Cumulus: Filesystem Backup to the Cloud

Michael Vrable, Stefan Savage, and Geoffrey M. Voelker

Presented by Hakim Weatherspoon

Good ‘ole Days

Replace your tape drives
with something truly scalable

Amazon S3 to the rescue

In Spring 2006, Amazon released a new storage API: Put, Get, List, Delete

Build whatever you want!

Quickly

Backing up the new way (S3)

- ***Smart***
- ***Scales***
  - ***- no longer our concern... Amazon's concern***
  - ***- all servers backup in parallel***
- ***Cheap***
  - ***- old cost = XXX per year***
  - ***- new cost = YYY per year***
  - ***where YYY < XXX***

Thin vs Thick Cloud

- E.g. Amazons S3 vs EMC's MozyPro
- Thin
  - Can change provider easier
  - Applications can work across providers
- Thick
  - Better performance
  - Locked into a provider
  - Provider can go out of business

Cumulus

- Simple storage backup utility for Thin Clouds
- Evaluates efficacy of cloud storage
- Working prototype
  - http://www.cs.ucsd.edu/~mvrable/cumulus/

Outline

- Motivation/Intro
- Related Work
- Design
- Evaluation
- Thoughts and Conclusions

Related Work

	Multiple snapshots	Simple server	Incremental forever	Sub-file delta storage	Encryption
rsync			✓	N/A	
rsnapshot	✓		✓		
rdiff-backup	✓		✓	✓	
Box Backup	✓		✓	✓	✓
Jungle Disk	✓	✓	✓		✓
duplicity	✓	✓		✓	✓
Brackup	✓	✓	✓		✓
Cumulus	✓	✓	✓	✓	✓

Outline

- Motivation/Intro
- Related Work
- Design
  - API
  - Segments
  - Snapshots
  - Subfile incrementals
  - Cleaning
  - restoring
- Evaluation
- Thoughts and Conclusions

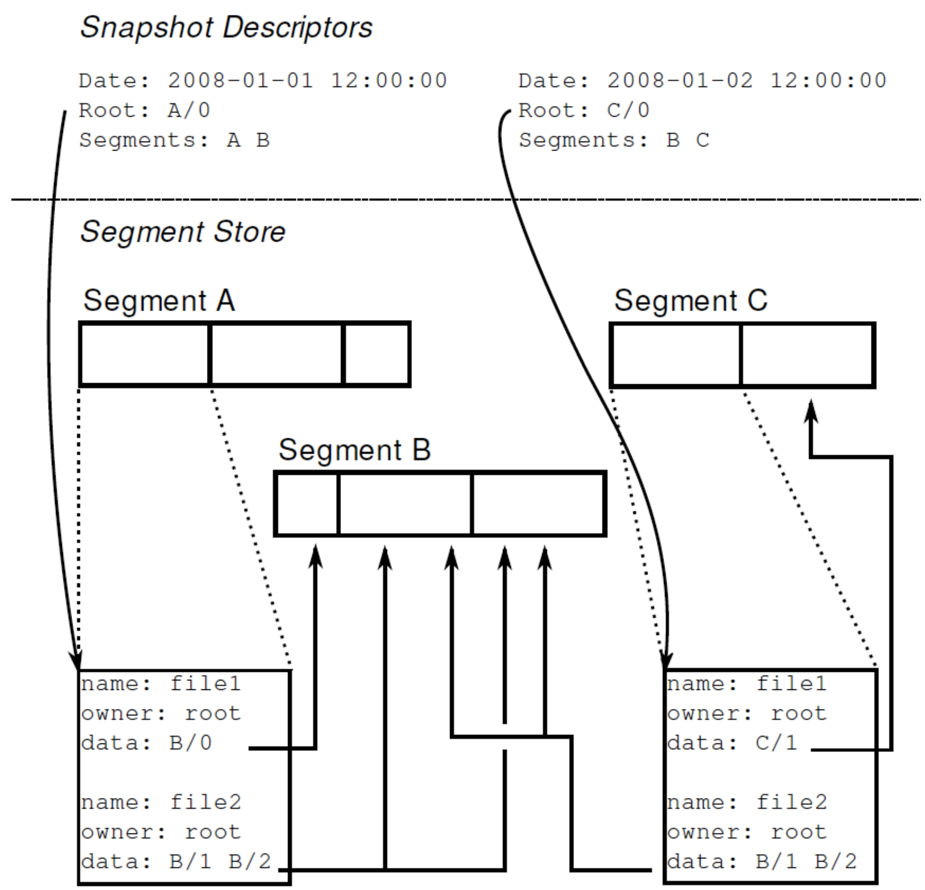
API

- Same as S3
  - Put, Get, List, Delete
- *Thin cloud* – does not rely on integrated services
  - Can easily change provider and network protocols
  - S3, FTP, SFTP
- WORM Model
  - Write-once, read-many
  - Requires writing new entirely file if changes occur
  - What are the cleaning overheads?

