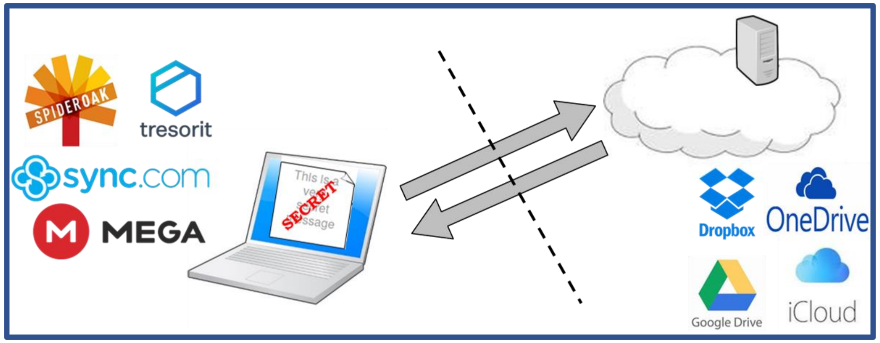
Delta Encoding Overhead Analysis of Cloud Storage Systems using Client-side Encryption

Eric Henziger, *Linköping University*

Niklas Carlsson, *Linköping University*

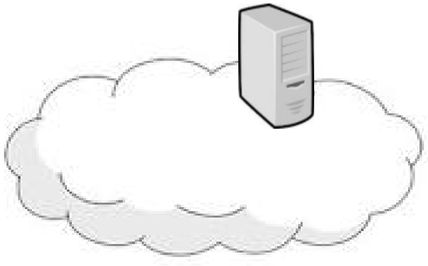
LINKÖPING UNIVERSITY



Proc. IEEE CloudCom, Sydney, Australia, Dec. 2019

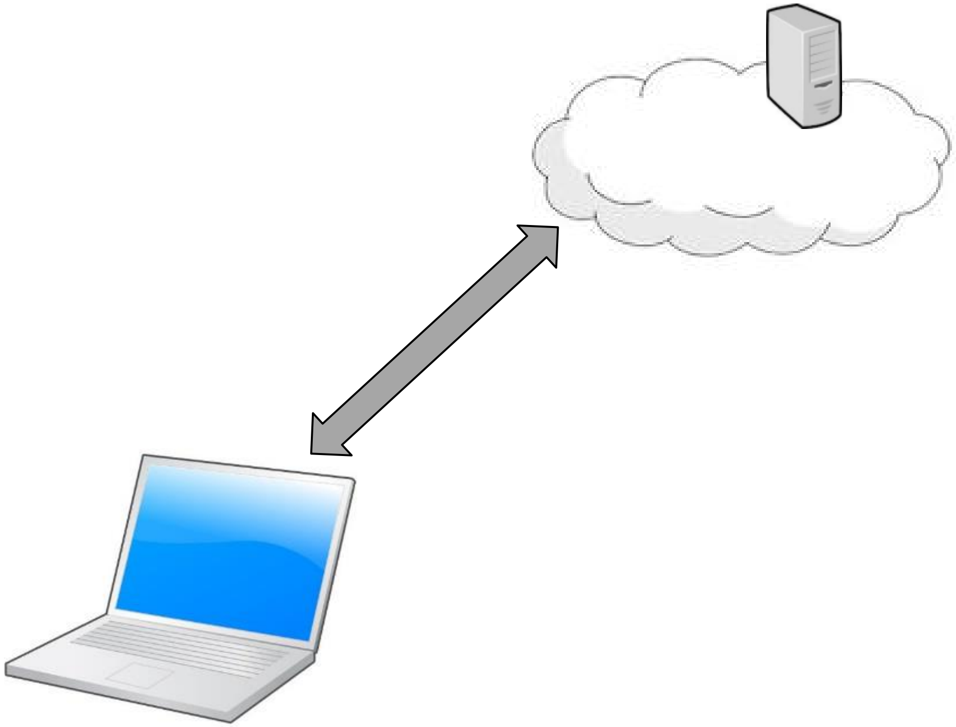
Motivation and problem

- Popular services: Some with 100s of millions of active users each month



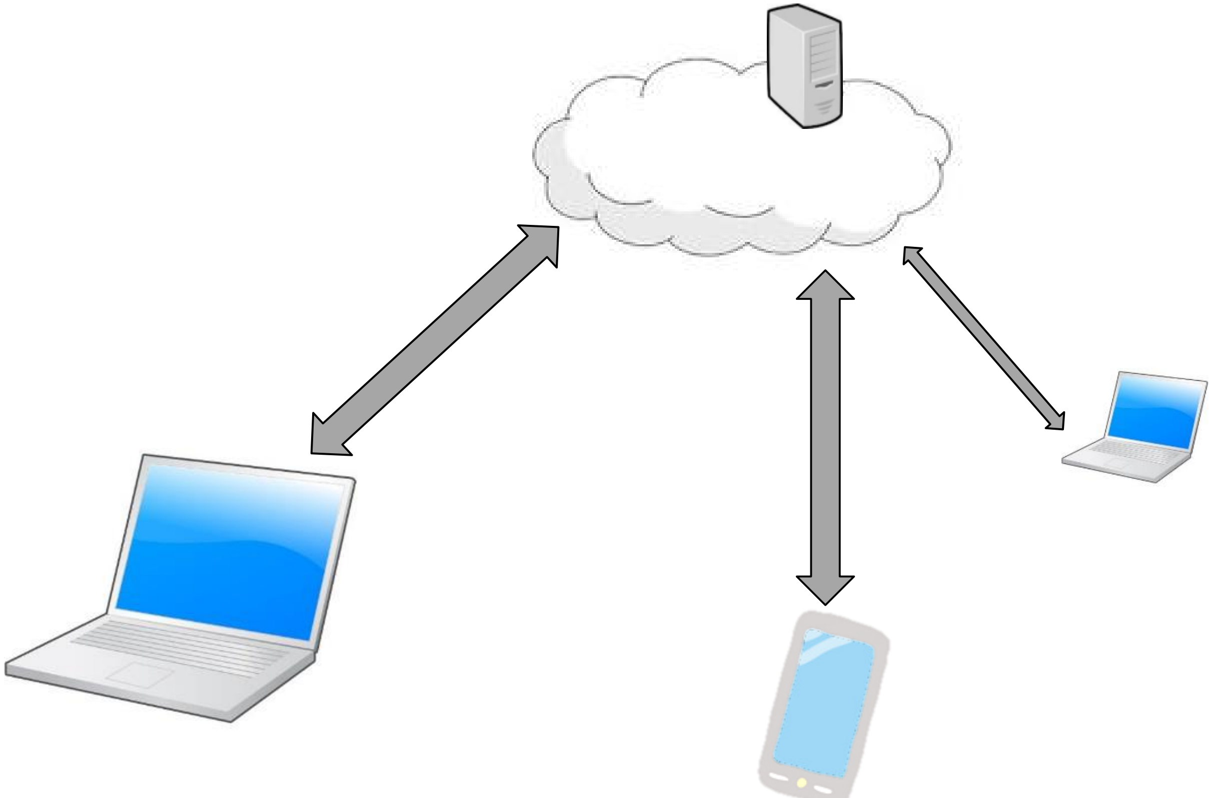
Motivation and problem

- Popular services: Some with 100s of millions of active users each month
- Cloud services have changed how users store and access data



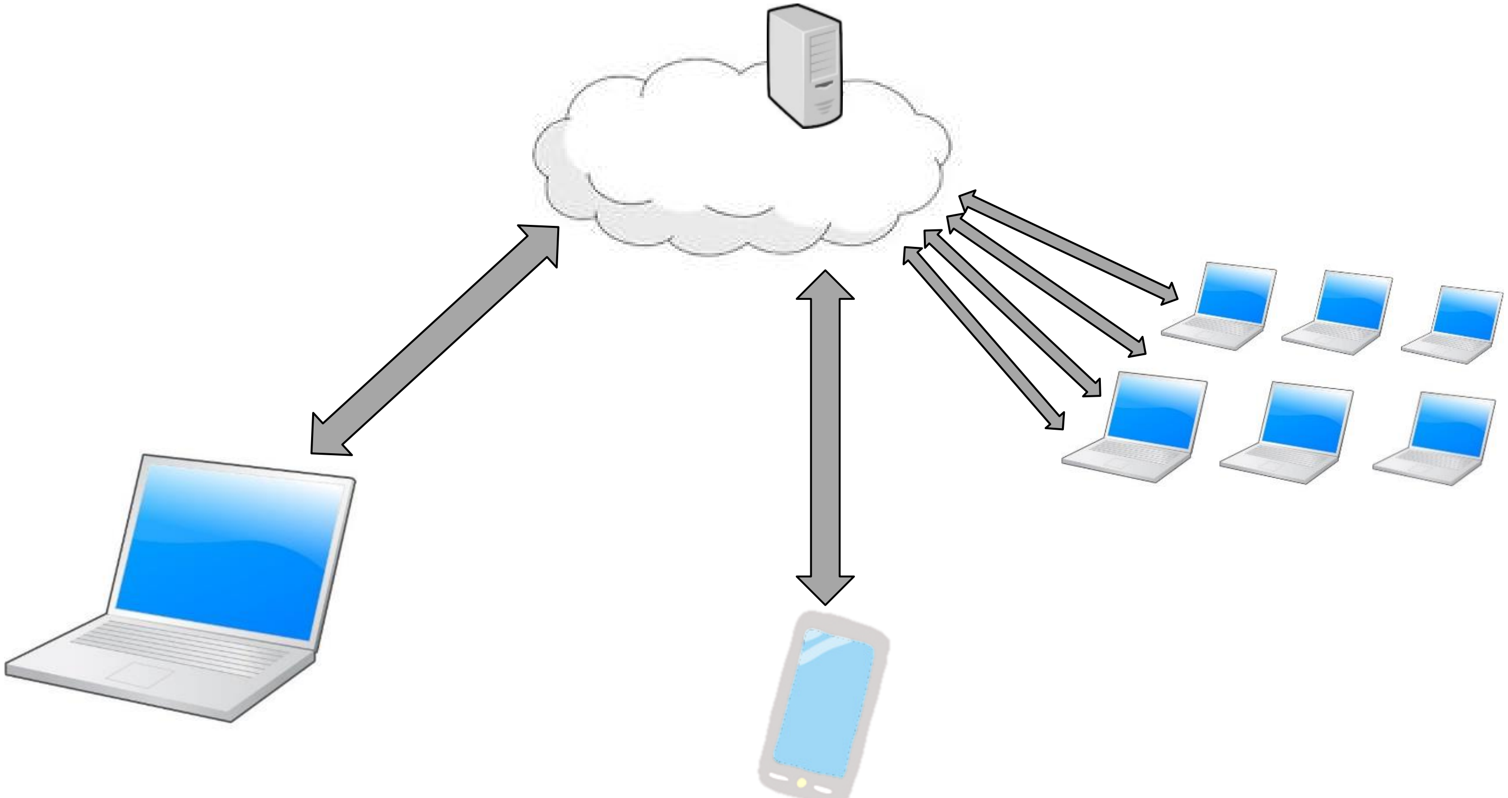
Motivation and problem

- Popular services: Some with 100s of millions of active users each month
- Cloud services have changed how users store and access data
  - E.g., often transparently across multiple devices



