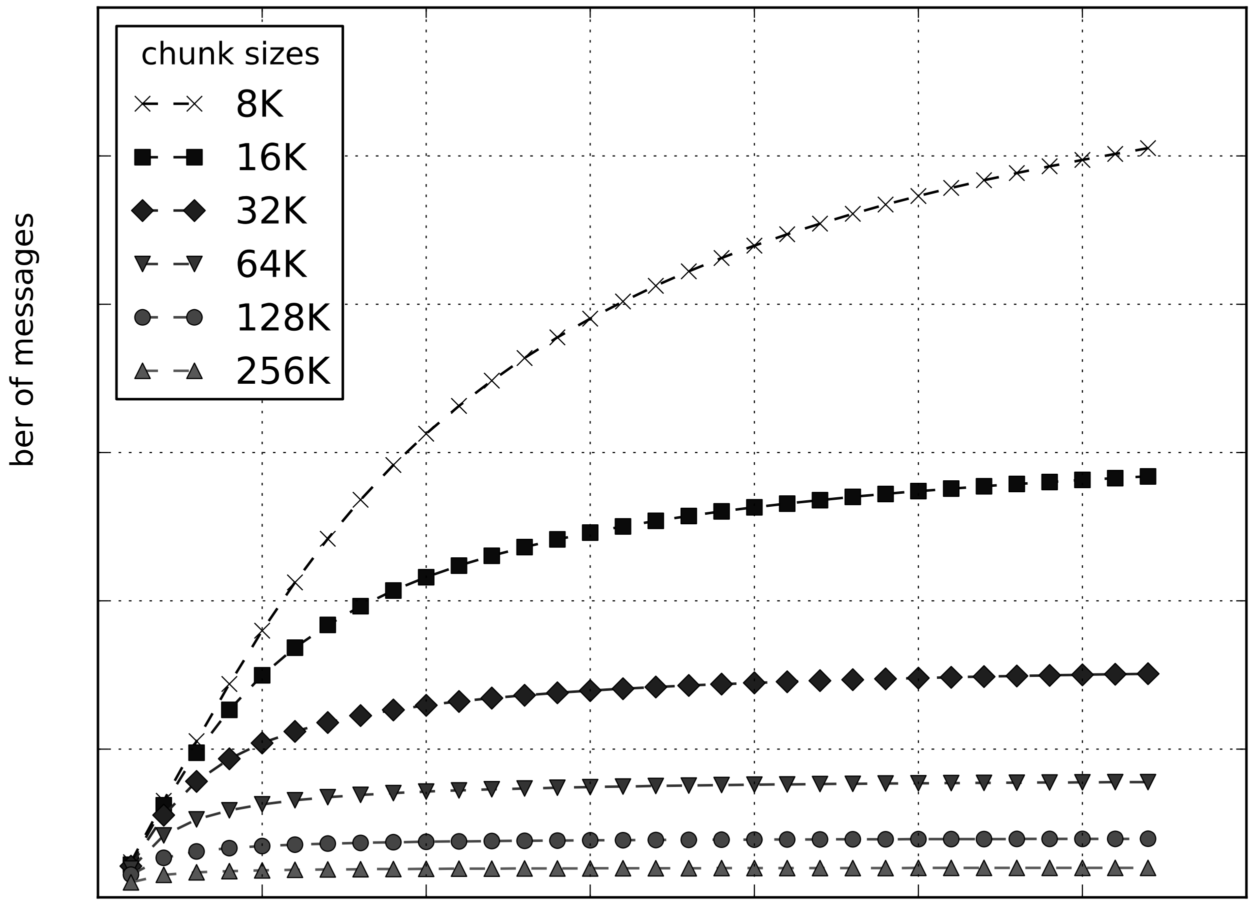
Prototype throughput (in MB/s)



Communication Limit

- Throughput relies on exchanging 100,000s of message/s
- More nodes
  - More write requests
  - Higher chunk lookup “fan-out”
  - Linear more ability to process messages
  - → Sub-linear scaling
- But only for small chunk sizes, small cluster
  - No fan-out for larger cluster
  - Scaling becomes more linear as cluster grows