Segments

- Aggregation via Segment Goals
  - Avoid costs due to small files
    - S3 charges on per file bases
    - Many small files
  - Avoid costs in network protocols
    - Small files have higher latency and other overheads
  - Compression
    - inter-file similarities
  - Privacy
    - Hide file boundaries
- Negative consequences?
  - Need an entire segment to write

Snapshots



Snapshot Descriptors
Date: 2008-01-01 12:00:00
Root: A/0
Segments: A B
Date: 2008-01-02 12:00:00
Root: C/0
Segments: B C
Segment Store
Segment A
Segment B
Segment C
name: file1
owner: root
data: B/0
name: file2
owner: root
data: B/1 B/2
name: file1
owner: root
data: C/1
name: file2
owner: root
data: B/1 B/2

Sub-File Incrementals

- Only stored changed part of files
- New snapshots point to old objects when data unchanged
  - Byte ranges – portions of old objcts to be reused

Segment Cleaning

- Similar to a log-structured file system (LFS)
- Clean based on utilization of segment, α
  - α = 0, no cleaning
  - α = 1, clean with the slightest change
- Cumulus
  - attempts to find an equilibrium for α
  - Uses a different process to clean
  - Marks a local database as "expired"
  - Then, next snapshot will not refer to expired segment

Restore

- Full Restore
  - Download all segments for a snapshot
- Partial Restore
  - Download snapshot descriptor, metadata, and only necessary segments

- What happens if client machine dies?
- How is latest snapshot descriptors identified?
- What about sharing between client machines?

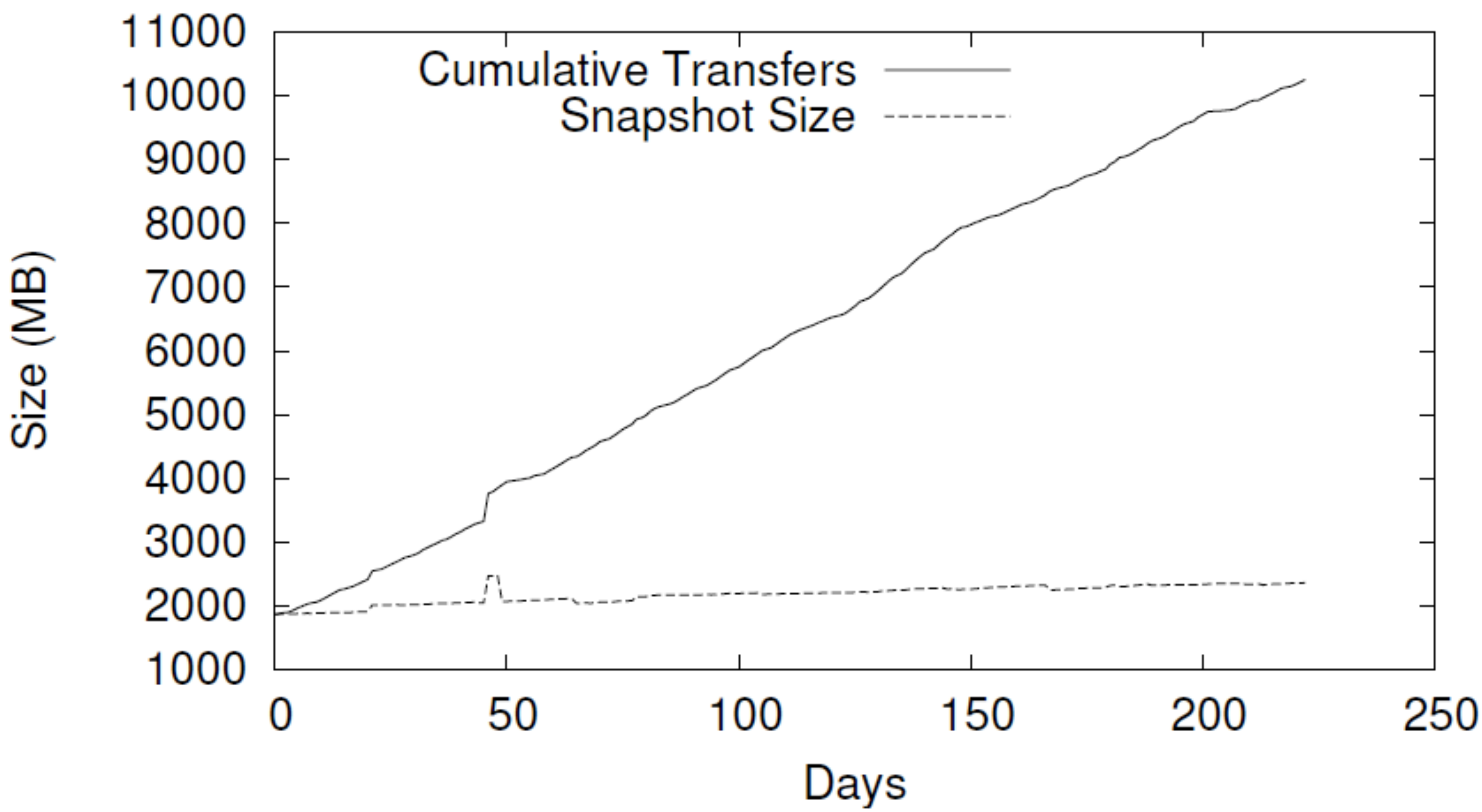
Outline

- Motivation/Intro
- Related Work
- Design
- Evaluation
  - Performance Case Study
  - Monetary Case Study
- Thoughts and Conclusions