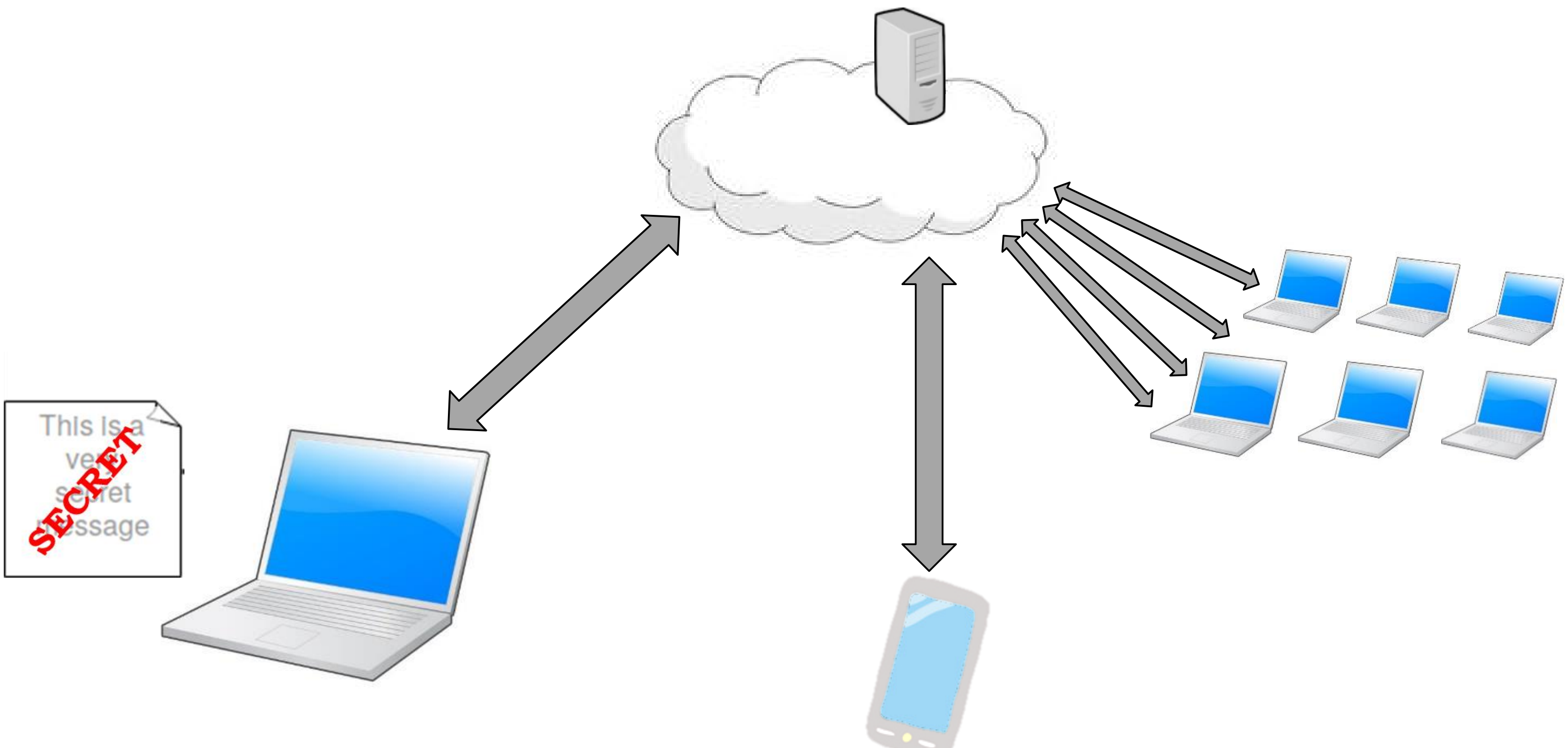
Motivation and problem

- Popular services: Some with 100s of millions of active users each month
- Cloud services have changed how users store and access data
  - E.g., often transparently across multiple devices or users



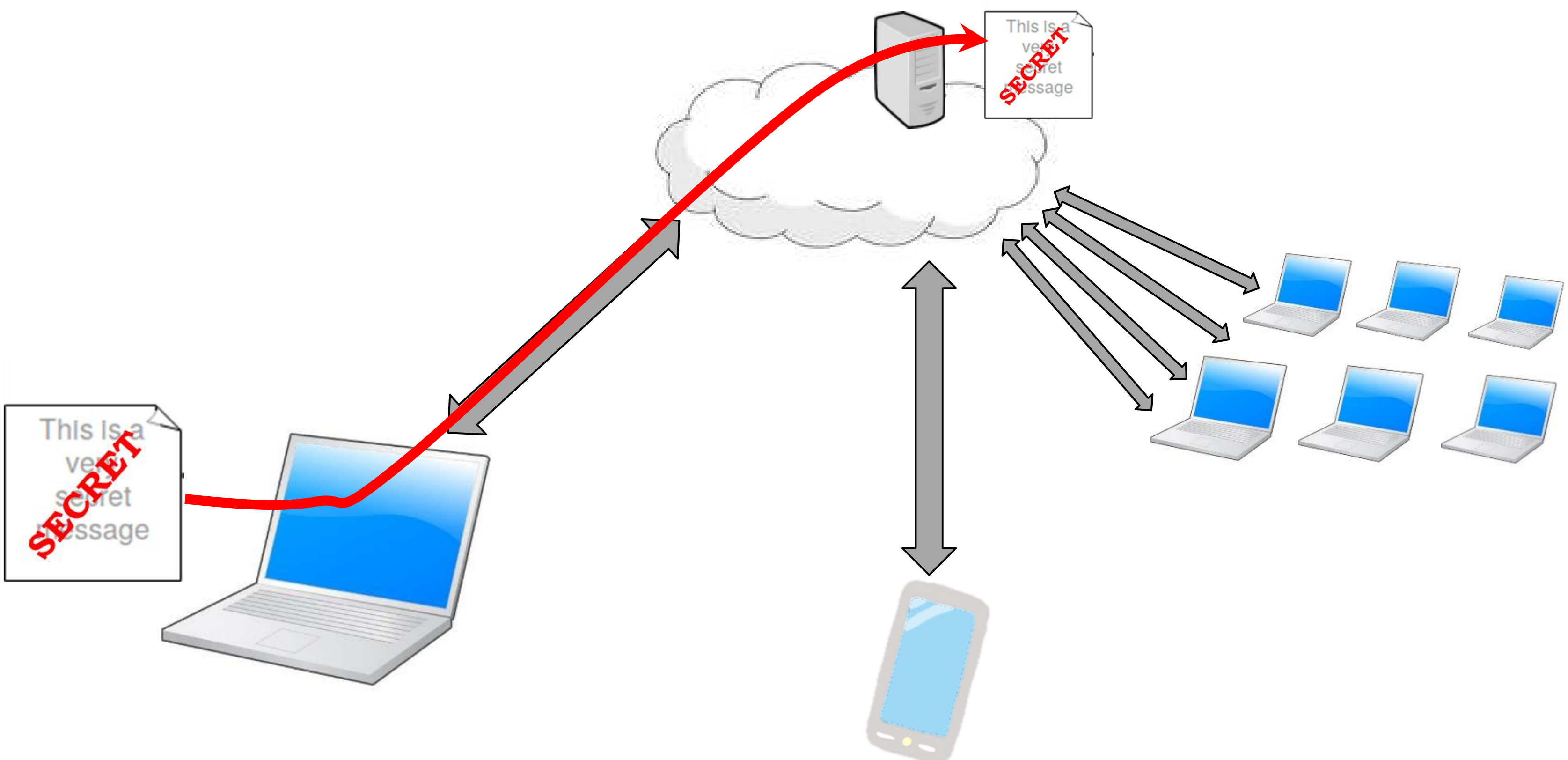
Motivation and problem

- Popular services: Some with 100s of millions of active users each month
- Cloud services have changed how users store and access data
  - E.g., often transparently across multiple devices or users
- Most services require that users fully trust the provider
  - Services gets access to all data and information



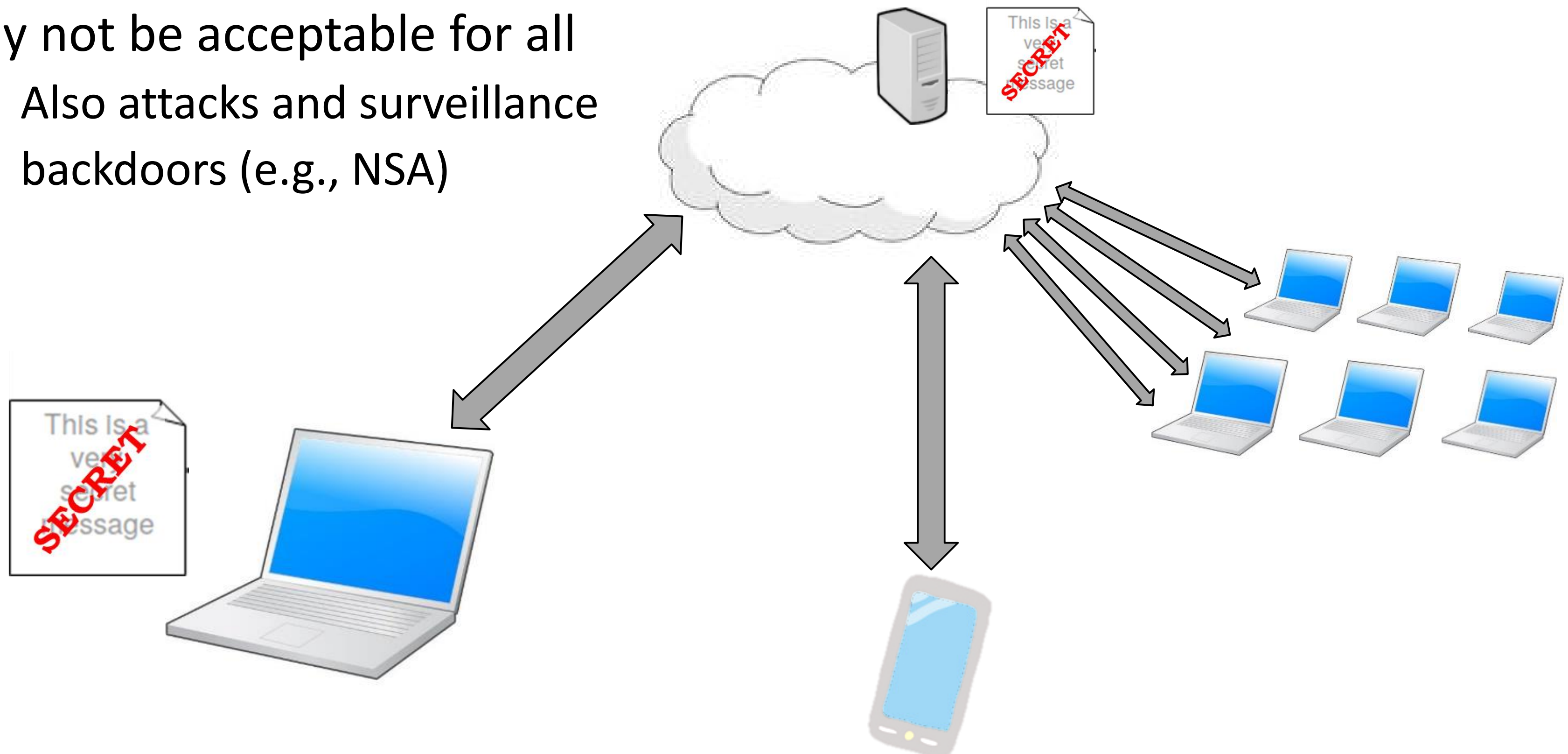
Motivation and problem

- Popular services: Some with 100s of millions of active users each month
- Cloud services have changed how users store and access data
  - E.g., often transparently across multiple devices or users
- Most services require that users fully trust the provider
  - Services gets access to all data and information



