# iDedup: Latency-aware, inline deduplication for primary storage

*Kiran Srinivasan, Tim Bisson, Garth Goodson, Swetha Krishnan, Kaladhar Voruganti*
Advanced Technology Group, NetApp Inc.

## Introduction

***Goal**: Develop a inline deduplication technique to mitigate over-provisioning by saving space instantly while not affecting performance of primary workloads*

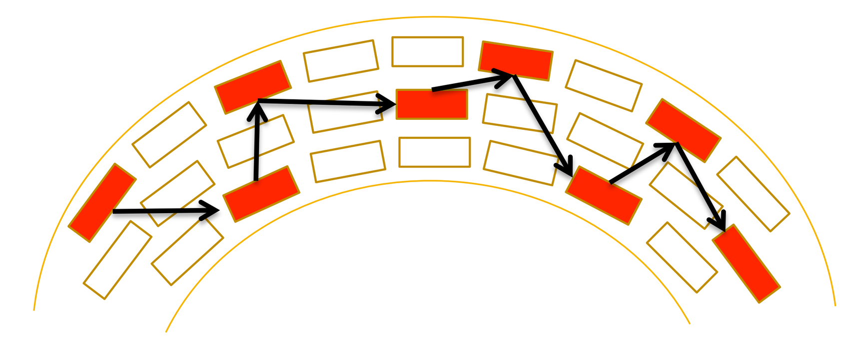
Why iDedup?

- Provisioning/Planning is easier
- Post-processing activities is optional
- Minimal performance impact
- Can be combined with offline dedupe

## Challenges- Reads

*Inherently, dedupe causes disk-level fragmentation !*

- *Sequential reads turn random => more seeks => more latency*
- RPC based protocols (CIFS/NFS/iSCI) are latency sensitive
- Fragmentation is a dataset/workload property

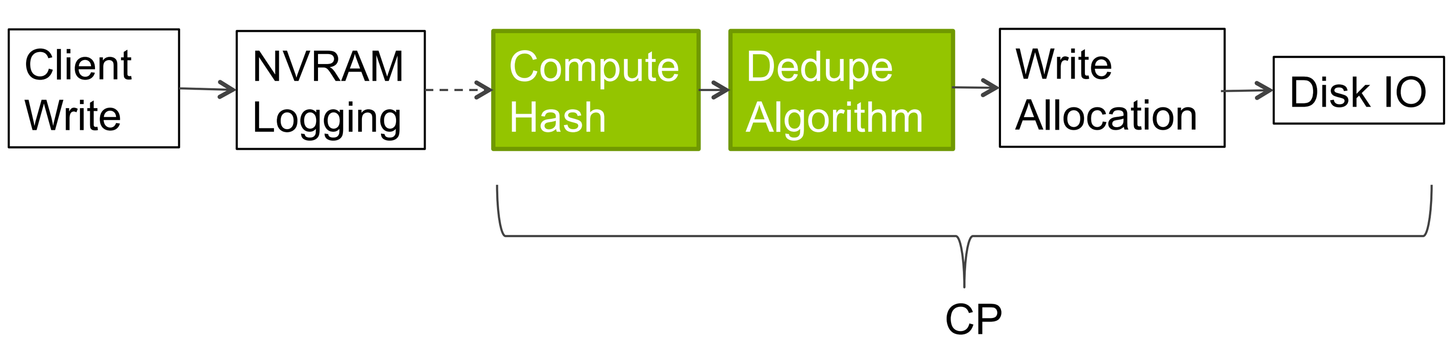


Fragmentation with random seeks

## Challenges- Writes

*CPU overheads in the critical write path*

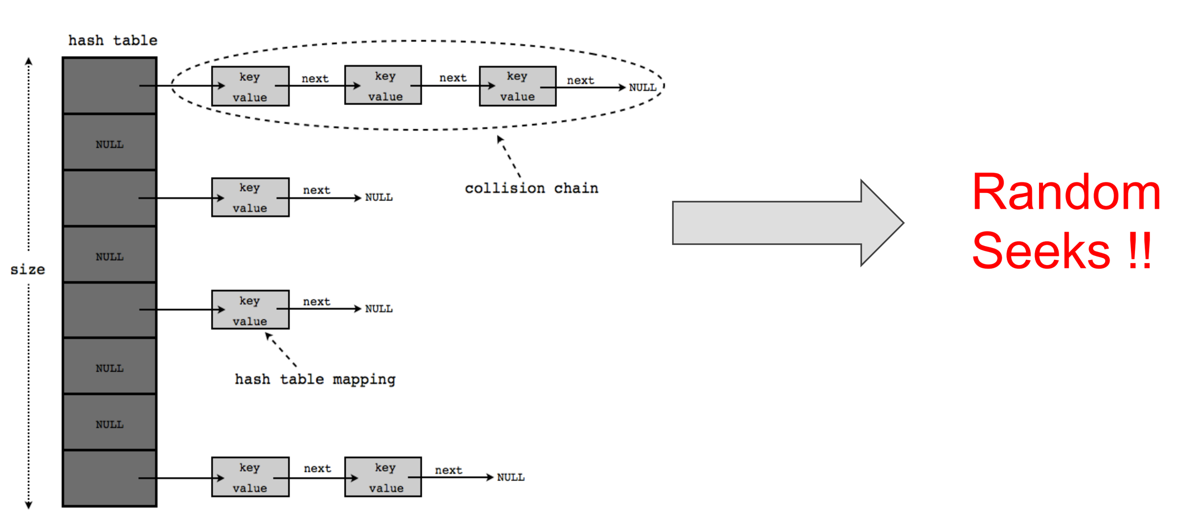
- Dedupe requires computing hash of each block
- Dedupe algorithm requires extra cycles