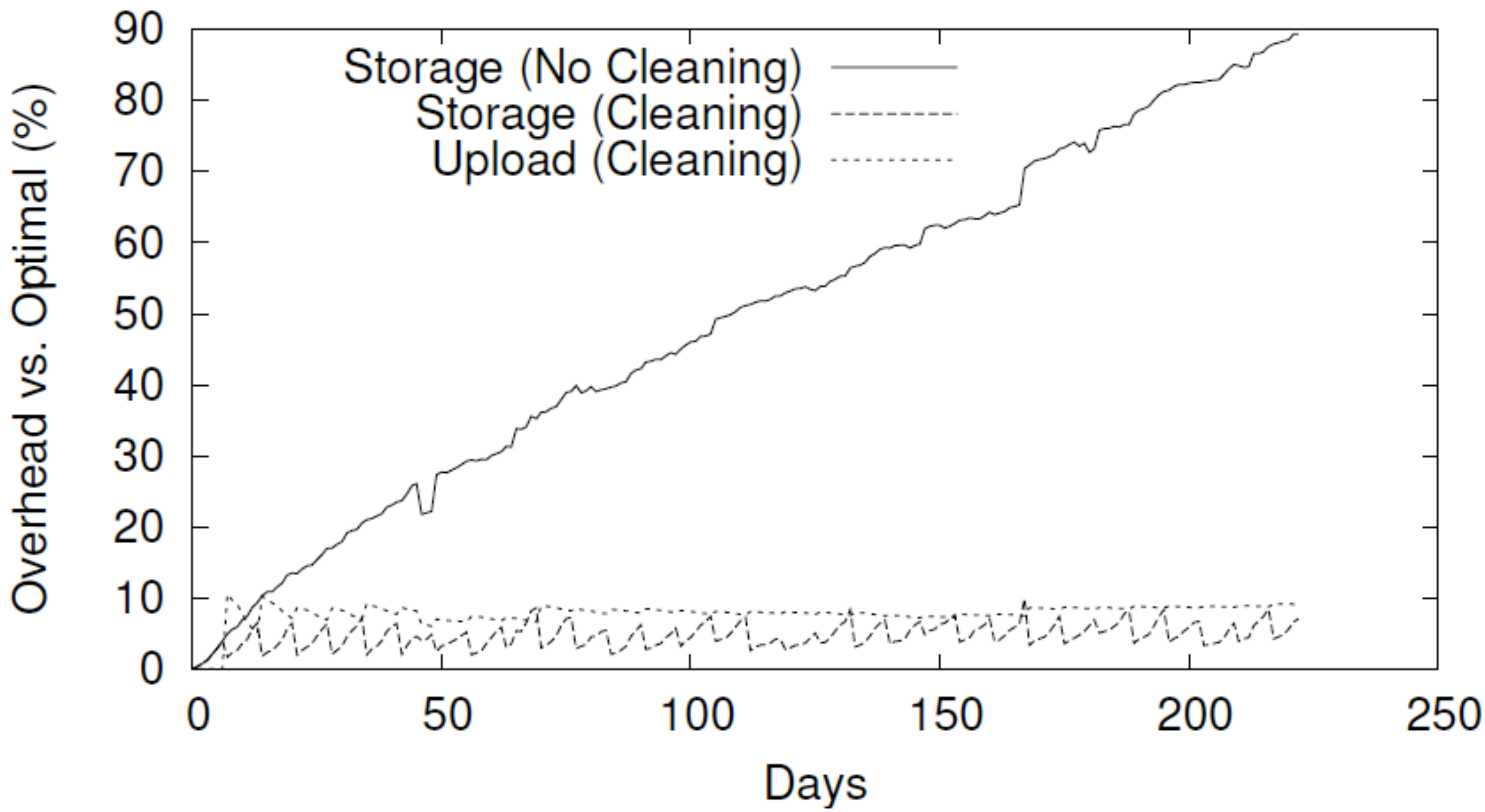
Evaluation Traces

	Fileserver	User
Duration (days)	157	223
Entries	26673083	122007
Files	24344167	116426
File Sizes		
Median	0.996 KB	4.4 KB
Average	153 KB	21.4 KB
Maximum	54.1 GB	169 MB
Total	3.47 TB	2.37 GB
Update Rates		
New data/day	9.50 GB	10.3 MB
Changed data/day	805 MB	29.9 MB
Total data/day	10.3 GB	40.2 MB