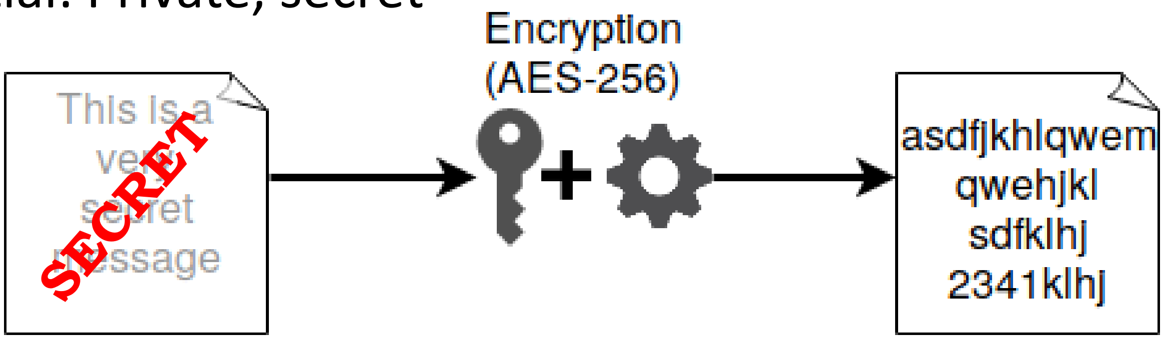
Motivation and problem

- Popular services: Some with 100s of millions of active users each month
- Cloud services have changed how users store and access data
  - E.g., often transparently across multiple devices or users
- Most services require that users fully trust the provider
  - Services gets access to all data and information
- May not be acceptable for all
  - Also attacks and surveillance backdoors (e.g., NSA)



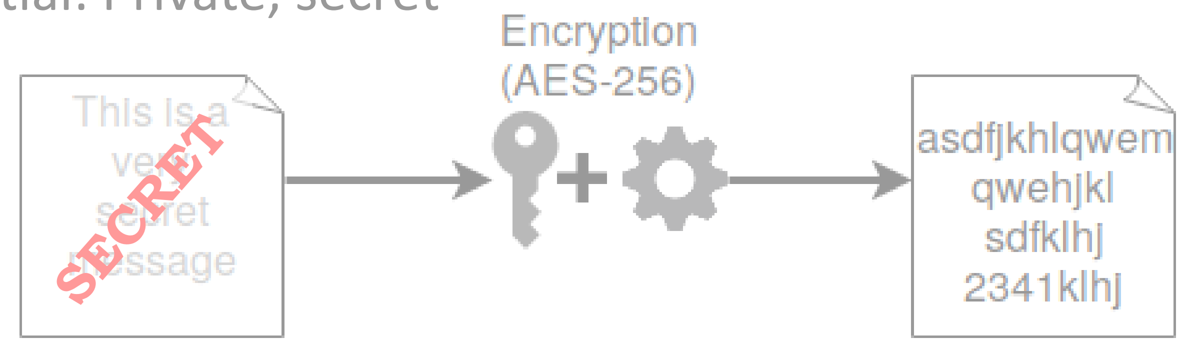
Client-side encryption (CSE)

- Confidential: Private, secret

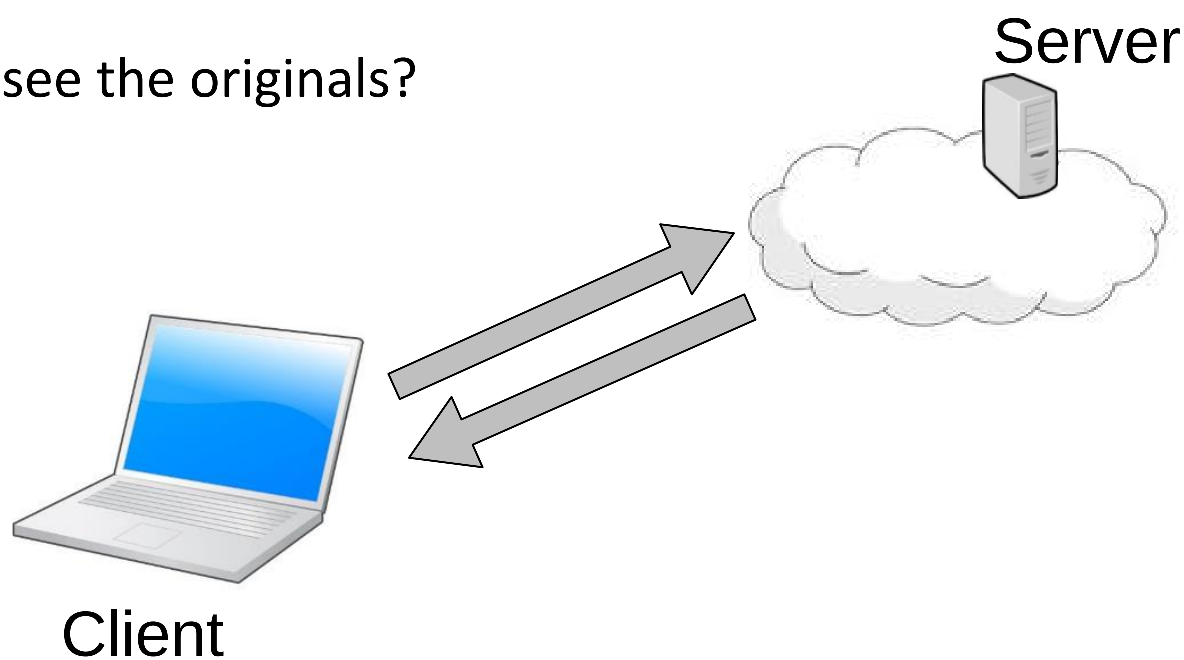


Client-side encryption (CSE)

- Confidential: Private, secret



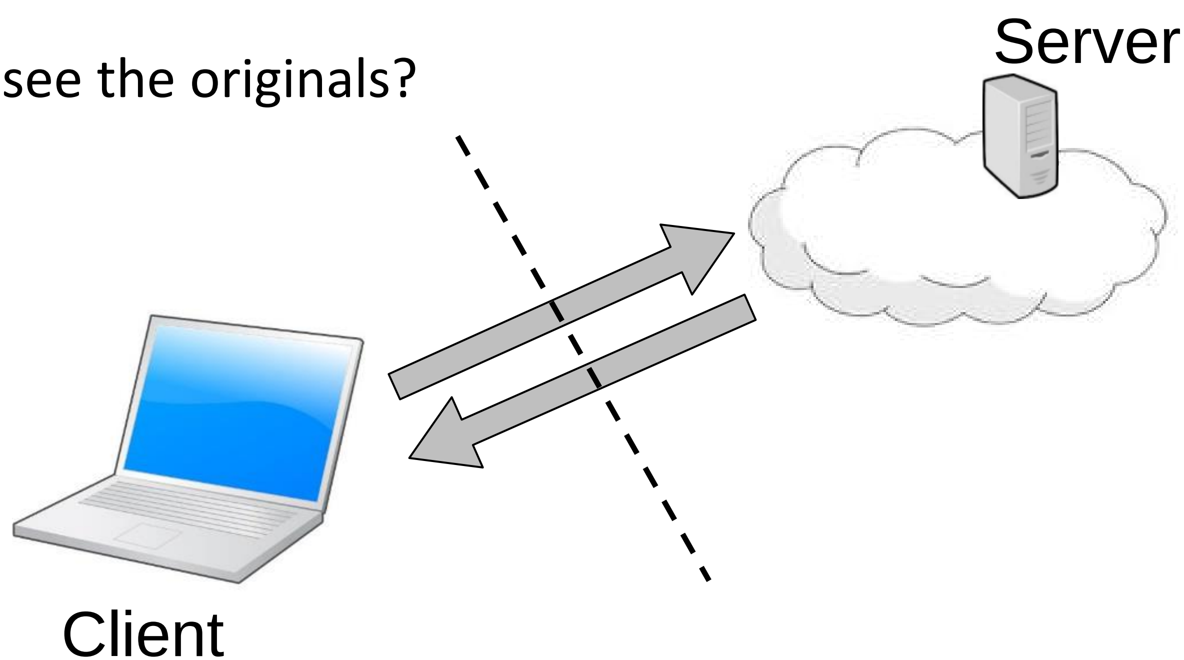
- Who can see the originals?



Client-side encryption (CSE)

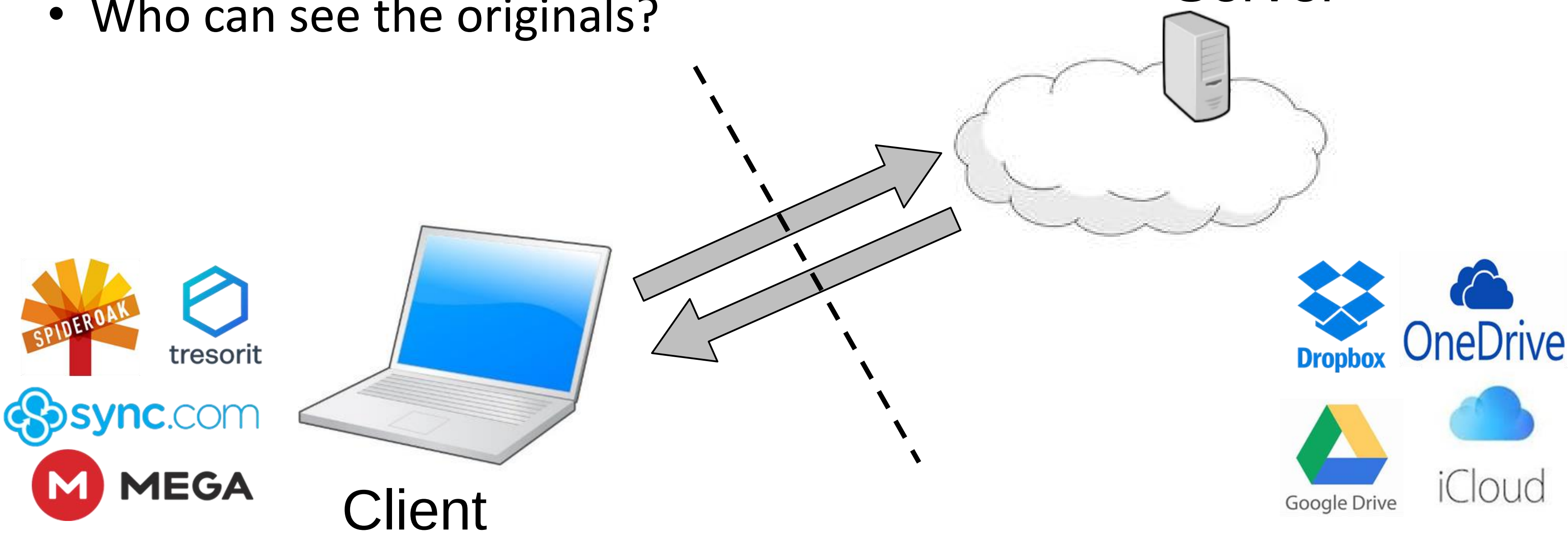
- Confidential: Private, secret

- Who can see the originals?