*Extra random I/Os due to dedupe algorithm*

- On-disk Dedupe metadata (FingerPrint DB) accesses
- Updating the refcount file

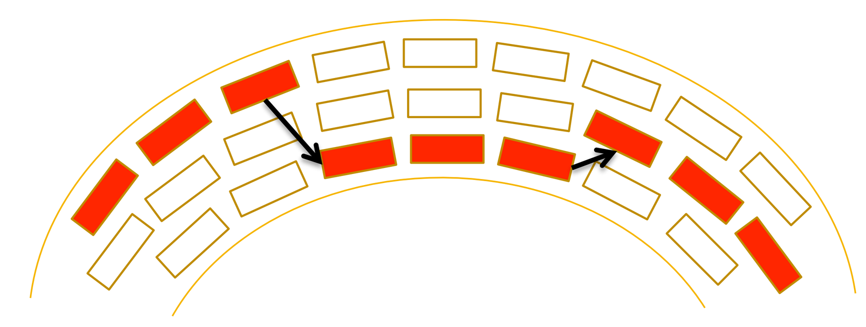
*Dedupe metadata (FPDB)*




## Solution

*Insight 1: Dedupe only sequences of duplicate blocks*

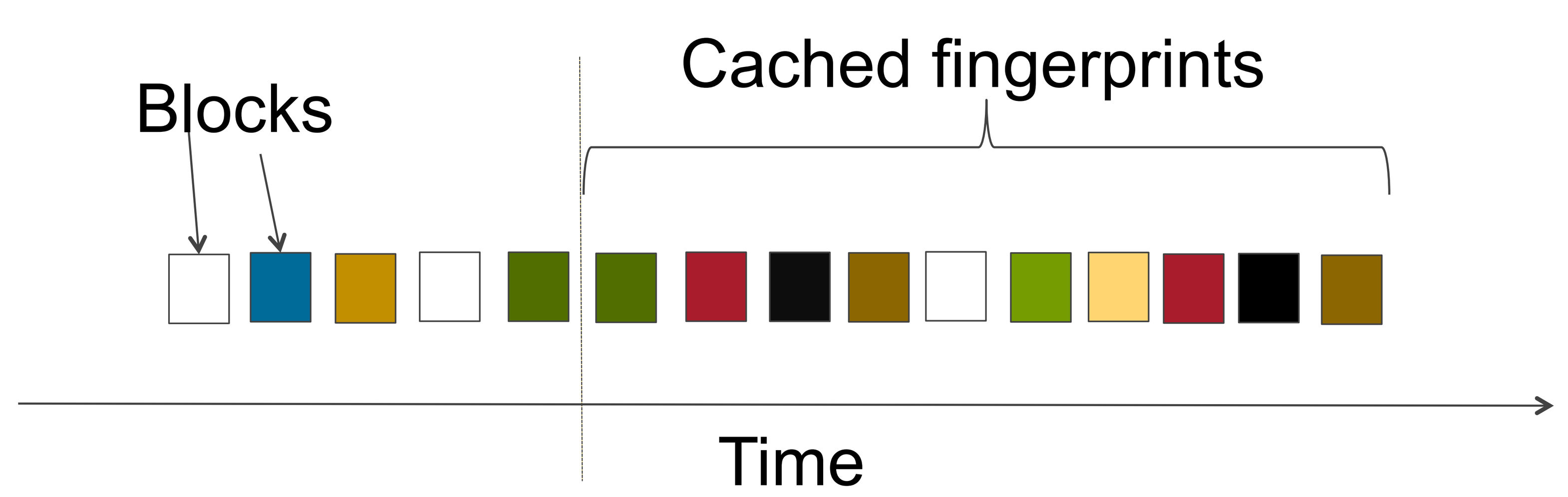
- Solves fragmentation => amortized seeks
- Configurable minimum sequence length - Threshold
- Selective dedupe, leverages *spatial locality*