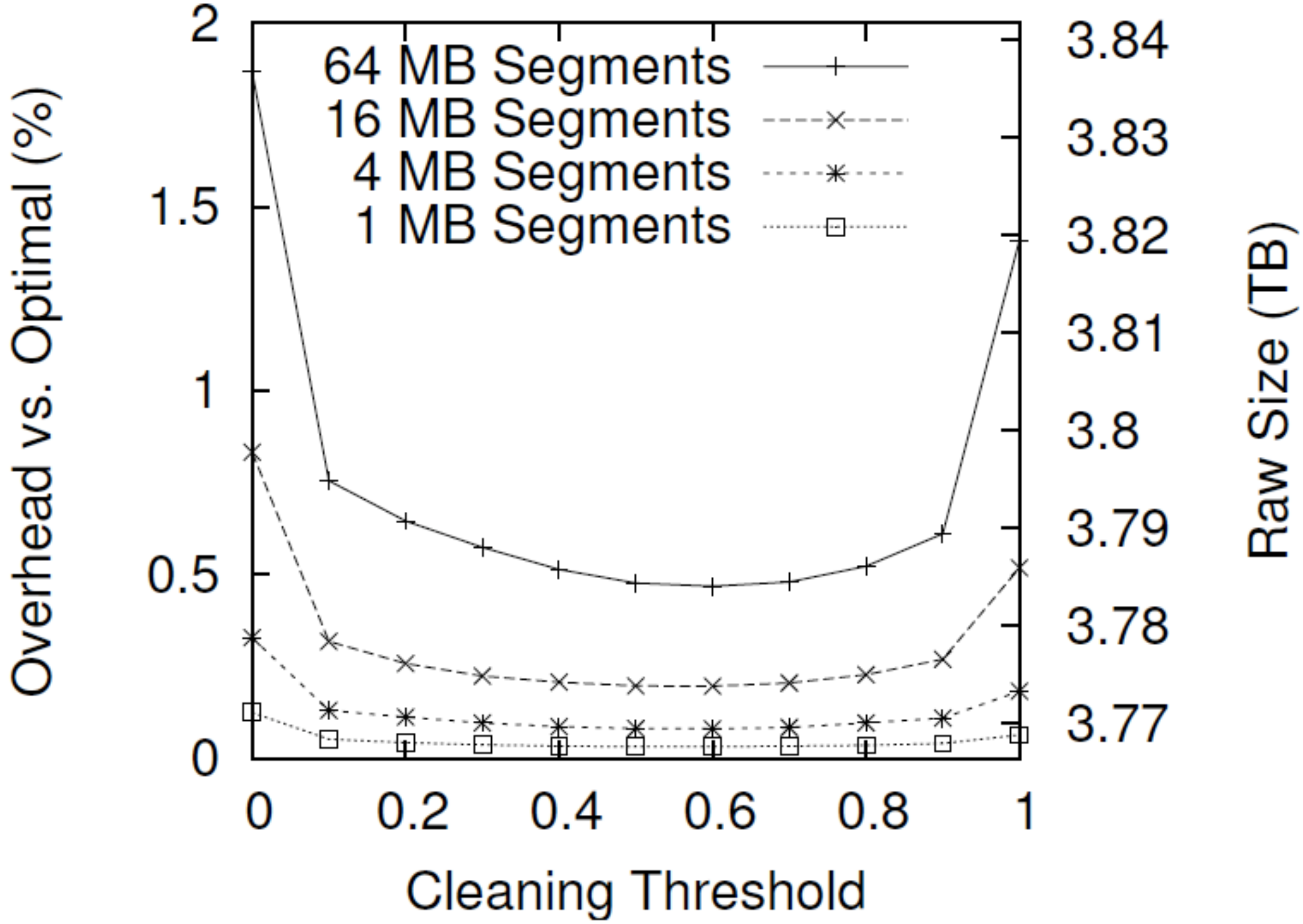
Backup over time (user trace)



