Building a High Performance Deduplication System

Fanglu Guo and Petros Efstathopoulos

Symantec Research Labs

Symantec

FY 2013 (4/1/2012 to 3/31/2013) Revenue: $ 6.9 billion

Segment	Revenue	Example Business
Consumer	30%	Norton security, Norton Zone
Security and compliance	30%	Symantec Endpoint Protection, Mail/Web security, Server protection
Storage and server management	36%	Backup and recovery, Information availability
Services	4%	Managed security service

Symantec Research Labs

- Leading experts in security, availability and systems doing innovative research across all of Symantec's businesses

"Our mission is to ensure Symantec's long-term leadership by fostering innovation, generating new ideas, and developing next-generation technologies across all of our businesses."

- A global organization:

Mountain View, CA | Culver City, CA | Herndon, VA | Waltham, MA | Sophia Antipolis, FR

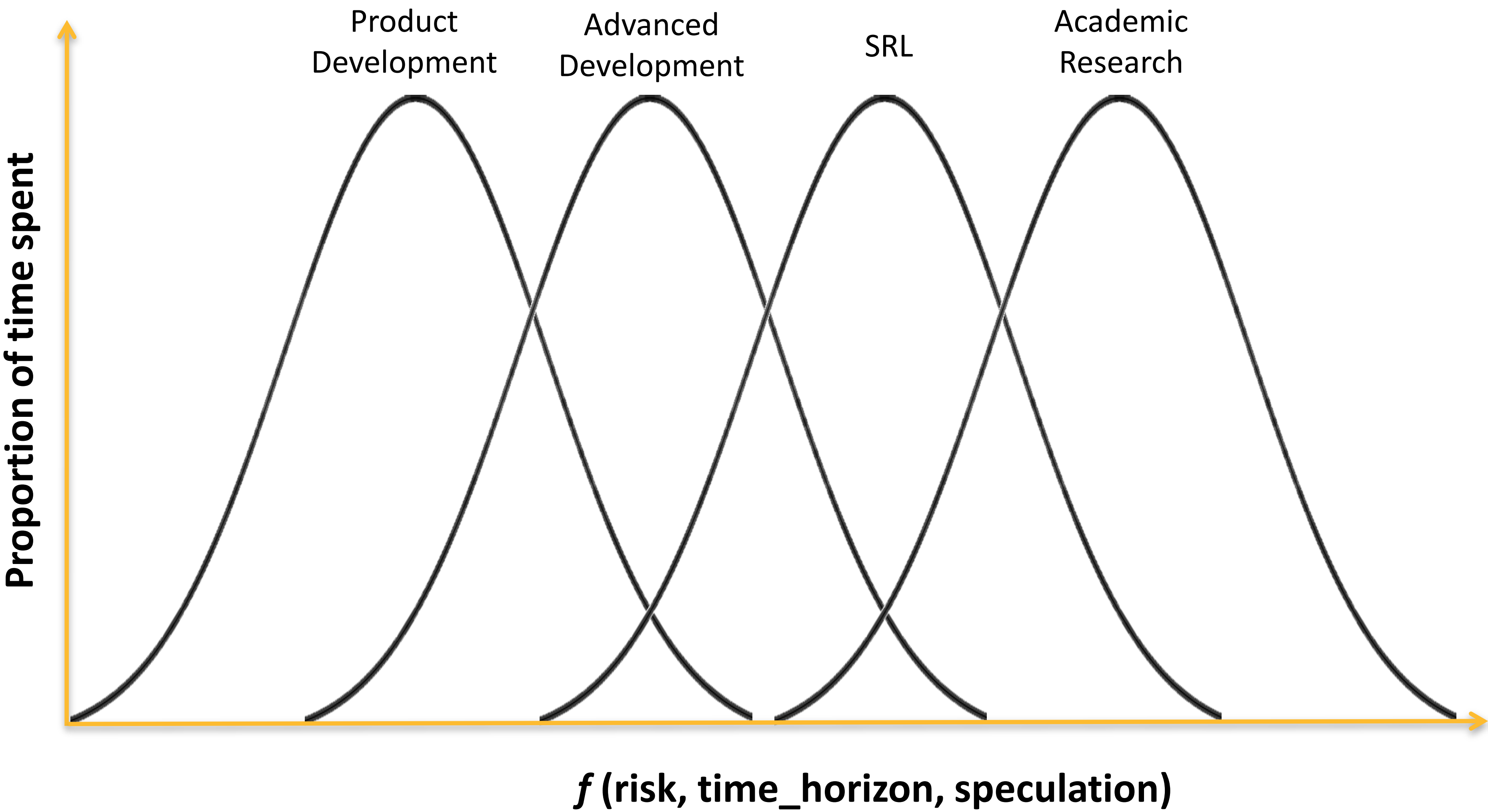
- Ongoing collaboration with other researchers, government agencies and universities such as:

U.S. Department of Homeland Security · NSF · European Union · Stanford University · Carnegie Mellon … and numerous others

Symantec Research Labs

- Charter
  - Foster innovation and develop key technologies
- Operational Model
  - Work closely with existing and emerging businesses
  - Identify and work on relevant hard technical challenges
- Success Factors
  - Transfer technology
  - Develop new intellectual property
  - Forge research collaborations and publish innovative work

Symantec Research Labs (SRL)



Building a High Performance Deduplication System

Deduplication is Maturing

- Deduplication (*dedupe*) is becoming a standard feature of backup/archival systems, file systems, cloud storage, network traffic optimization, …
- Many approaches, algorithms, techniques
  - Inline/offline, file/block level, fixed/variable block size, global/local dedupe
  - Near-optimal duplicate detection and usage of available raw storage
- However, in practice, scalability is still an issue
  - Single node capacity < a few hundred TB
- Scalability achieved (mostly) using multi-node systems
  - Capacity still relatively low (usually single-digit PB)
  - High acquisition/maintenance cost
  - Complex design, performance impact

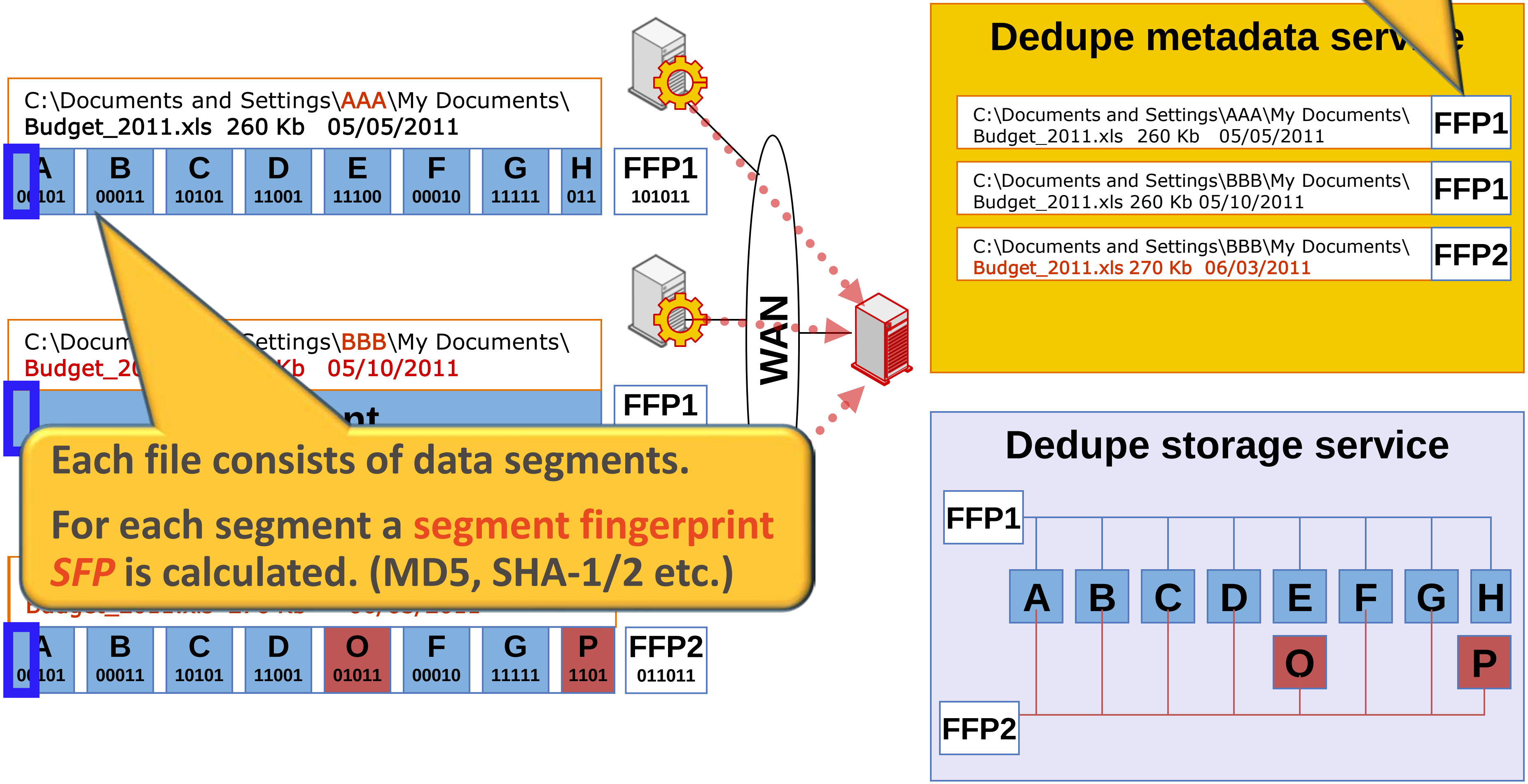
Hypothesis and Goal

We advocate that improving single-node performance is critical

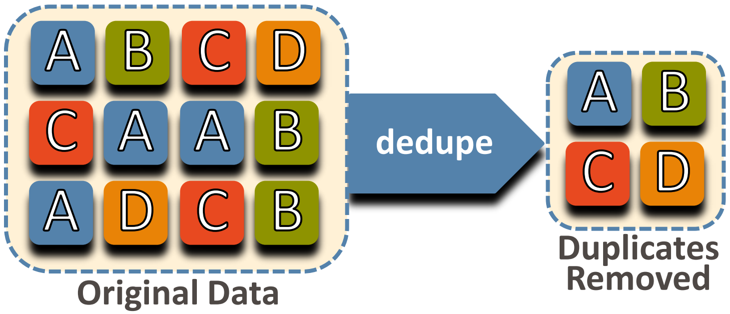
- Why?