Client-side encryption (CSE)

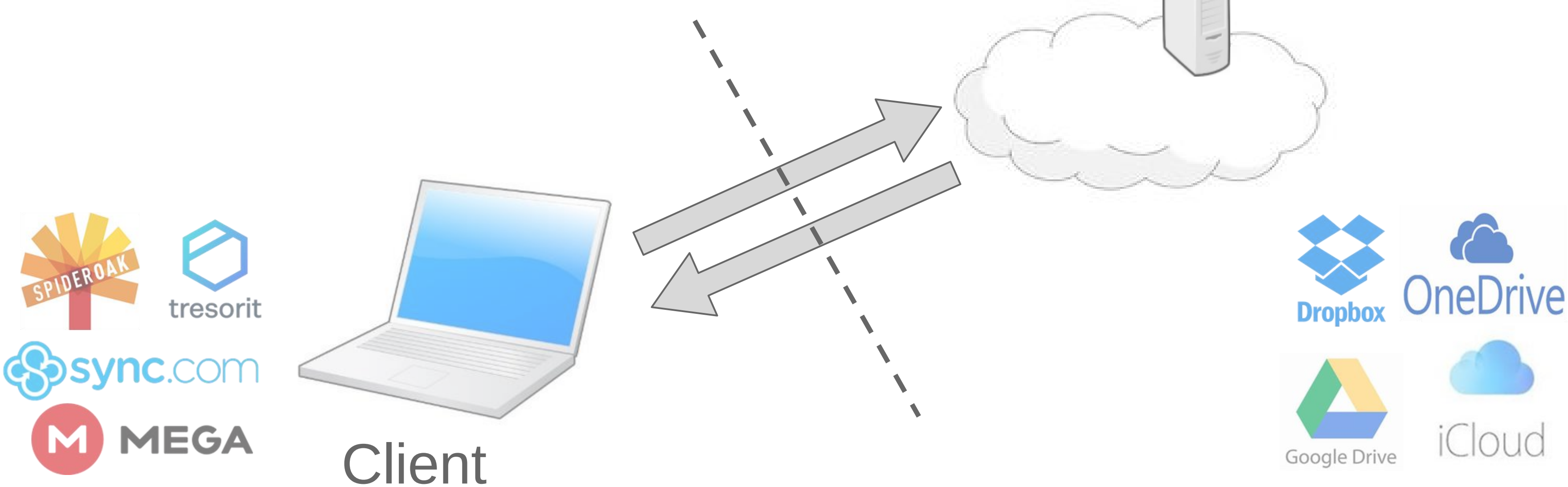
- Who can see the originals?



Client-side encryption (CSE)

However, CSE complicates some bandwidth saving features such as deduplication and delta encoding …

- Who can see the originals?



E. Henziger and N. Carlsson, "The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems", Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

Feature summary

	Services	Compression	Deduplication	Delta Sync
		Feature/capability		
non-CSE	Dropbox	**Yes**	**Yes**	**Yes**
non-CSE	iCloud	No	**Yes**	**Yes**
non-CSE	Google Drive	**Conditional**	No	No
non-CSE	OneDrive	No	**Sometimes**	No
CSE	Mega	No	**Yes**	No
CSE	Sync.com	No	**Yes**	No
CSE	SpiderOak	**Yes**	**Yes**	**Yes**
CSE	Tresorit	**Yes**	No	No

- No clear difference between CSE vs non-CSEs
- Instead, large variations within each group

E. Henziger and N. Carlsson, "The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems", Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

Feature summary

		Feature/capability		
	Services	Compression	Deduplication	Delta Sync
non-CSE	Dropbox	**Yes**	**Yes**	**Yes**
	iCloud	No	**Yes**	**Yes**
	Google Drive	**Conditional**	No	No
	OneDrive	No	**Sometimes**	No
CSE	Mega	No	**Yes**	No
	Sync.com	No	**Yes**	No
	SpiderOak	**Yes**	**Yes**	**Yes**
	Tresorit	**Yes**	No	No

- No clear difference between CSE vs non-CSEs
- Instead, large variations within each group

E. Henziger and N. Carlsson, “The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems”, Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

Feature summary

		Feature/capability		
	Services	Compression	Deduplication	Delta Sync
non-CSE	Dropbox	**Yes**	**Yes**	**Yes**
	iCloud	No	**Yes**	**Yes**
	Google Drive	**Conditional**	No	No
	OneDrive	No	**Sometimes**	No
CSE	Mega	No	**Yes**	No
	Sync.com	No	**Yes**	No
	SpiderOak	**Yes**	**Yes**	**Yes**
	Tresorit	**Yes**	No	No

- No clear difference between CSE vs non-CSEs
- Instead, large variations within each group

E. Henziger and N. Carlsson, "The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems", Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

Feature summary

		Feature/capability		
	Services	Compression	Deduplication	Delta Sync
non-CSE	Dropbox	**Yes**	**Yes**	**Yes**
	iCloud	No	**Yes**	**Yes**
	Google Drive	**Conditional**	No	No
	OneDrive	No	**Sometimes**	No
CSE	Mega	No	**Yes**	No
	Sync.com	No	**Yes**	No
	SpiderOak	**Yes**	**Yes**	**Yes**
	Tresorit	**Yes**	No	No

- No clear difference between CSE vs non-CSEs
- Instead, large variations within each group

E. Henziger and N. Carlsson, "The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems", Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

Feature summary

		Feature/capability		
	Services	Compression	Deduplication	Delta Sync
non-CSE	Dropbox	**Yes**	**Yes**	**Yes**
	iCloud	No	**Yes**	**Yes**
	Google Drive	**Conditional**	No	No
	OneDrive	No	**Sometimes**	No
CSE	Mega	No	**Yes**	No
	Sync.com	No	**Yes**	No
	SpiderOak	**Yes**	**Yes**	**Yes**
	Tresorit	**Yes**	No	No

- No clear difference between CSE vs non-CSEs
- Instead, large variations within each group
- Only Dropbox (non-CSE) and SpiderOak (CSE) has all three features

E. Henziger and N. Carlsson, "The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems", Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

Feature summary

		Feature/capability		
	Services	Compression	Deduplication	Delta Sync
non-CSE	Dropbox	Yes	Yes	Yes
non-CSE	iCloud	No	Yes	Yes
non-CSE	Google Drive	Conditional	No	No
non-CSE	OneDrive	No	Sometimes	No
CSE	Mega	No	Yes	No
CSE	Sync.com	No	Yes	No
CSE	SpiderOak	Yes	Yes	Yes
CSE	Tresorit	Yes	No	No

- No clear difference between CSE vs non-CSEs
- Instead, large variations within each group
- Only Dropbox (non-CSE) and SpiderOak (CSE) has all three features
- All services implement at least some feature (but different)

E. Henziger and N. Carlsson, “The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems”, Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

Feature summary

		Feature/capability		
	Services	Compression	Deduplication	Delta Sync
non-CSE	Dropbox	**Yes**	**Yes**	**Yes**
	iCloud	No	**Yes**	**Yes**
	Google Drive	**Conditional**	No	No
	OneDrive	No	**Sometimes**	No
CSE	Mega	No	**Yes**	No
	Sync.com	No	**Yes**	No
	SpiderOak	**Yes**	**Yes**	**Yes**
	Tresorit	**Yes**	No	No

- No clear difference between CSE vs non-CSEs
- Instead, large variations within each group
- Only Dropbox (non-CSE) and SpiderOak (CSE) has all three features
- All services implement at least some feature (but different)
- Furthermore: Delta encoding efficiency differ substantially …

E. Henziger and N. Carlsson, "The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems", Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

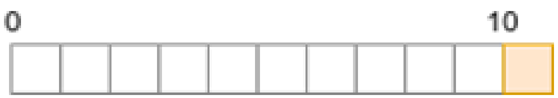
Feature 3: Delta encoding

Test method

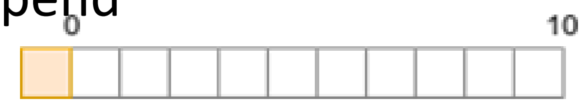
- Make sequence of changes
- Measure size of updates (full vs part)

File modifications considered

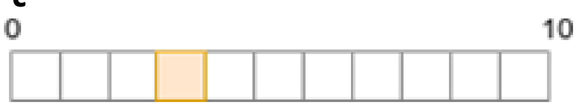
- Append



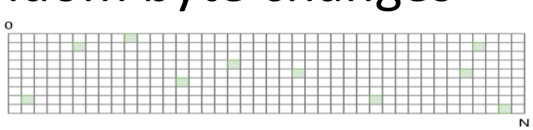
- Prepend



- Insert



- N random byte changes



E. Henziger and N. Carlsson, "The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems", Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

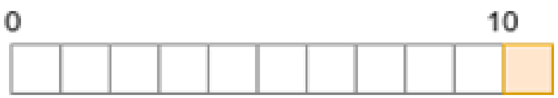
Feature 3: Delta encoding

Test method

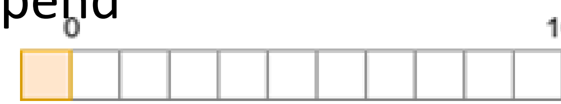
- Make sequence of changes
- Measure size of updates (full vs part)

File modifications considered

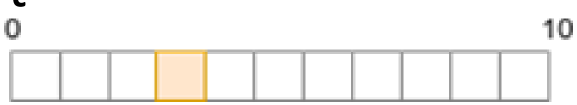
- Append



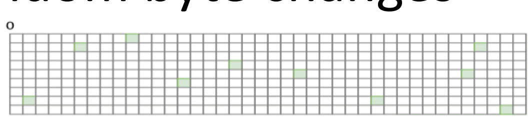
- Prepend



- Insert



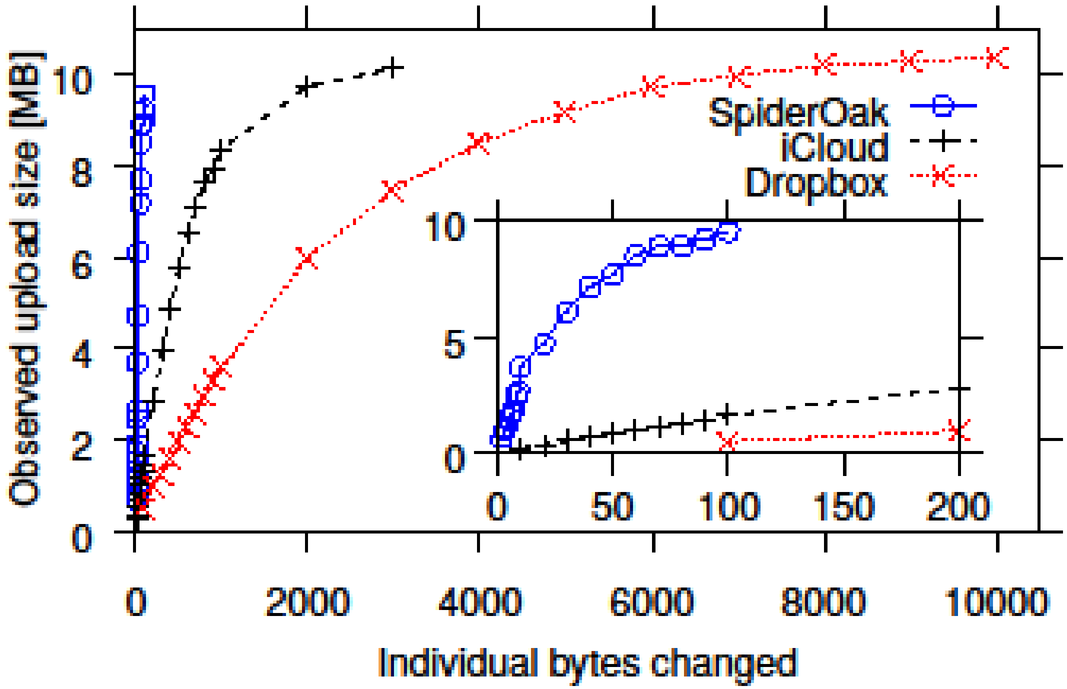
- N random byte changes

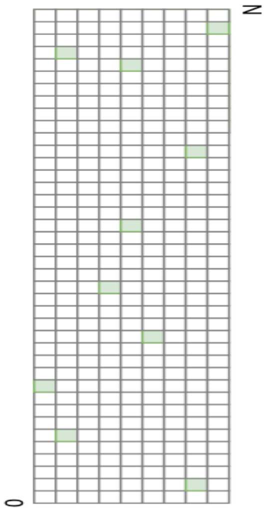


	Yes	No
Non-CSE	Dropbox, iCloud	Google Drive, OneDrive
CSE	SPIDEROAK	sync.com, MEGA, tresorit

E. Henziger and N. Carlsson, "The Overhead of Confidentiality and Client-side Encryption in Cloud Storage Systems", Proc. IEEE/ACM UCC, Auckland, New Zealand, Dec. 2019.