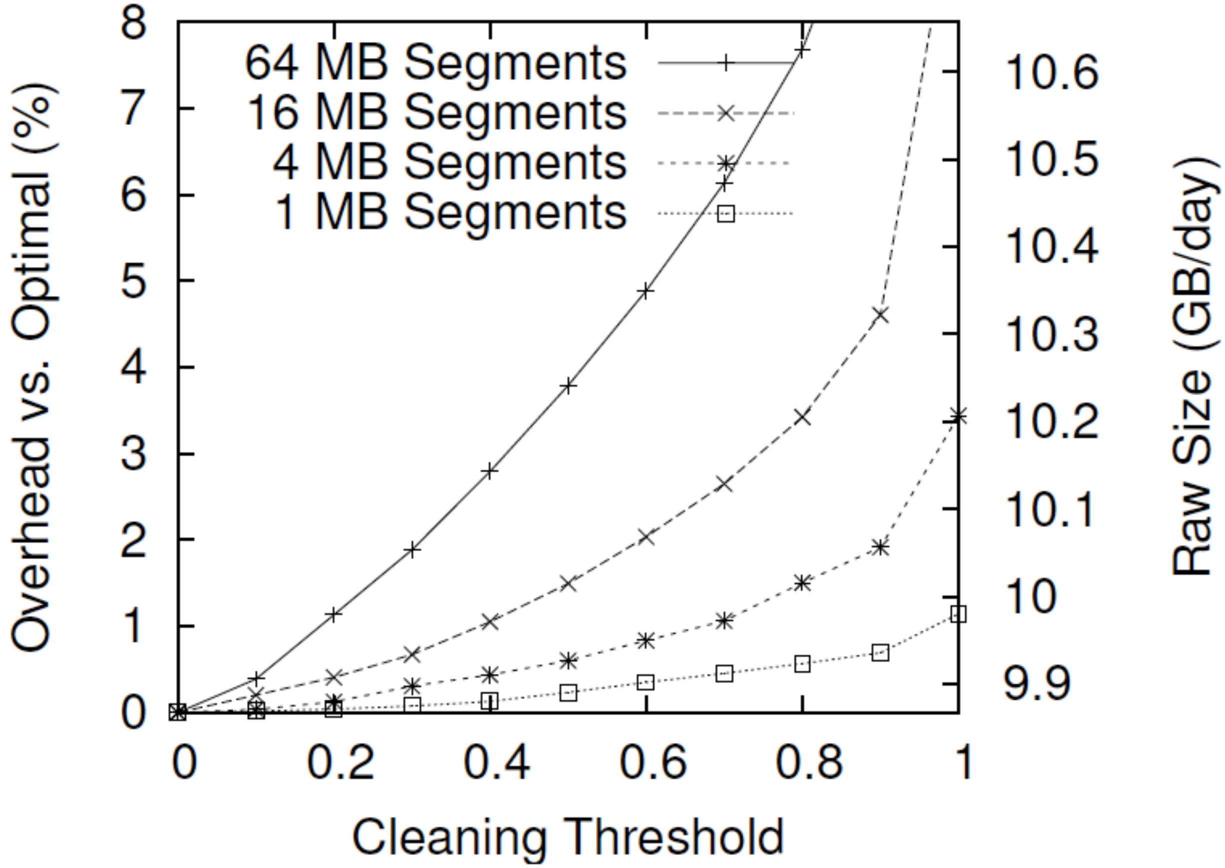
Backup w/out Segment Cleaning (user trace)



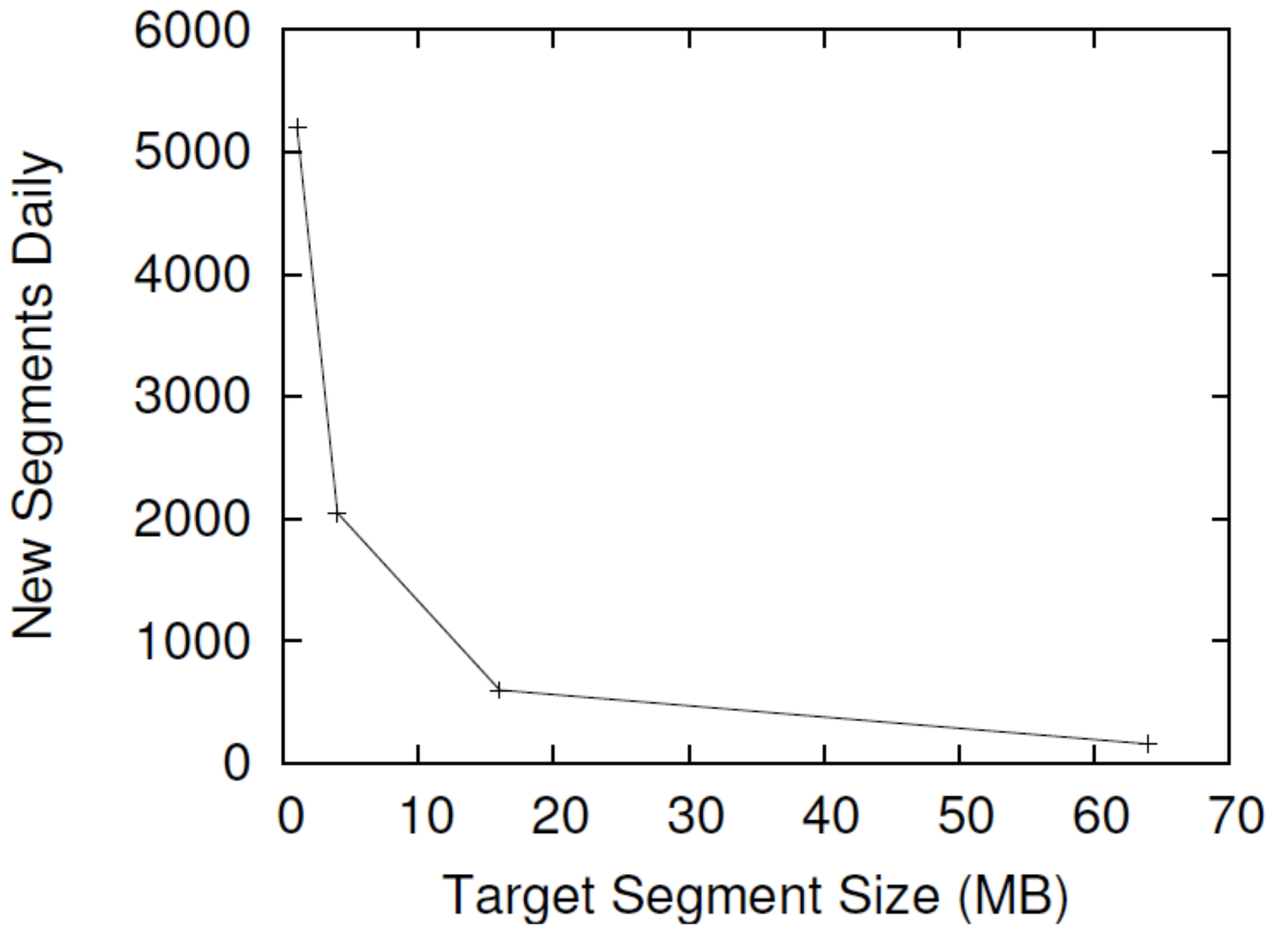
Average Daily Storage (fileserver)