  - Better building blocks – fewer nodes necessary
  - Potential for single-node deployments – for small/medium businesses
  - Lower acquisition/management/energy cost
  - Hosting/relocation flexibility
- How?
  - Single-node performance = high scalability and throughput, good dedupe efficiency
  - Making the best of a node's resources to address challenges
- Built and tested ***a complete deduplication system*** from scratch
  - Including client, server and network components

Core Deduplication Functionality

Each file is identified by a **file fingerprint *FFP***



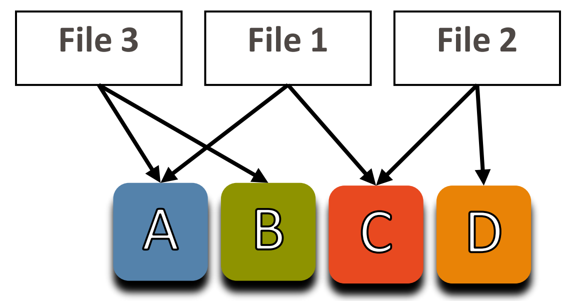
Challenges Addressed



Have I seen segment X before?
(and if yes, where is it stored?)

- Index: maps SFPs to disk location
- **Segment indexing challenges**
  - System capacity bound by indexing capacity
  - Increasing the segment size not a good solution
  - System performance bound by index speed
- **Reference management challenges**
  - Need to keep track of who is using what
  - Capacity bound by the ability to track references and manage/reclaim segments
  - Reference update performance may also become a bottleneck
- **Client and network performance challenges**

Item	Scale	Remarks
Capacity	C = 1000 TB	
Segment size	S = 4 KB	
Num of segments	N = $250*10^9$	N = C/S
Metadata per segment	E = 22 B	
Total segment metadata	I = 5500 GB	I = N*E
Disk speed	Z = 400 MB/sec	
Lookup speed goal	100 Kops/sec	Z/S