Delta encoding efficiency …





Large differences among service implementing (some) delta encoding

- SpiderOak (CSE) performs much worse than iCloud (non-CSE) and Dropbox (non-CSE)

Important to understand these differences and how much the CSE performance can be improved …

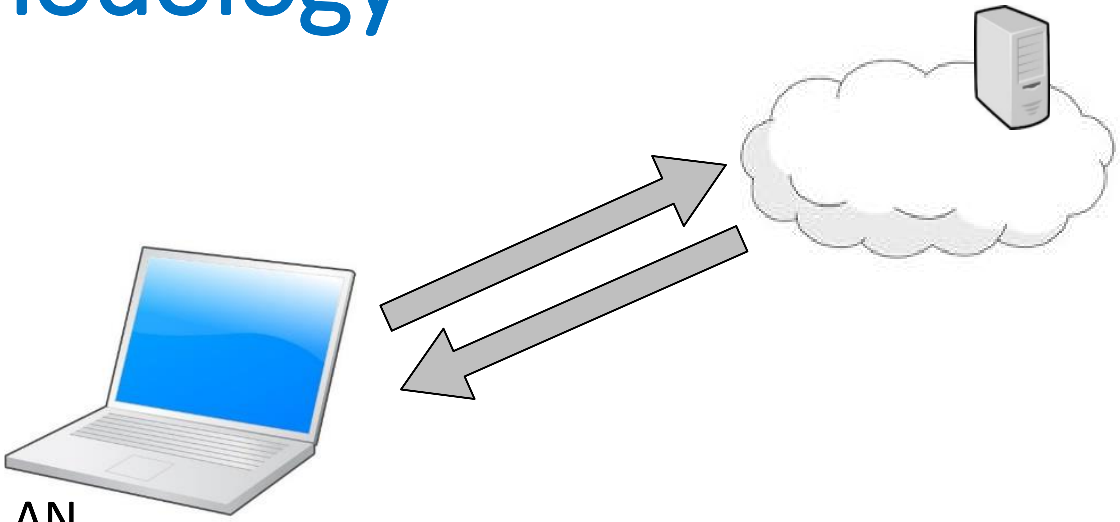
Contributions (at a glance)

Targeted experiments and a model-based analysis to

1. demonstrate the delta encoding problem associated with CSE
2. characterize the practical overheads associated with delta encoding
3. determine the potential room for further improvements.

Results demonstrate significant cost saving opportunities not yet used by current CSEs

Baseline methodology



macOS
campus LAN

E. Bocchi, I. Drago, and M. Mellia, "Personal Cloud Storage Benchmarks and Comparison," IEEE Transactions on Cloud Computing, vol. 5, no. 4, pp. 751–764, 2017.

Baseline methodology

1. Start cloud storage application

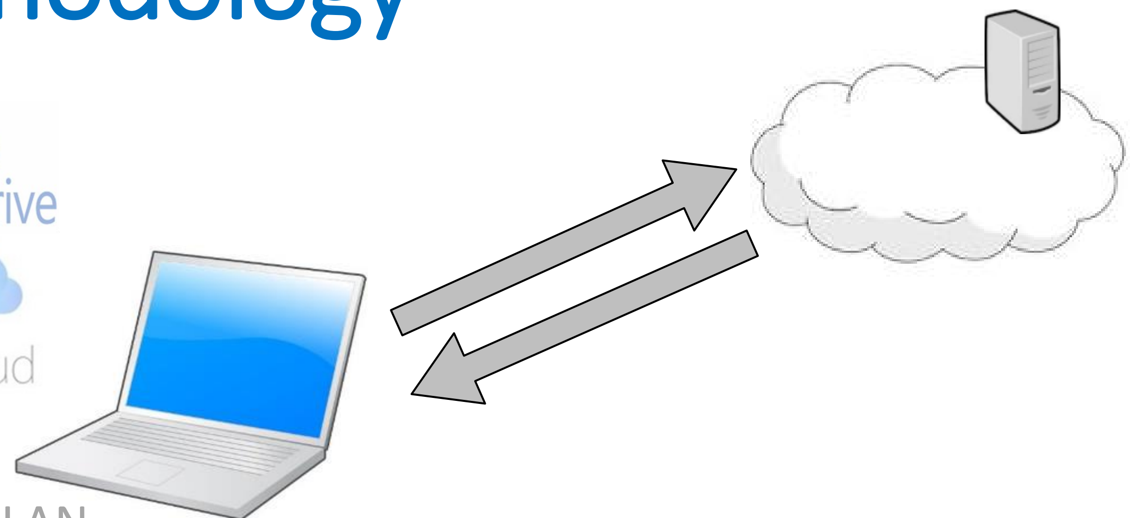
E. Bocchi, I. Drago, and M. Mellia, "Personal Cloud Storage Benchmarks and Comparison," IEEE Transactions on Cloud Computing, vol. 5, no. 4, pp. 751–764, 2017.

Baseline methodology

SpiderOak, tresorit, sync.com, MEGA

vs

Dropbox, OneDrive, Google Drive, iCloud



macOS
campus LAN

python™

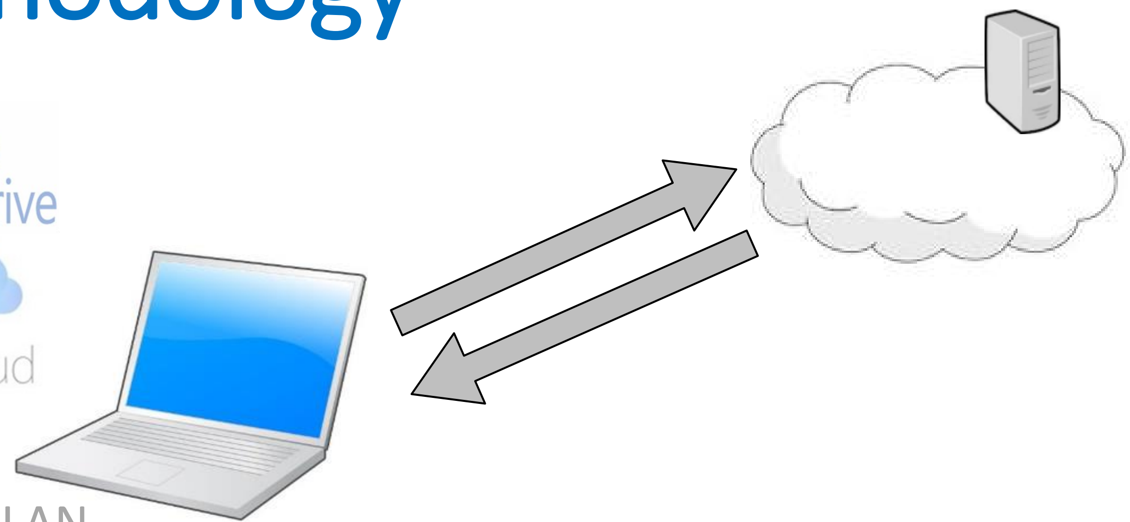
- netifaces
- pcapy
- psutil
- numpy
- scipy

1. Start cloud storage application
2. Capture network traffic
3. Measure CPU, memory, disk utilization

E. Bocchi, I. Drago, and M. Mellia, "Personal Cloud Storage Benchmarks and Comparison," IEEE Transactions on Cloud Computing, vol. 5, no. 4, pp. 751–764, 2017.

Baseline methodology

vs



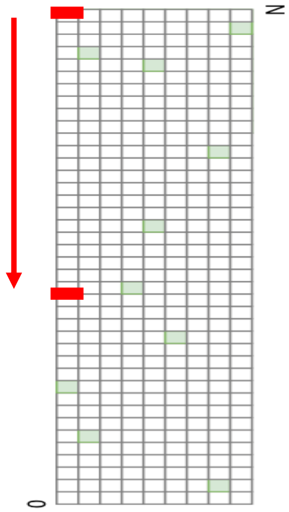
macOS
campus LAN

- netifaces
- pcapy
- psutil
- numpy
- scipy

1. Start cloud storage application
2. Capture network traffic
3. Measure CPU, memory, disk utilization
4. Place file in sync folder
5. Wait for synchronization to finish
6. Process capture files and measurements

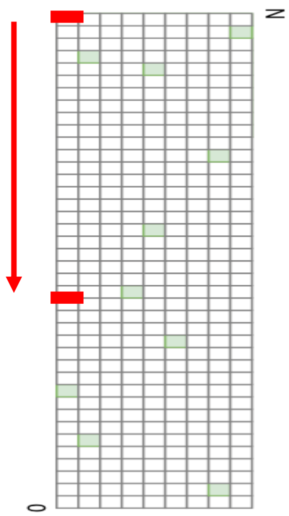
E. Bocchi, I. Drago, and M. Mellia, "Personal Cloud Storage Benchmarks and Comparison," IEEE Transactions on Cloud Computing, vol. 5, no. 4, pp. 751–764, 2017.

SpiderOak: Bytes uploaded



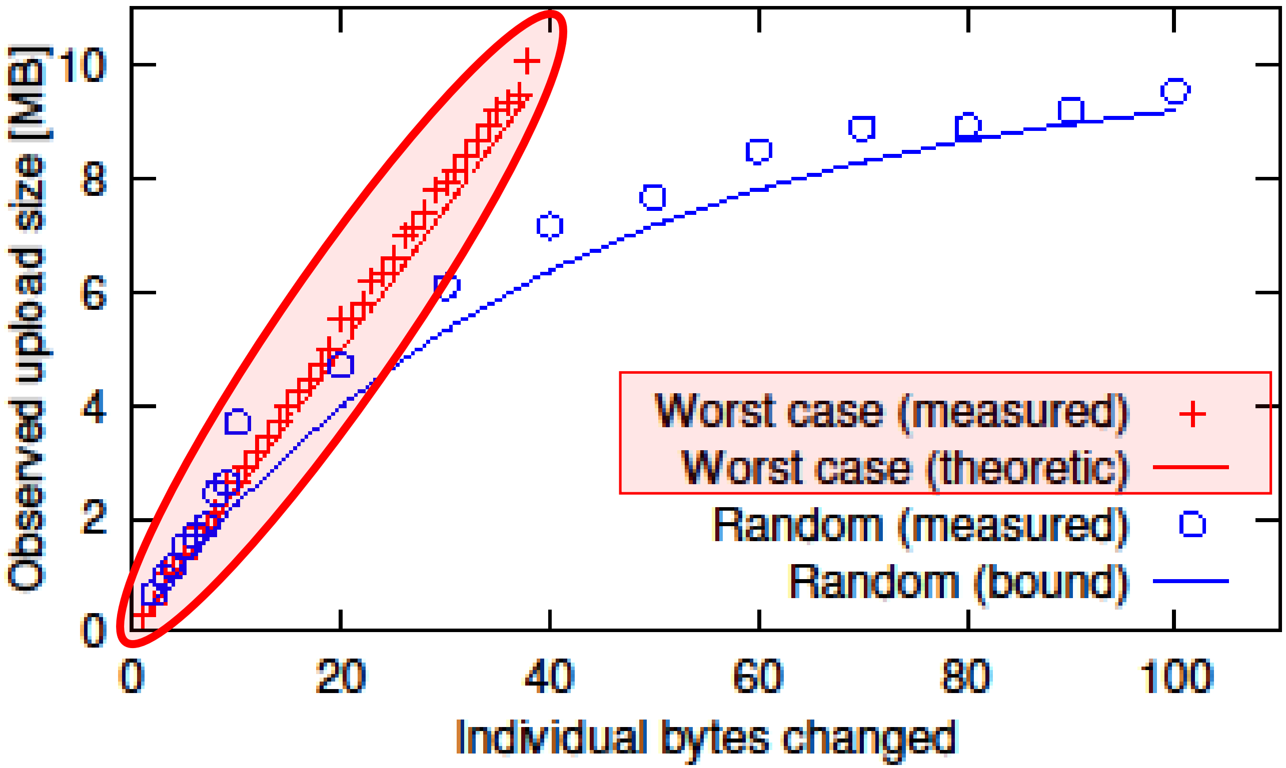
- Block-based encoding (size: 256kB)