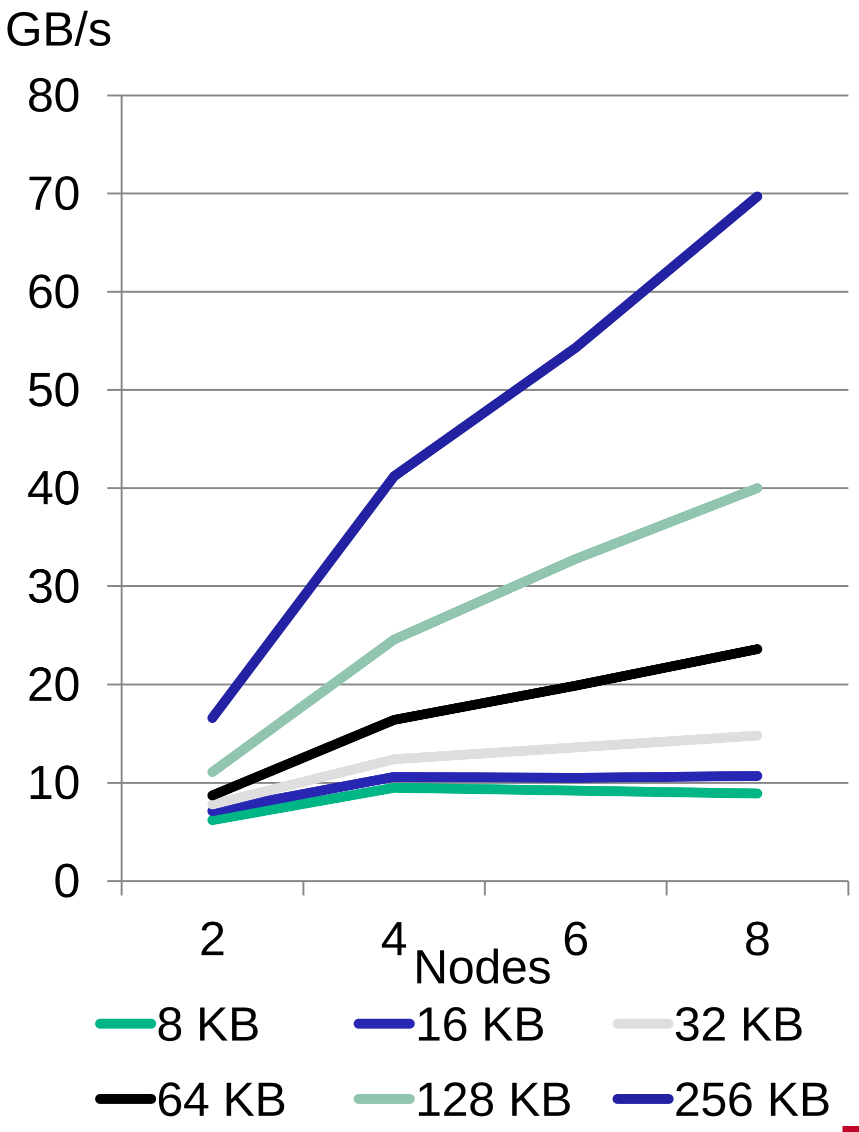
Expected number of messages



Communication Limits: Results

- Measured messages based on communication pattern
- Estimated throughput based message rates