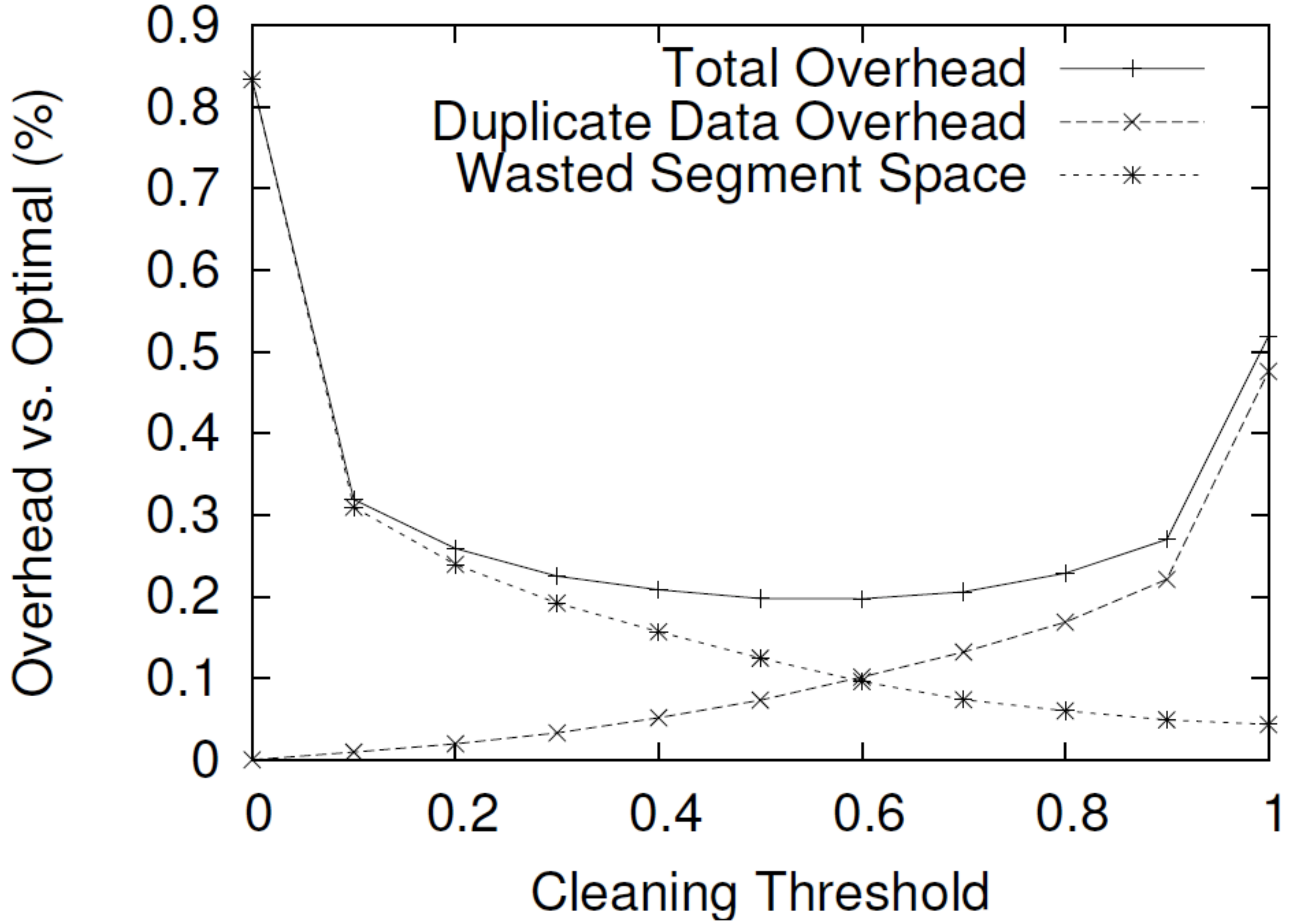
Average Daily Upload (fileserver)