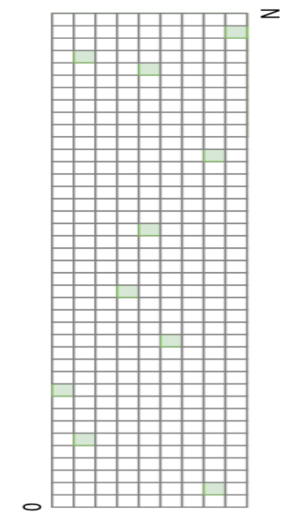
SpiderOak: Bytes uploaded



- Block-based encoding (size: 256kB)
- Worst-case overhead: change one byte in each 256kB block

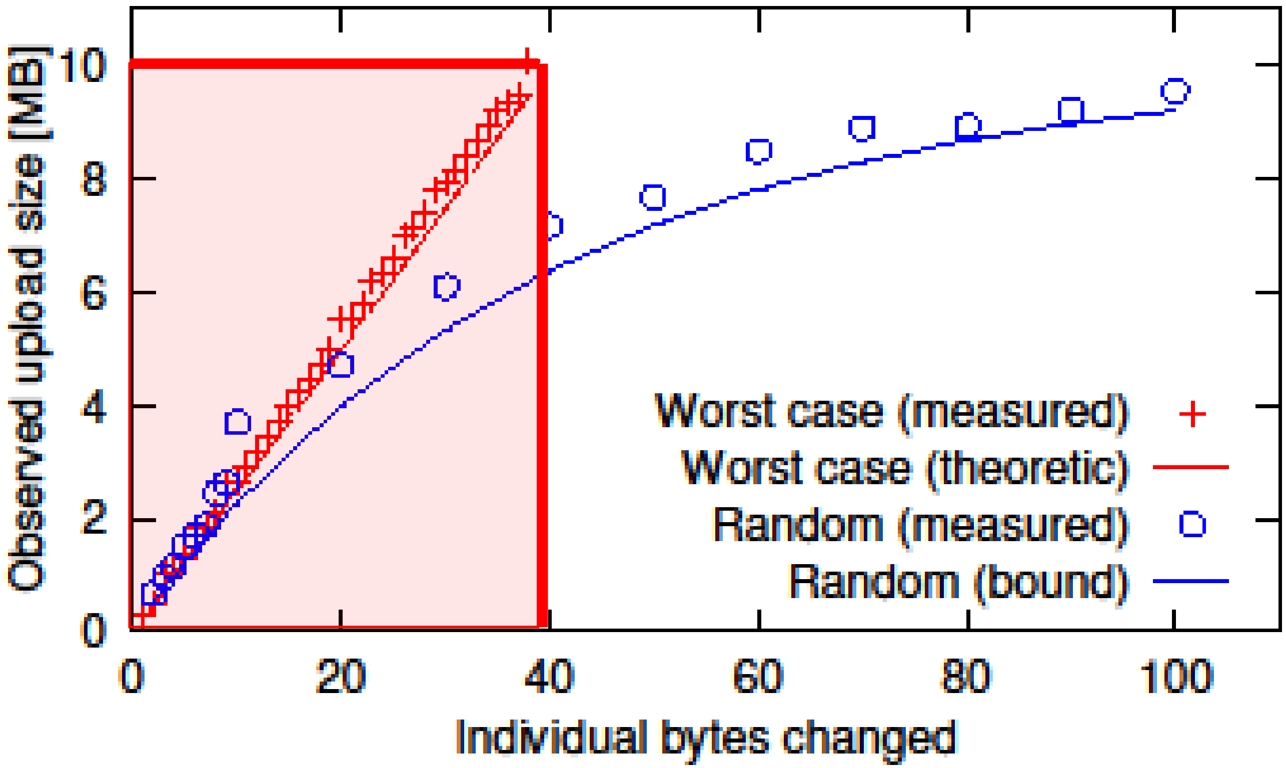
SpiderOak: Bytes uploaded

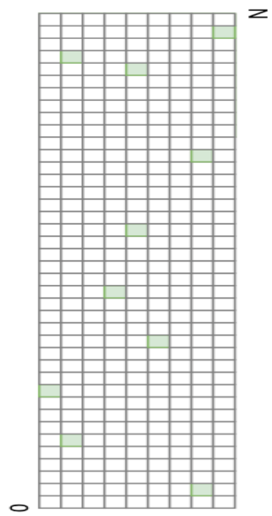




- Block-based encoding (size: 256kB)
- Worst-case overhead: change one byte in each 256kB block

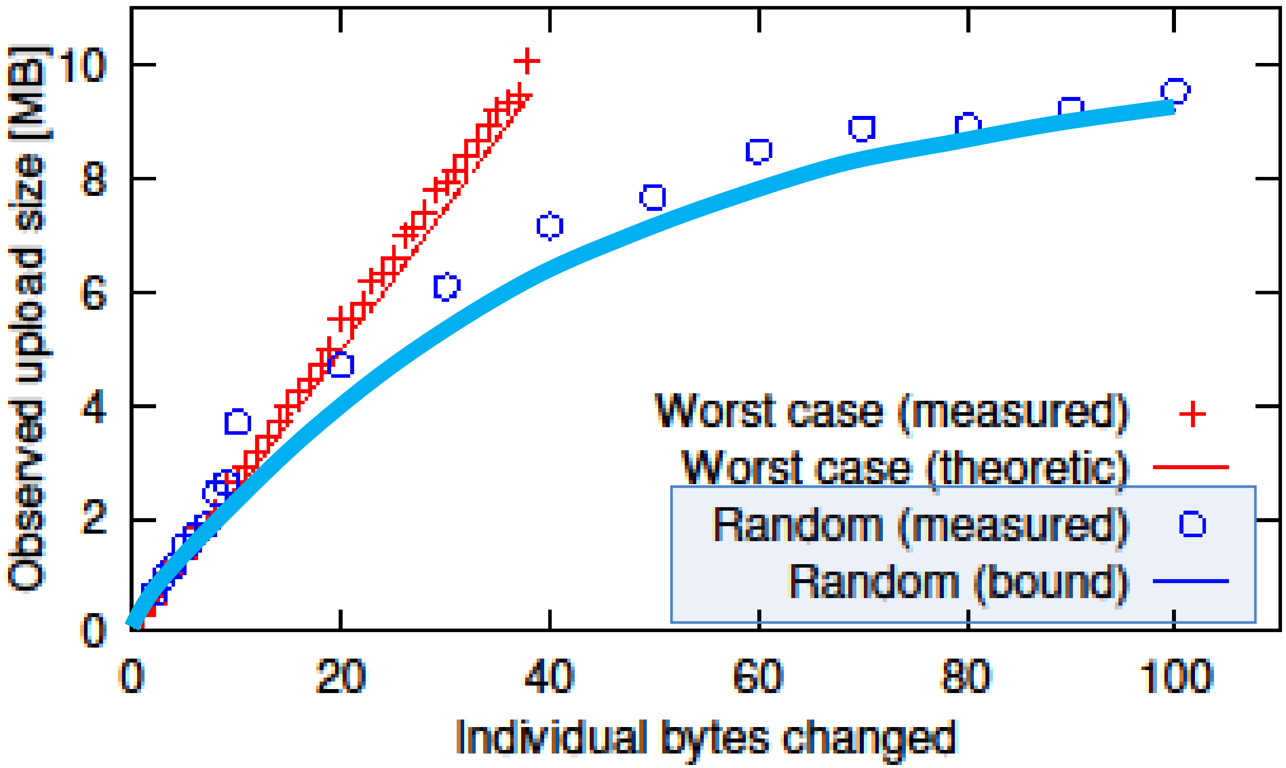
SpiderOak: Bytes uploaded

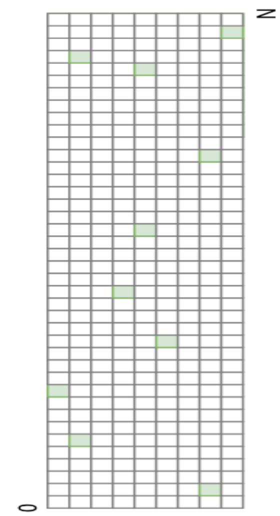




- Block-based encoding (size: 256kB)
- Worst-case overhead: change one byte in each 256kB block
  - E.g., need to change 40 bytes for delta encoding to be 10MB (i.e., the file size)