- Performance excluding Storage, Chunking, …
  - Only communication



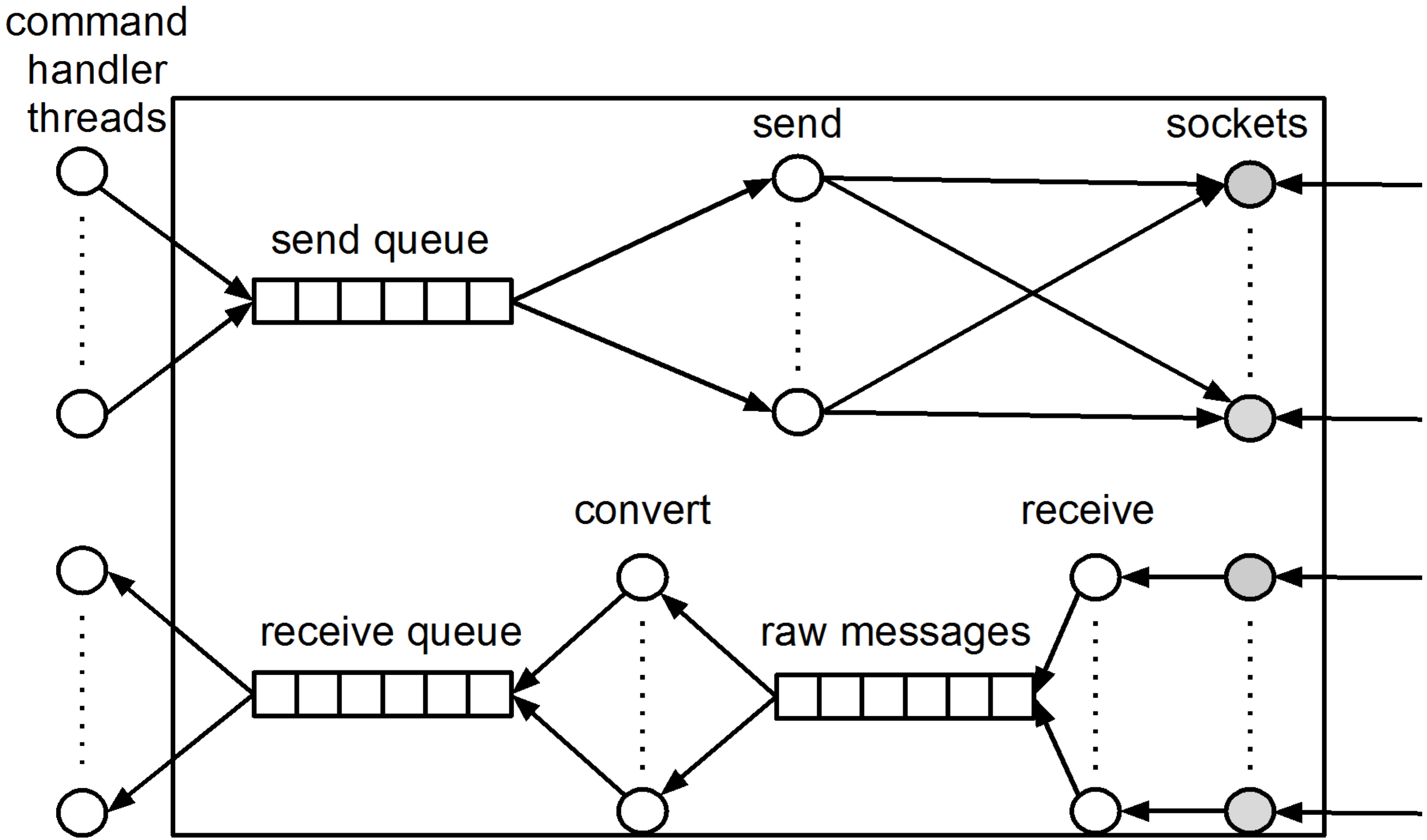
Conclusion

- Exact, inline deduplication cluster
  - Architecture
  - Prototype
- Exact deduplication clusters with small chunk sizes are possible
- However, message exchange limits scalability

Questions?