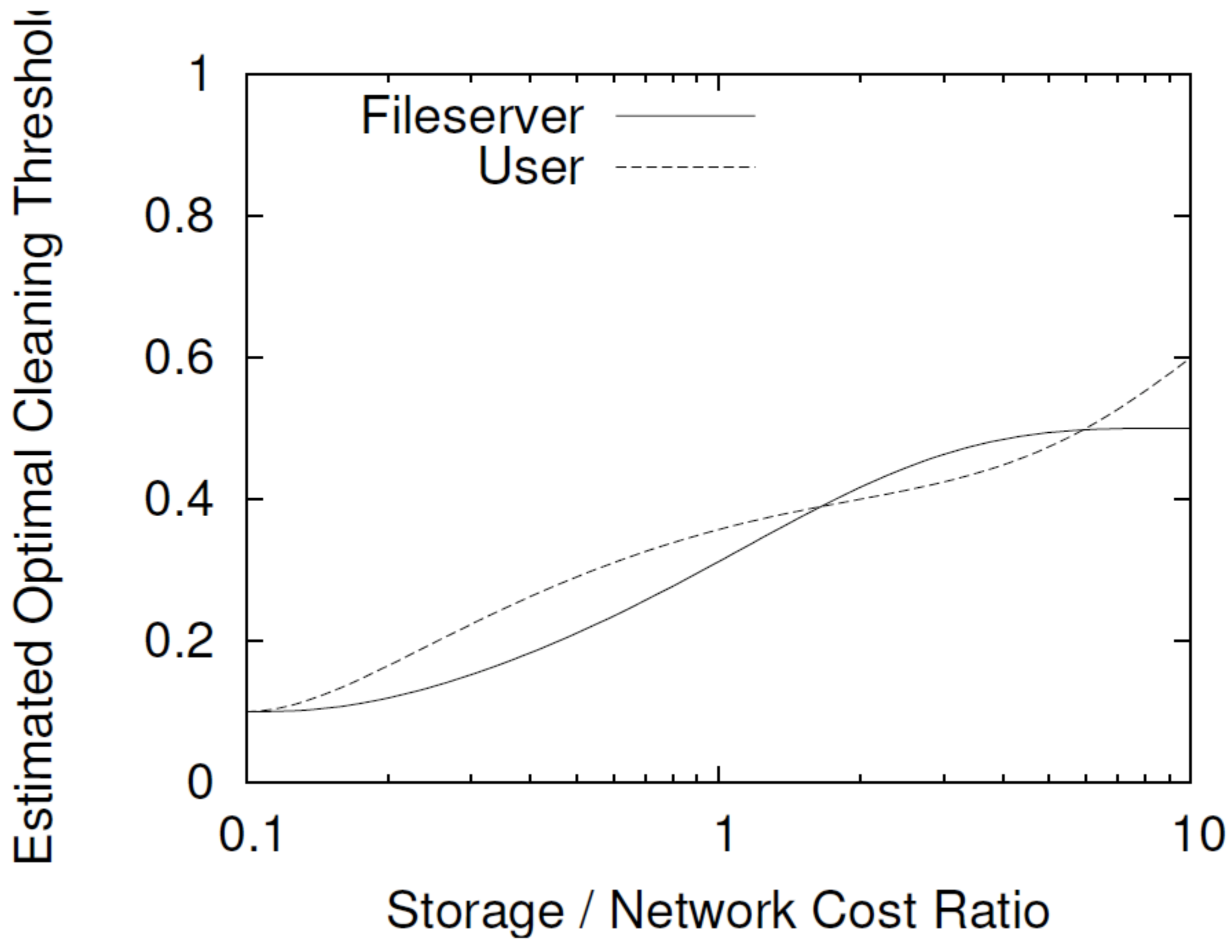
Average Segments per Day (fileserver)



Storage overhead for 16MB Segment (fileserver)



Optimal Cleaning Threshold



Overheads

	File A	File B
File size	4.860 MB	5.890 MB
Compressed size	1.547 MB	2.396 MB
Cumulus size	5.190 MB	3.081 MB
Size overhead	235%	29%
rdiff delta	1.421 MB	122 KB
Cumulus delta	1.527 MB	181 KB
Delta overhead	7%	48%

Monetary Case Study

- Storage: $0.15 per GB . Month
- Upload: $0.10 per GB
- Segment: $0.01 per 1000 files uploaded