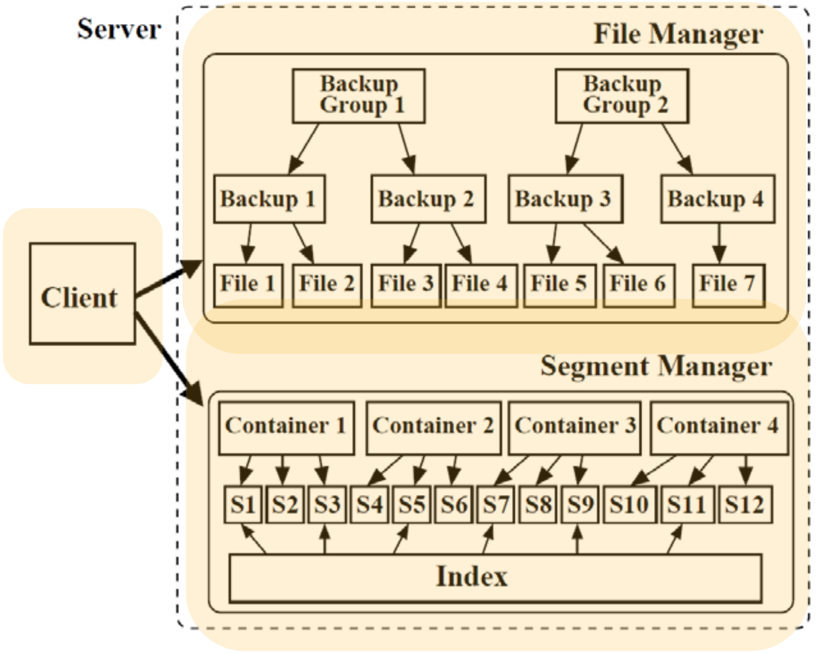
Performance Goals

- **Single-node scalability goals**
  - 100+ billion objects per node, with high throughput
- **Single-node performance goals**
  - Near-raw-disk throughput for backup, restore and deletion (i.e., space reclamation)
  - Reasonable deduplication efficiency
- **Client-server interface capable of delivering desired throughput**
- *Assumptions*:
  - Opportunity for improving duplicate detection is becoming scarce
  - Aiming for perfect duplicate detection may limit scalability
    - Willing to trade *some* deduplication efficiency for high scalability

Outline

1. Architecture Overview
2. Progressive Sampled Indexing
3. Grouped Mark-and-Sweep
4. Evaluation

Architecture Overview – Design



- **Client – Server architecture**
  - Client component may reside on server
- **Client**
  - Initiates backup creation/restore
  - Performs client I/O
  - Performs segmentation, fingerprint calculation
  - Issues fingerprint lookups to server – initiates data transfers only for new segments
- **Server File Manager**
  - Hierarchical list of backup and file metadata
  - Central concept: ***backup*** – represents a list of files
  - Organizes backups into (roughly equal-size) ***backup groups***
- **Server Segment Manager**
  - Manages storage units, called ***containers***, and implements storage logic
  - Hosts the fingerprint index – responsible for duplicate detection
  - Performs reference management operations

Architecture Overview – Main Concepts

- Containers represented by unique container ID (CID)
- Container contents: raw data segments + catalogue of segment SFPs

- File represented by a list of <SFP, CID> pairs
- Identified by its file fingerprint FFP = hash(SFP1, SFP2, …, SFP4)

FFP → File = (< SFP1, CID1 > < SFP2, CID1 > … < SFP3, CID2 > < SFP4, CID3 >)

- Notice that file segment location stored inline
  - Data segments directly locatable from file metadata
  - No need for location services by the index – e.g., at restore time

Sampled Indexing

- Directly locatable objects + relaxed dedupe goals
- No requirement for a complete index → freedom to do sampling

- Sampled Indexing: keep 1 out of ***T*** SFPs
  - E.g., “modulo T” sampling

- ***Sampling rate R*** (“fullness” level) = 1/T =

  = function (RAM, storage capacity, desired segment size)

Progressive Sampled Indexing

- Estimate of sampling rate **R = (M/E) / (C/S)**
  - M = memory GBs,   E = bytes/entry → M/E = total index entries
  - C = storage TBs, S = segment size KBs → C/S = total segments
- Example: 32 bytes/entry, 4KB segments and 32GB of RAM
  - No sampling (R=1) → 4 TB storage
  - R ~= **1/100** – i.e., "keep 1 out of 100 SFPs" → 400 TB
- ***Progressive sampling:*** used storage VS available storage
  - Sampling rate = function(***used storage***, available RAM)
  - Start with no sampling (R=1)
  - Progressively decrease R, down to R_{min} = (M/E) / (C/S)

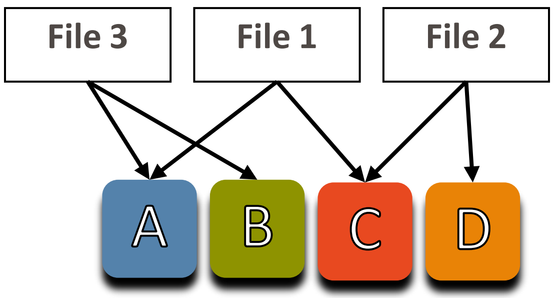
Fingerprint Caching

- Straight sampling → poor deduplication efficiency
  - Only 1 out of T segments deduplicatable

- **Solution: take advantage of spatial locality**
  - Index hit → <SFP, CID> → use CID to pre-fetch SFPs in container catalogue
  - Part of index memory: **SFP cache**
  - Lower bound for sampling rate: at least one sampled entry per container