THANK YOU

Architecture



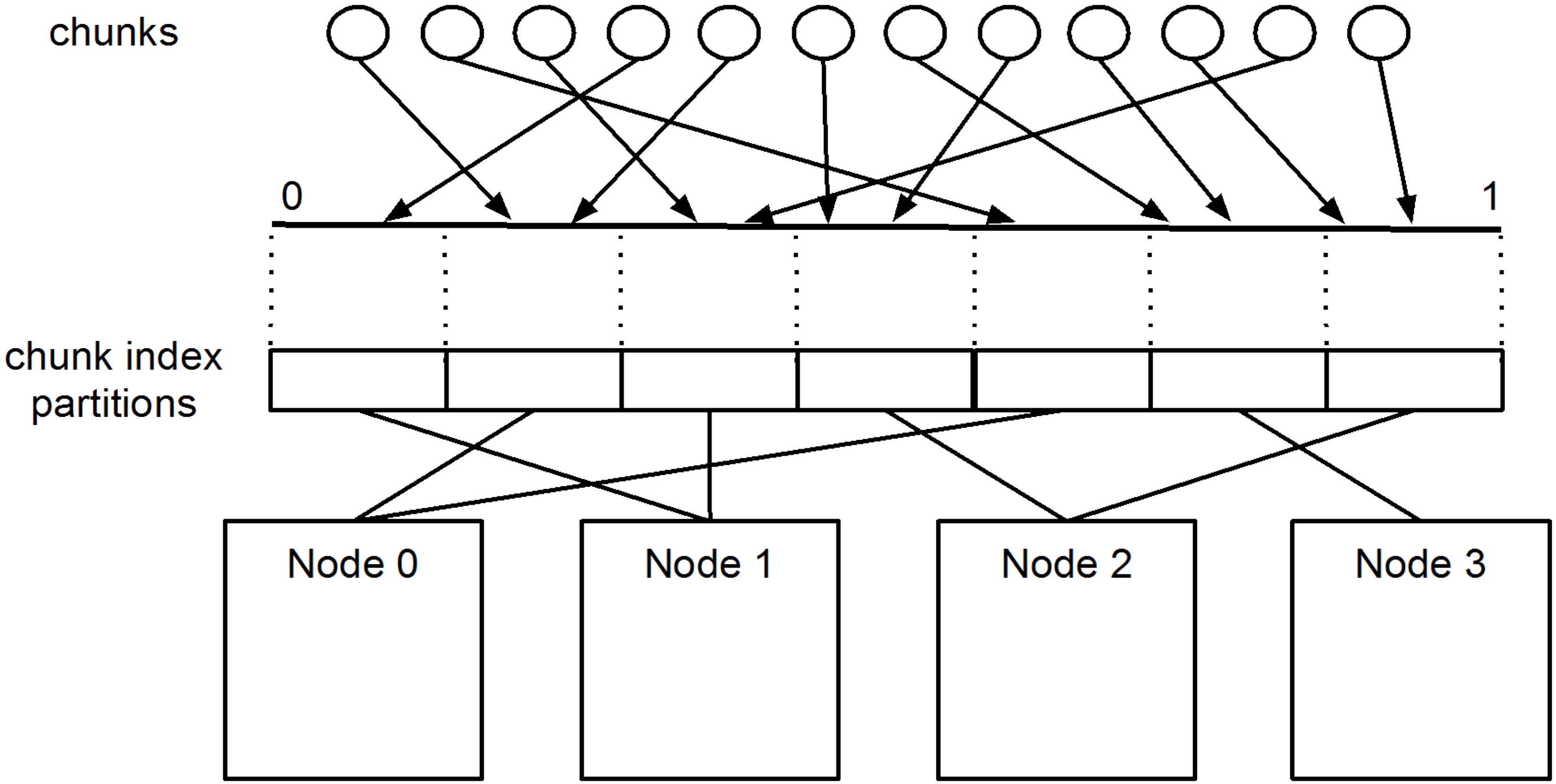
Load Balancing

- Chunk Index
  - Load balancing not necessary
  - SHA1 prefix ensures good distribution
- Block Index
  - Load imbalance due to skewed access
  - Move partitions between nodes to balance load
  - Move volume mapping as a last resort
- Container Storage
  - Load imbalance due to skewed read access
  - Move partitions between nodes to balance load

Network Limit?

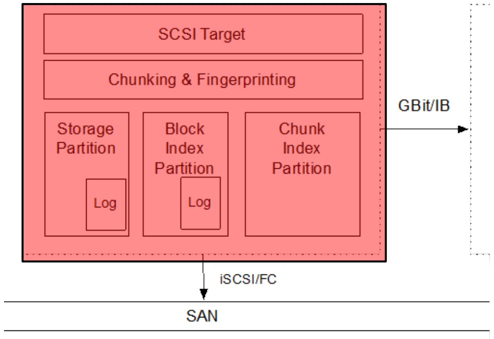
- Current limit: Nodes ability to receive and process messages
- Network switch can become bottleneck
  - High-performance switches are surprisingly capable
  - Probably only in larger clusters
  - Not seen any slowdown based on network / switch
    - Up to 24 nodes

Distributed Chunk Index



Deduplication Nodes

- Provide SCSI target interface
- Process incoming SCSI requests
- Chunking and Fingerprinting



- Contain parts of indexes
  - Chunk Index
  - Block Index

- Can directly access parts of stored chunk data
  - Container Storage