Sequences, with amortized seeks

*Insight 2: Keep a smaller FPDB as an in-memory cache*

- No extra I/Os, leverages temporal locality characteristics
- FPDB keeps a subset of all blocks => some loss in dedupe




## Evaluation

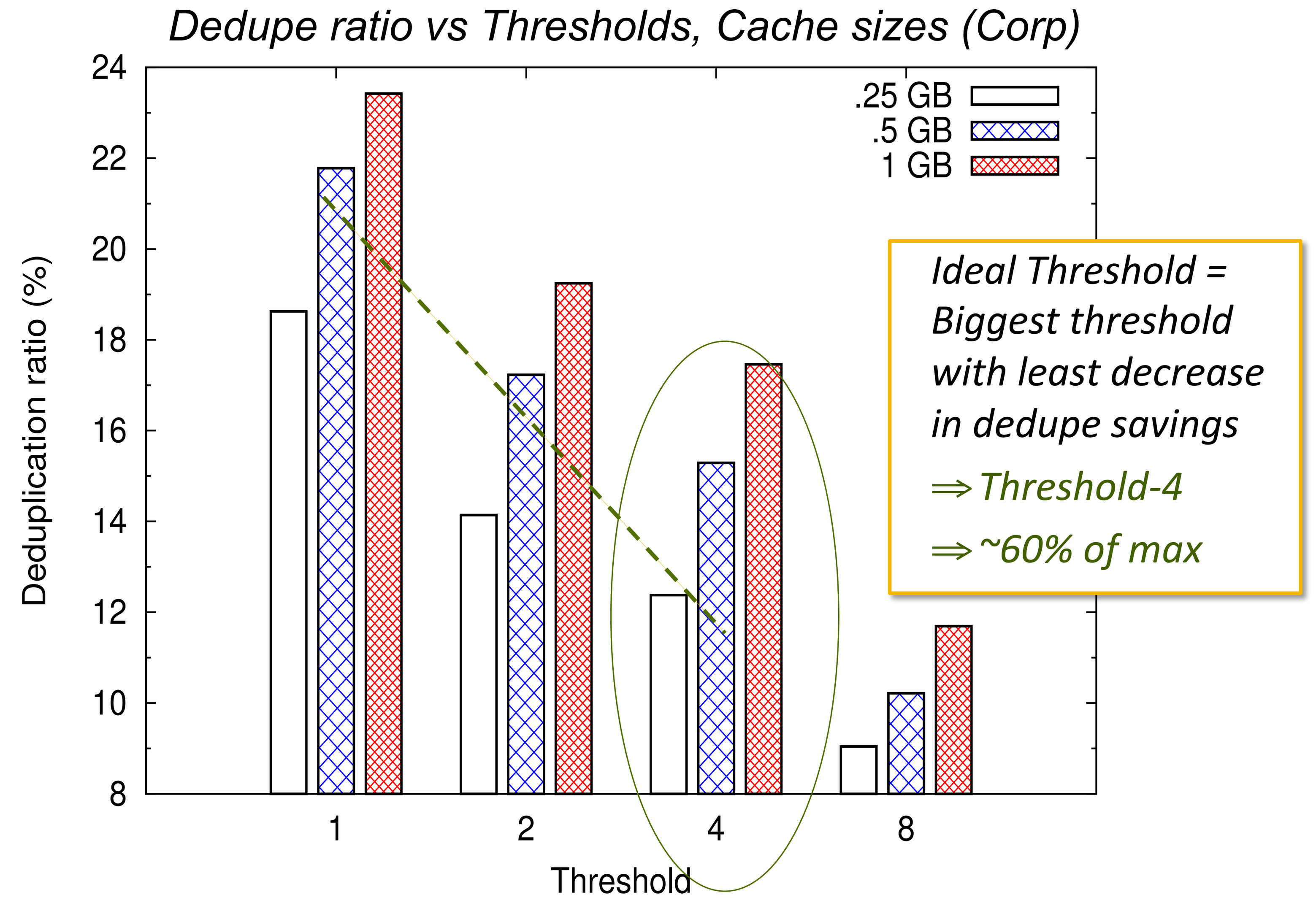
*Evaluated by replaying CIFS traces (NetApp DC)*

- Corporate traces: 204GB Reads, 93GB Writes
- Engineering traces: 192GB Reads, 92GB Writes

*Design parameters*

- Threshold sizes – 1, 2, 4, 8
- Dedupe metadata cache size – 0.25GB, 0.5GB, 1GB
- Baseline - System with iDedup disabled

*Dedupe ratio vs Thresholds, Cache sizes (Corp)*




*CDF of block request sizes (Engg, 1GB)*