Reference Management Challenges and Goals

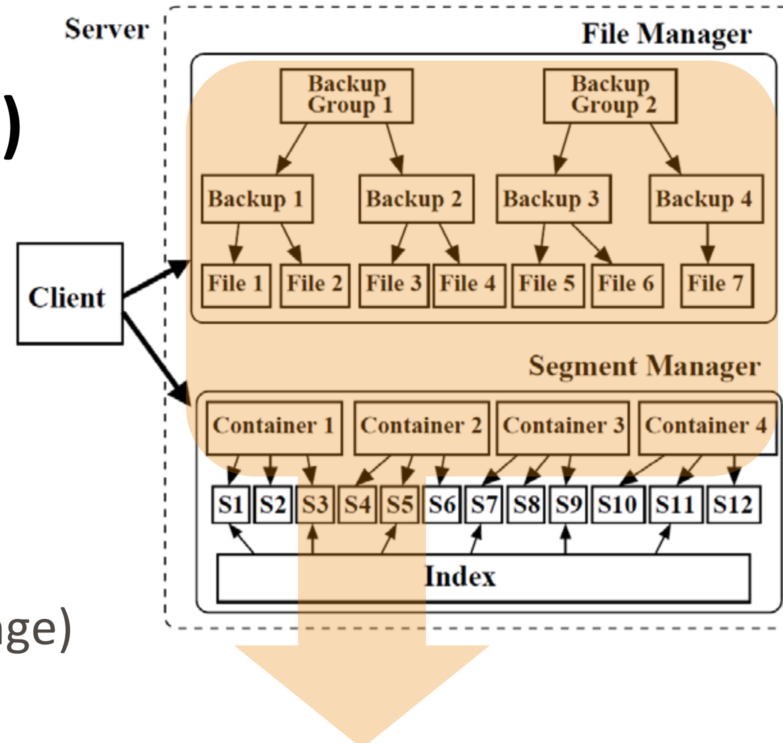
- Problem: *"Is it safe to delete a segment? Is anyone using it?"*
- Challenge: do it ***correctly*** and ***efficiently***

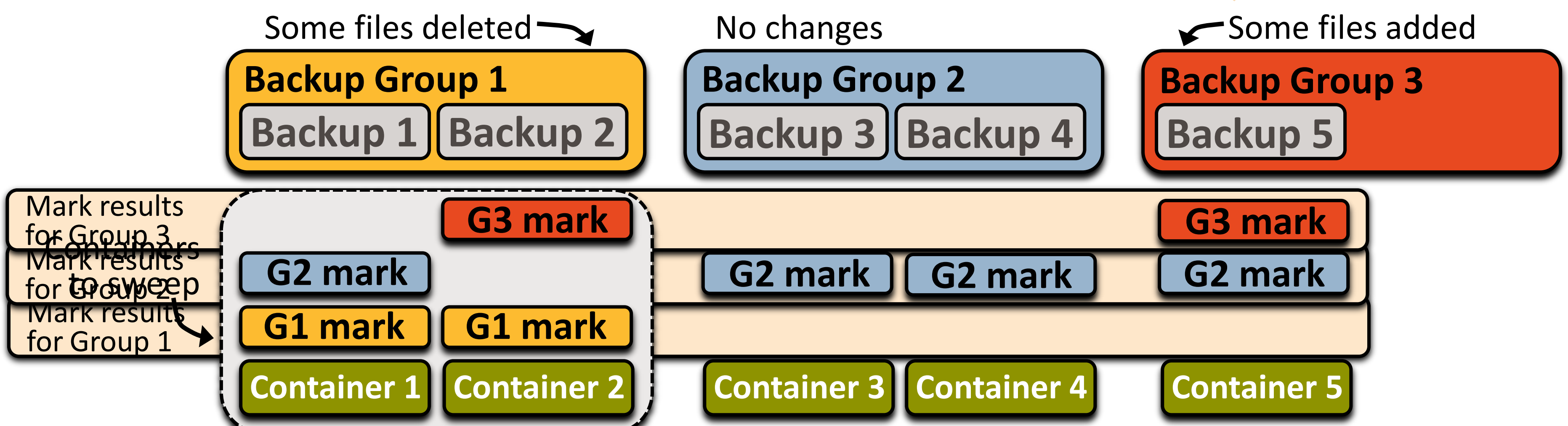


  - Mistakes can not be tolerated – e.g., after a crash
  - Potentially used in a distributed environment
  - Performance requirements: > 100 Kops/sec
- Reference counting: efficient but hard to guarantee correctness
  - Repeated and/or lost updates lead to irrecoverable errors
- Reference lists: repeated updates solved, lost updates still a problem
  - Very inefficient (list maintenance, serialization to disk, etc.)
- Mark-and-Sweep (aka Garbage Collection)
  - Workload proportional to system capacity
    - E.g., 400 TB system, 4 KB segments, 22 byte SFP → ~2.2 TB read from disk during mark phase
    - x10 dedupe factor → ~ 22 TB