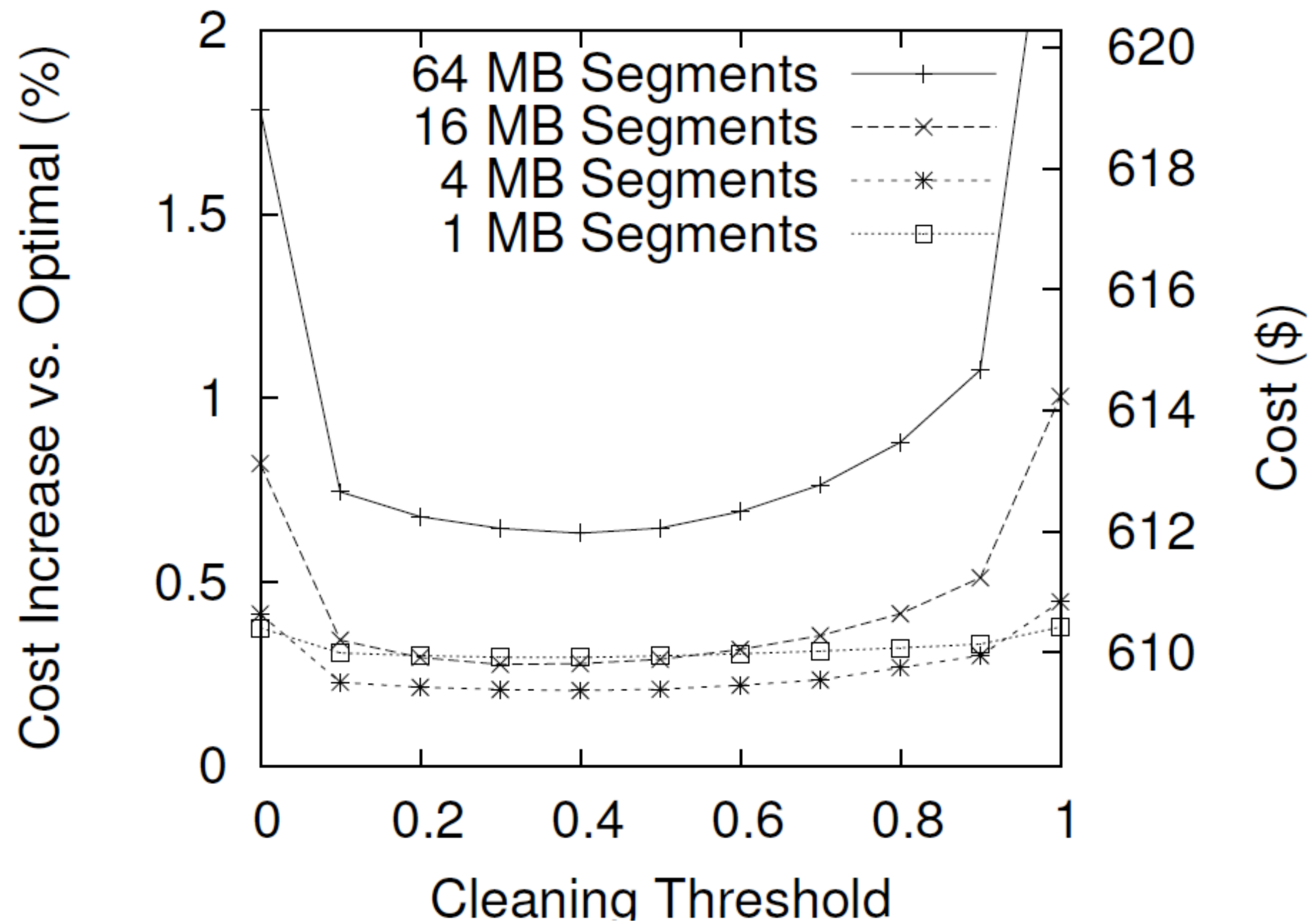
- *We are charged this amount, so please be careful with your labs and projects!!!*

Monetary Costs for Backup

Fileserver	Amount	Cost
Initial upload	3563 GB	$356.30
Upload	303 GB/month	$30.30/month
Storage	3858 GB	$578.70/month

User	Amount	Cost
Initial upload	1.82 GB	$0.27
Upload	1.11 GB/month	$0.11/month
Storage	2.68 GB	$0.40/month

Costs for Backup (fileserver)



Monetary Cost Comparison (user trace)

System	Storage	Upload	Operations
Jungle Disk	$\approx$ 2 GB	1.26 GB	30000
	$0.30	$0.126	$0.30
Brackup	1.340 GB	0.760 GB	9027
(default)	$0.201	$0.076	$0.090
Brackup	1.353 GB	0.713 GB	1403
(aggregated)	$0.203	$0.071	$0.014
Cumulus	1.264 GB	0.465 GB	419
	$0.190	$0.047	$0.004

Outline

- Motivation/Intro
- Related Work
- Design
- Evaluation
- Thoughts and Conclusions

Discussion

- Thoughts?
- Did paper make case for Thin Clouds?
- Sharing between clients ignored?
- What every happened to P2P?!

Lab 0

- http://s3.amazonaws.com/edu-cornell-cs-cs5300/aws-get-started.html

Next Time

- Read *NFS* and write review:
  - *Design and Implementation of a Network File System*, Sandberg, Goldberg, Kleiman, Walsh, and Lyon, USENIX 1985

- Do Lab 0

- Check website for updated schedule