Grouped Mark-and-Sweep (GMS)

- Mark: divide and save
  - Track changes to backup groups
  - Only re-mark changed groups
  - Mark results saved and reused
- Sweep: track affected containers
  - Only sweep containers that may have deletions
- Result: workload = function(amount of change)
  - Instead of system capacity





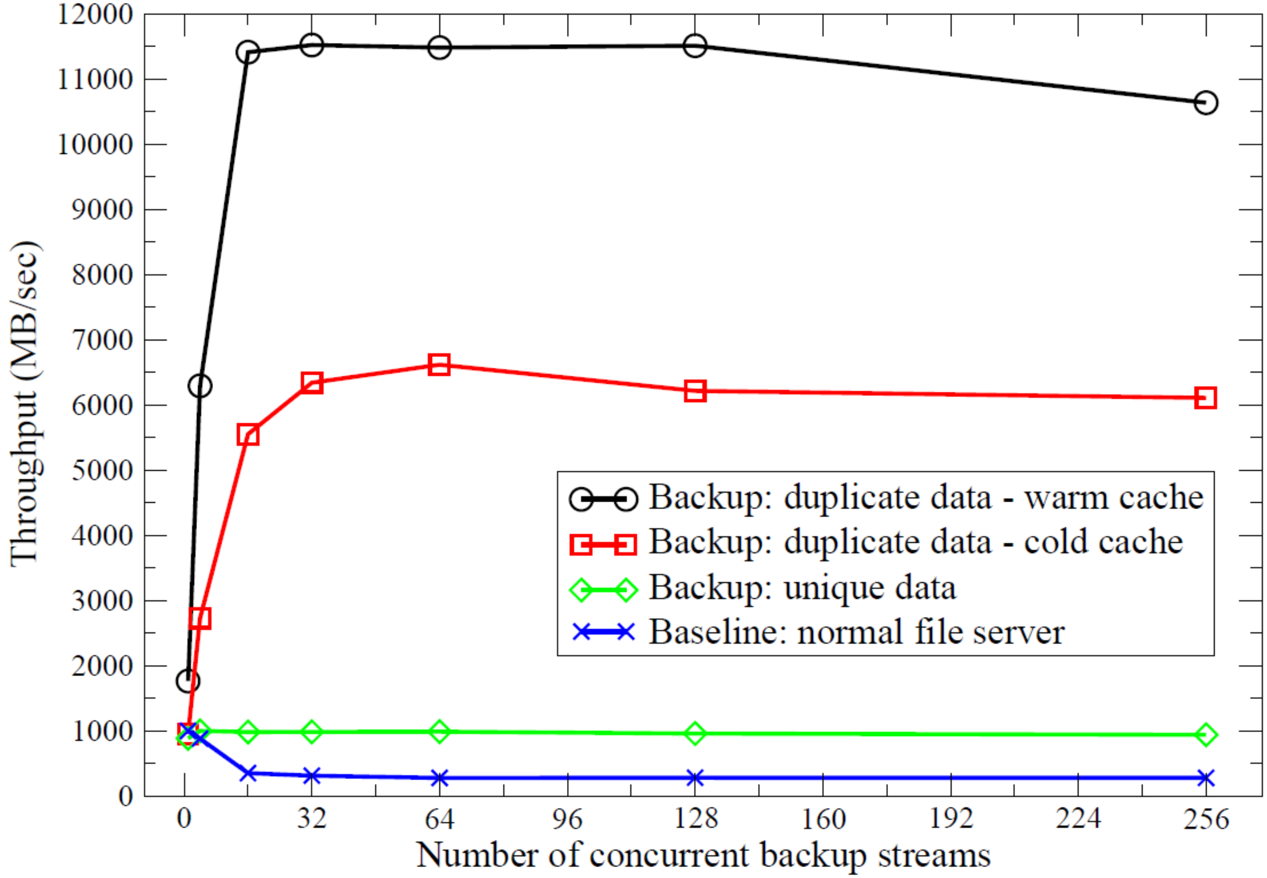
Deduplication Server Evaluation

- Evaluation metrics: single-node ***throughput***, ***scalability*** and ***dedupe efficiency***

- Implemenation in C++, using Pthreads
- Portable (Linux and Windows)
- Index implementation: pointer-free, three cache eviction policies
- 8-core server, 32 GB RAM, 24 TB disk array, 128 GB PCI-X SSD, 64-bit Linux
- Baseline disk array throughput: ~1,000 MB/sec (both read/write)
- Two types of data-sets:
  - Synthetic data-set: multiple 3 GB files 100% unique segments
  - Virtual machine image files: 4 versions of a "golden image"
- Evaluation configuration: R = 1/101, 25 GB index
  - 200 TB per 10 GB of RAM (500 TB in our case)

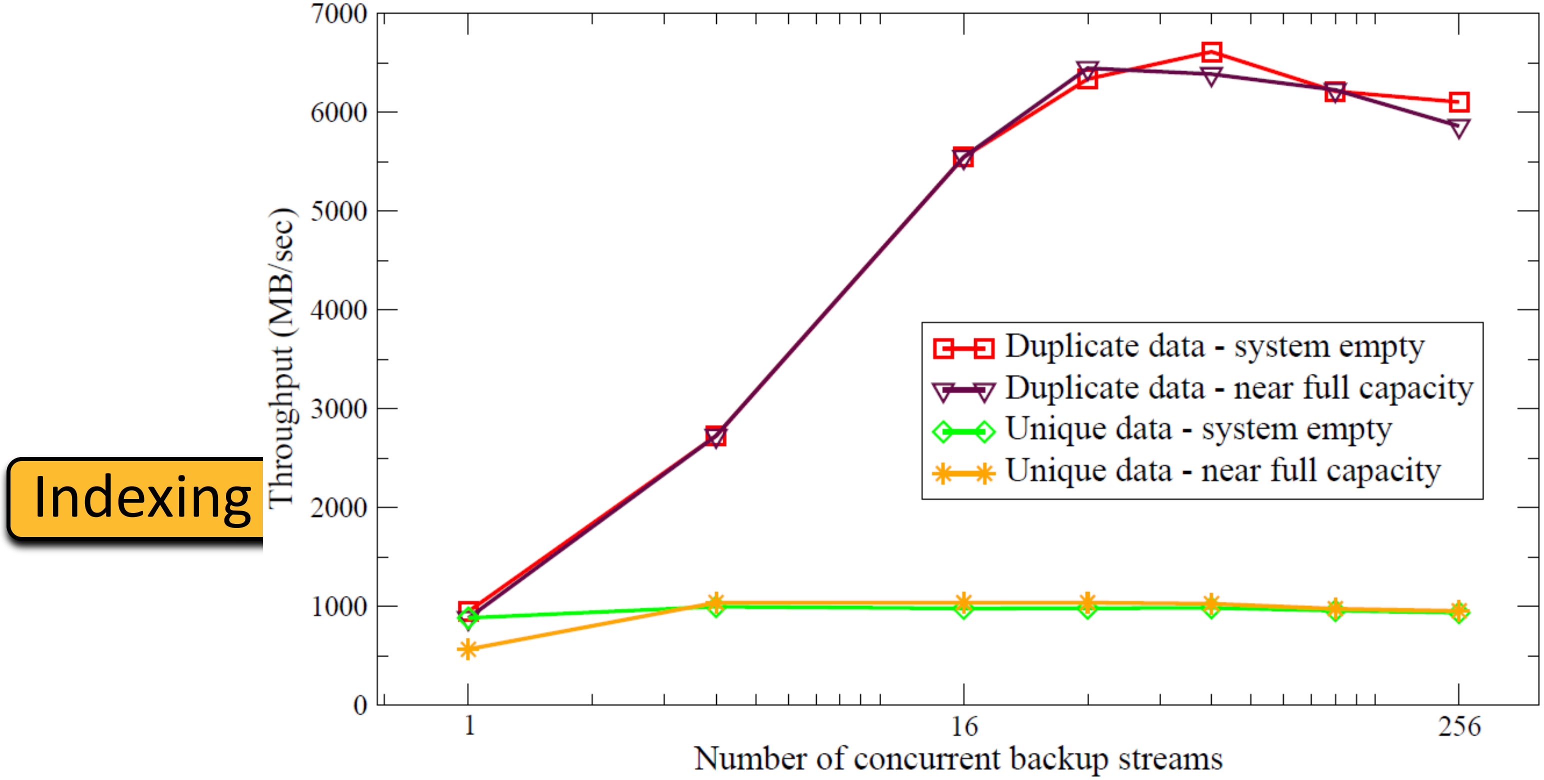
Backup Throughput: Synthetic data

- Multiple backup streams of synthetic data
- Concurrency levels: 1, 16, 32, 64, 128, 256 backup streams



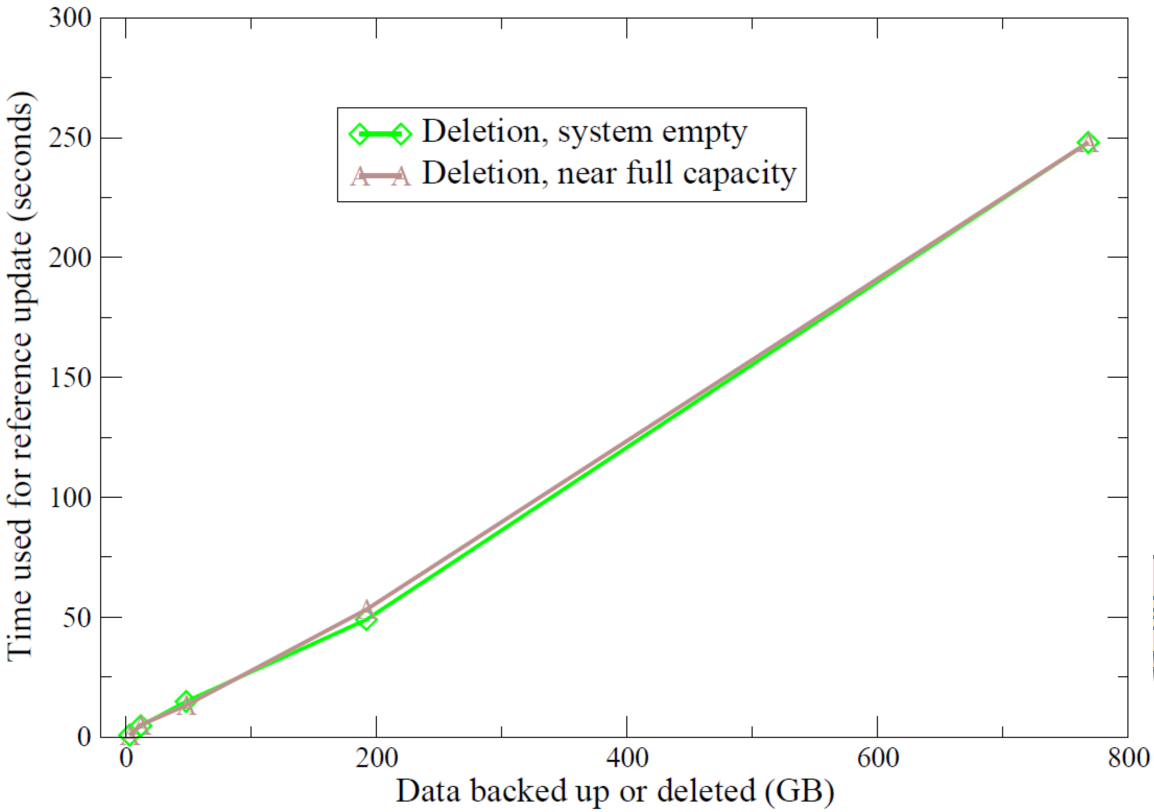
Backup Scalability

- Populate system at 95% capacity (480 out of 500 TB)
  - All containers created, container catalogues, metadata stored, but no raw data
- Repeated throughput tests



Grouped Mark-and-Sweep Throughput and Scalability

- Reference update performance is critical – happens daily!
- Tests: backup add and delete (reference addition and deletion)
  - When system is empty and when near-full



ghput ~= 3.1 GB/sec

Deduplication Efficiency

- Synthetic data-set: multiple backups, no less than 97%
- VM images a very common and important dedupe workload
- Data-set:
  - VM0: “Golden image”, corporate WinXP SP2 + some utilites
  - VM1: VM0 + all Microsoft Service Packs, Patches etc.
  - VM2: VM1 + large anti-virus suite
  - VM3: VM2 + more utilities (pdf readers, compression tools etc.)
- Calculated “ground truth” for all image segments (unique/duplicates)

	VM Unique Segments	Globally Unique	Ideal Disk Usage (MB)	Actual Disk Usage (MB)	Success Rate
VM0	518,326	518,326	2,123	2,211	96%
VM1	733,267	921,522	3,775	3,938	96%
VM2	904,579	1,189,230	4,871	5,085	96%
VM3	1,145,029	1,616,585	6,621	6,860	97%

Conclusions

- We have built and tested a complete deduplication system
  - Scale: hundreds of TBs per node
  - Backup throughput: ~1,000 MB/sec (unique data), ~6,000 MB/sec (duplicates)
  - Reference management throughput: more than 3 GB/sec
  - Deduplication efficiency: ~ 97% of duplicates detected
- We introduced new mechanisms for scalability & performance
  - Sampled (SSD) indexing, grouped mark-and-sweep, pipelined client architecture
- Our results demonstrate that high single-node deduplication server performance is possible

Status

- Most of the technologies are in use in the product. Make impact!
- Symantec is increasing R&D investment. We are hiring!
- Fellowship

Thank you!

Questions?