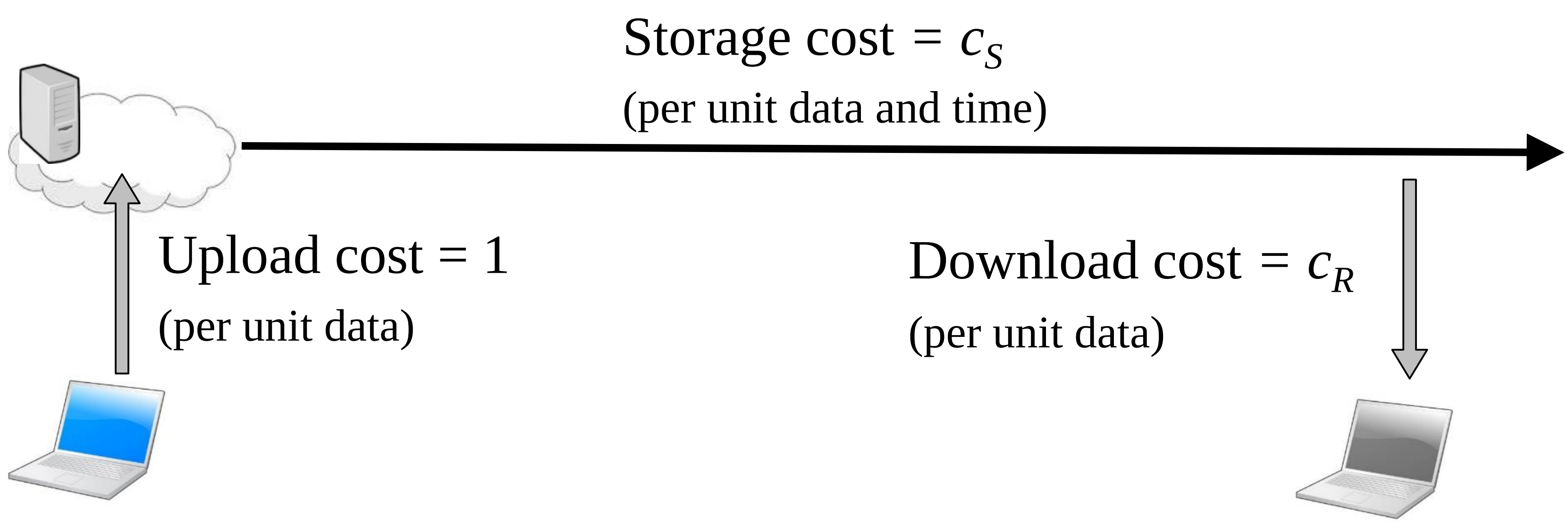
SpiderOak: Bytes uploaded





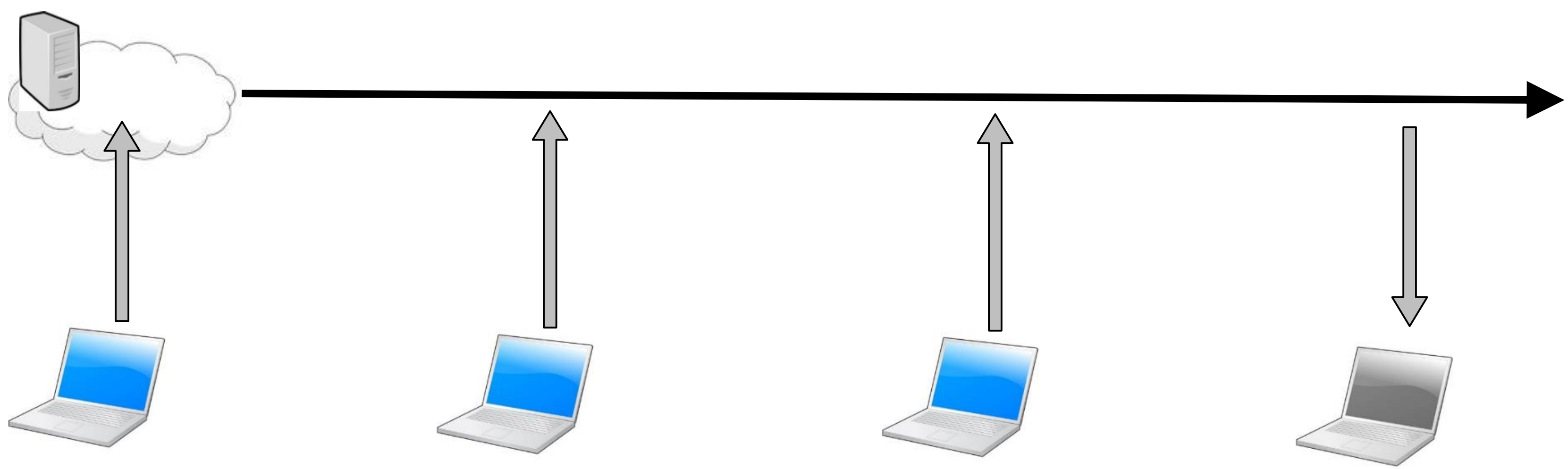
- Block-based encoding (size: 256kB)
- Worst-case overhead: change one byte in each 256kB block
  - E.g., need to change 40 bytes for delta encoding to be 10MB (i.e., the file size)
- Random byte changes: lower bounded by $M(1–(1–1/M)^n) \times 256$kB

Cost model



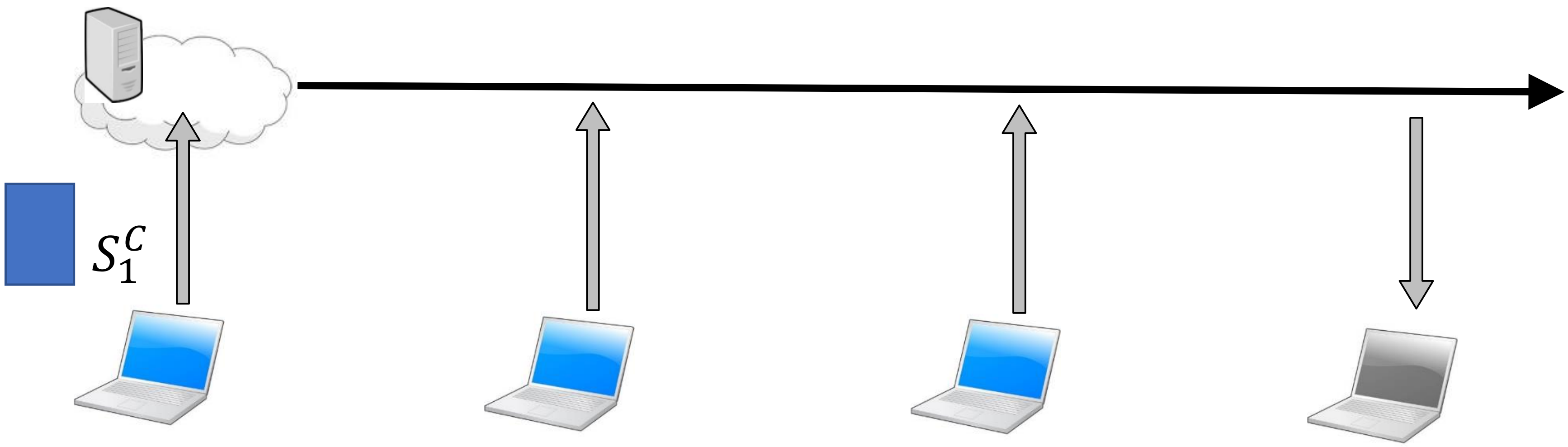
- Normalized upload cost 1 (per unit data); rest relative to this

Cost model



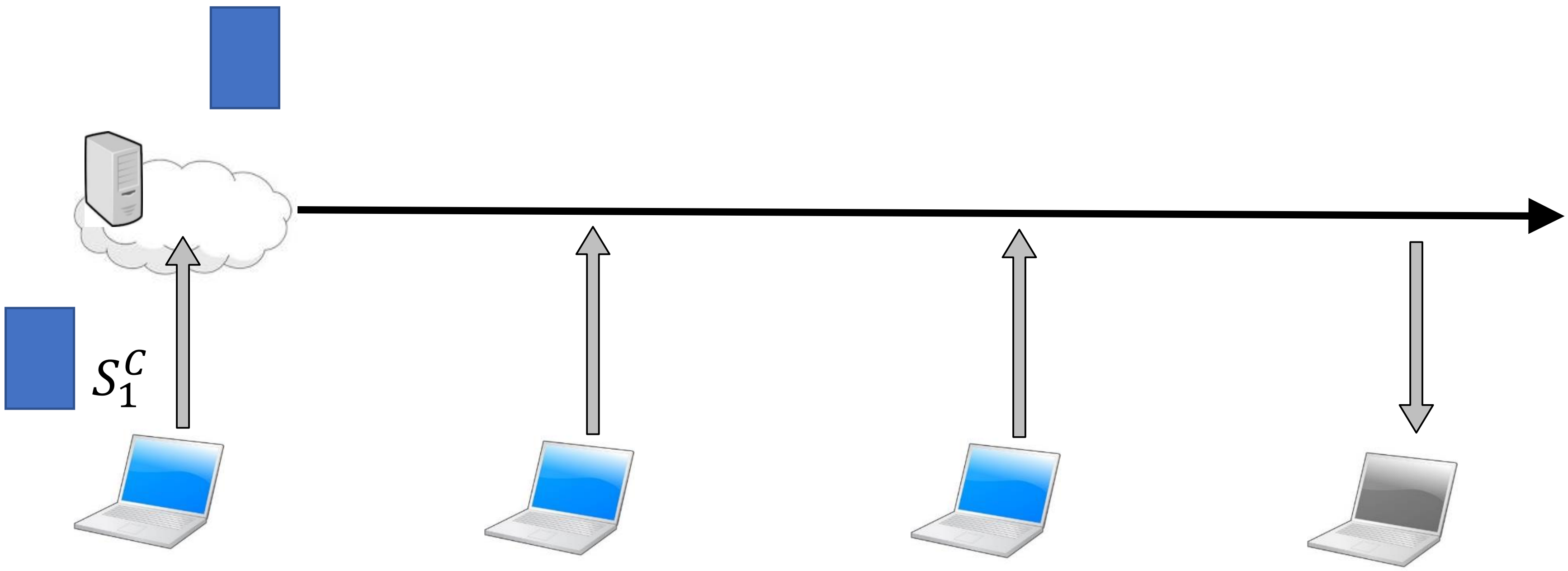
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively

Cost model: Example



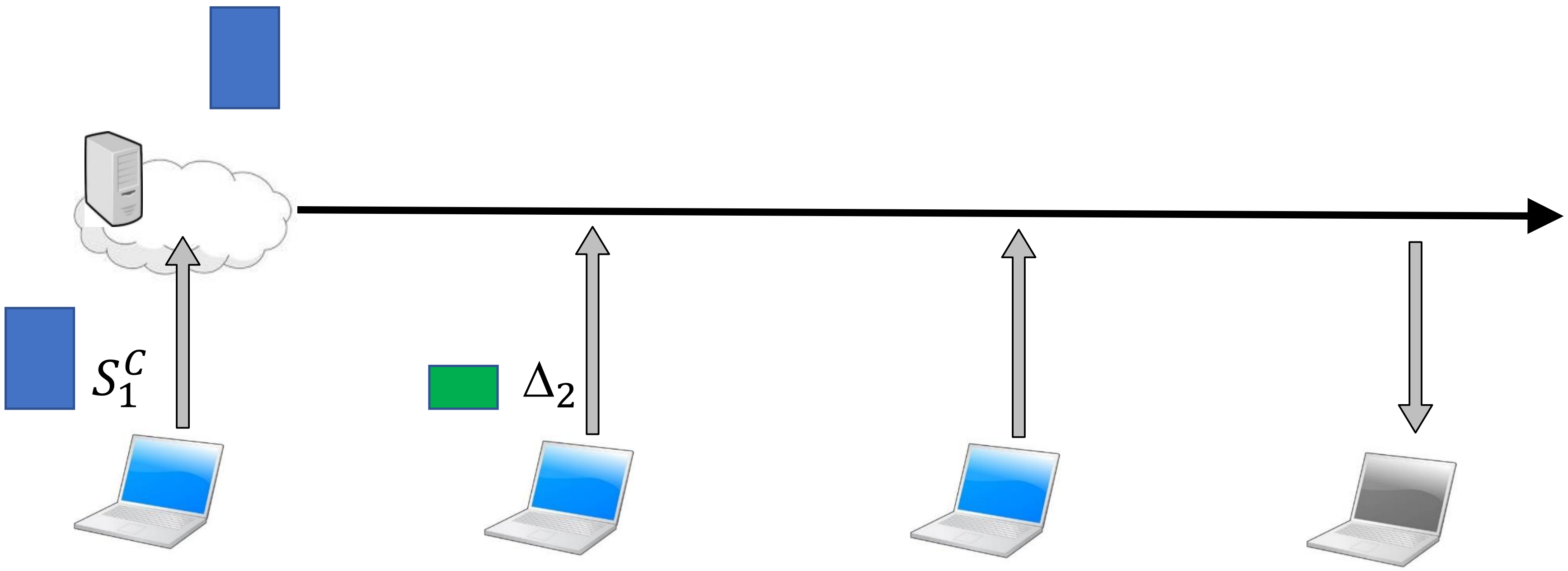
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively

Cost model: Example



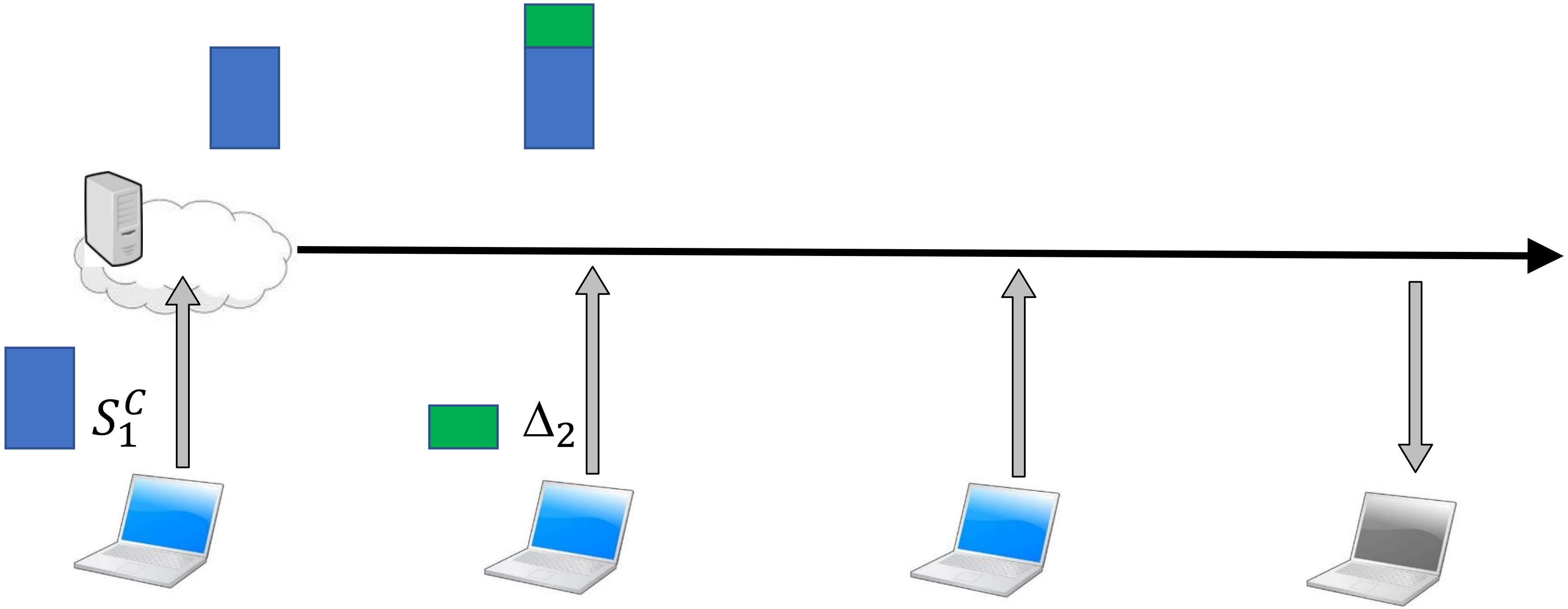
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively

Cost model: Example



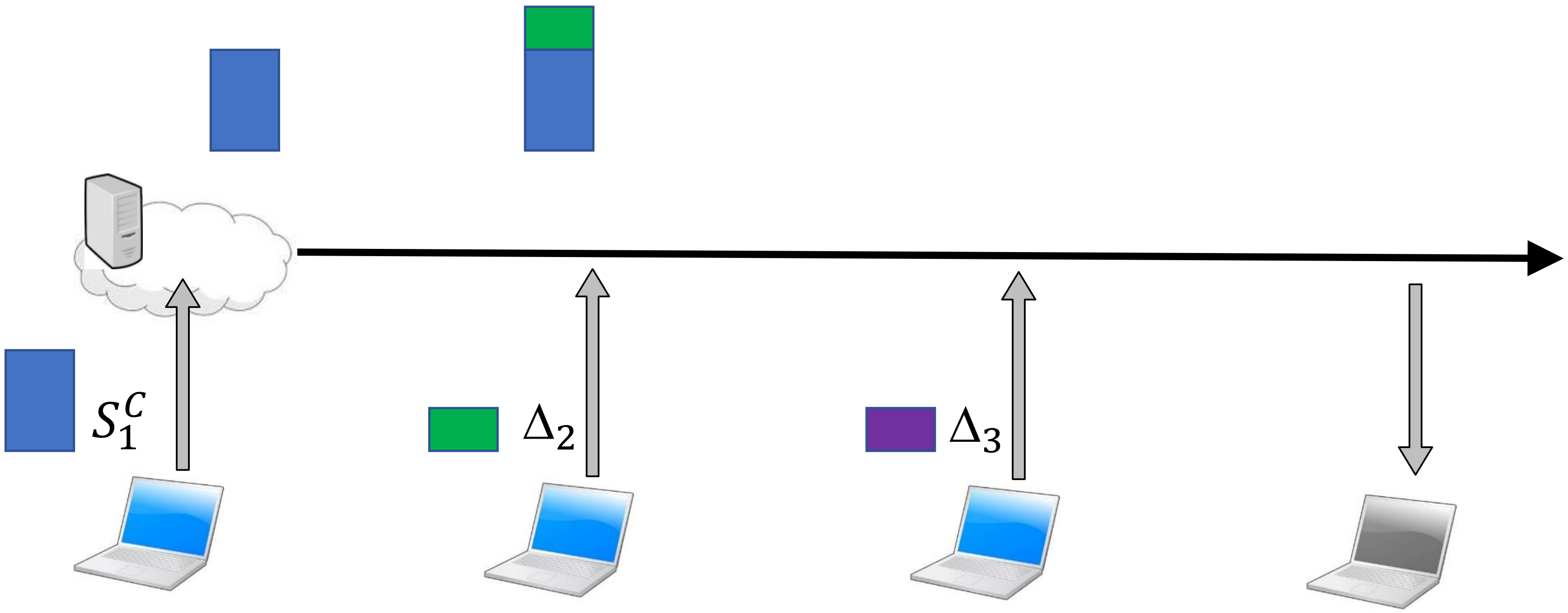
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively

Cost model: Example



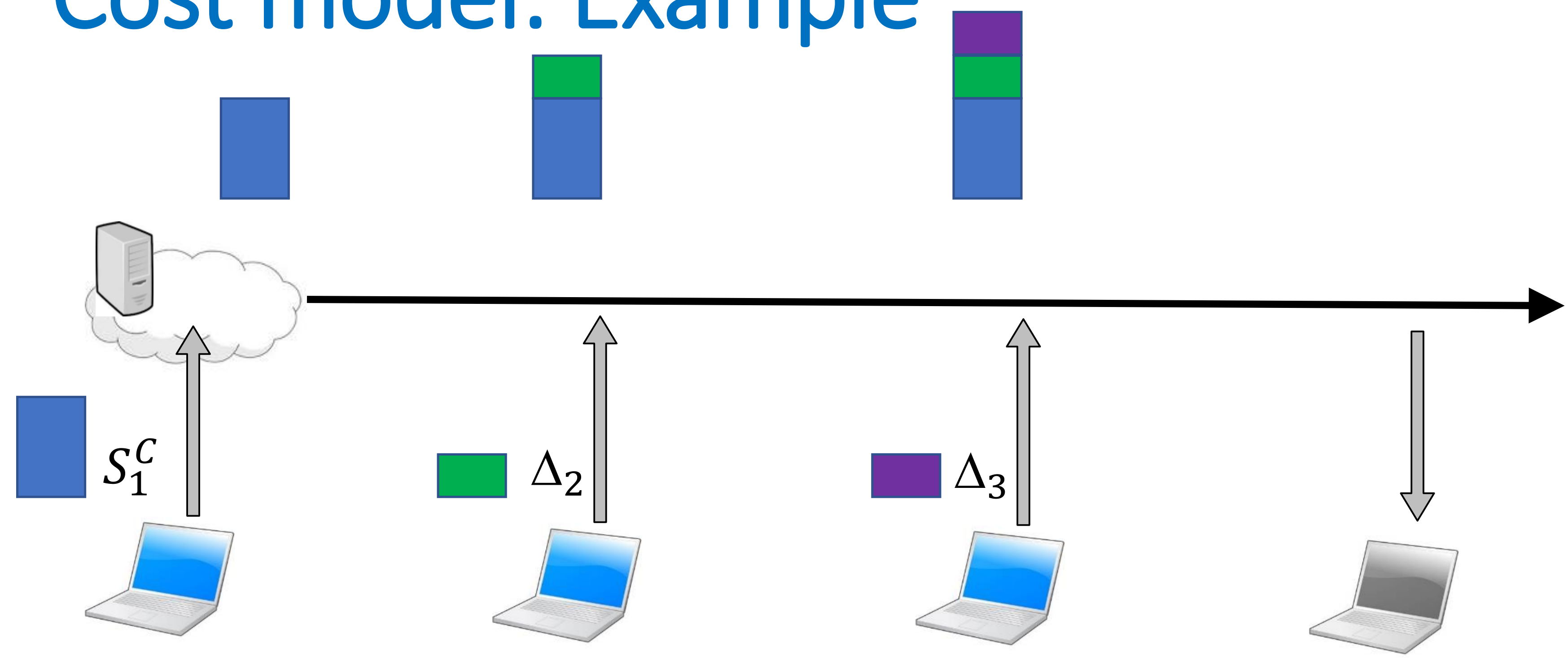
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively

Cost model: Example



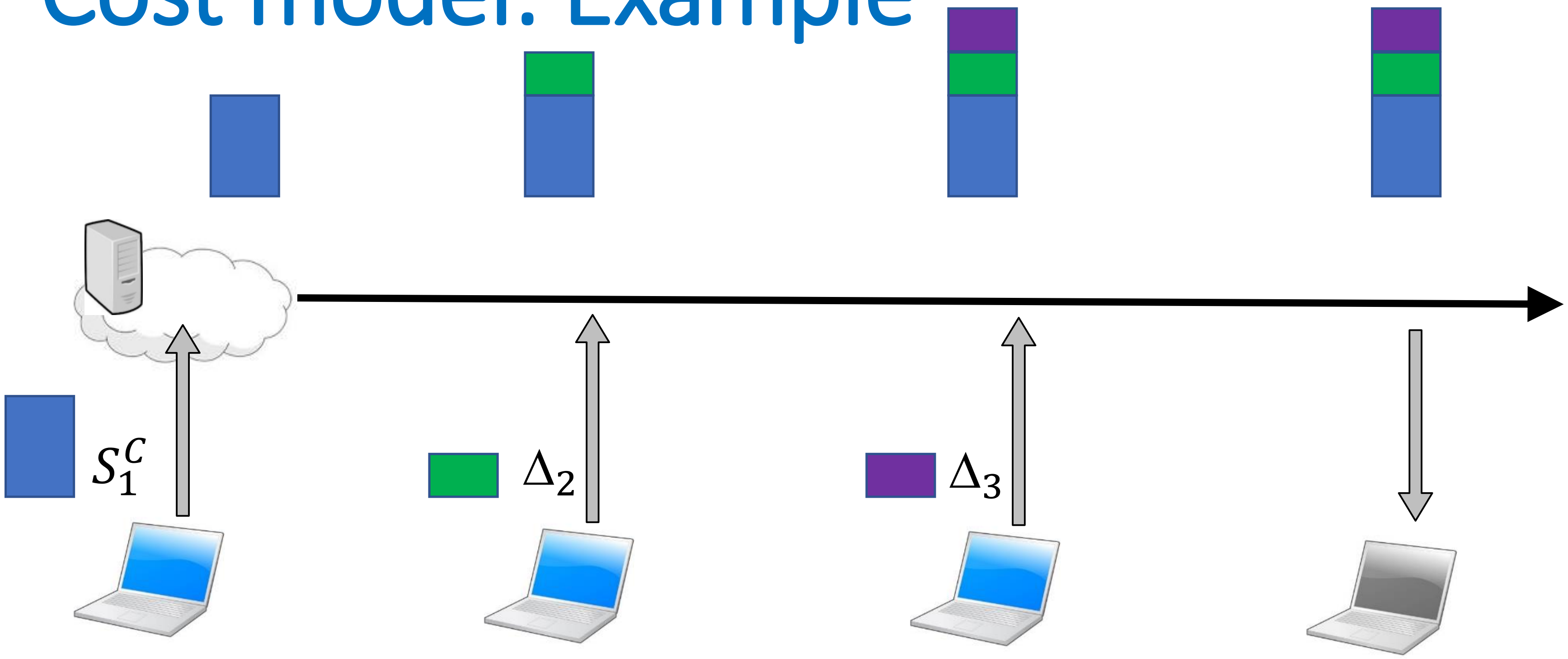
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively

Cost model: Example



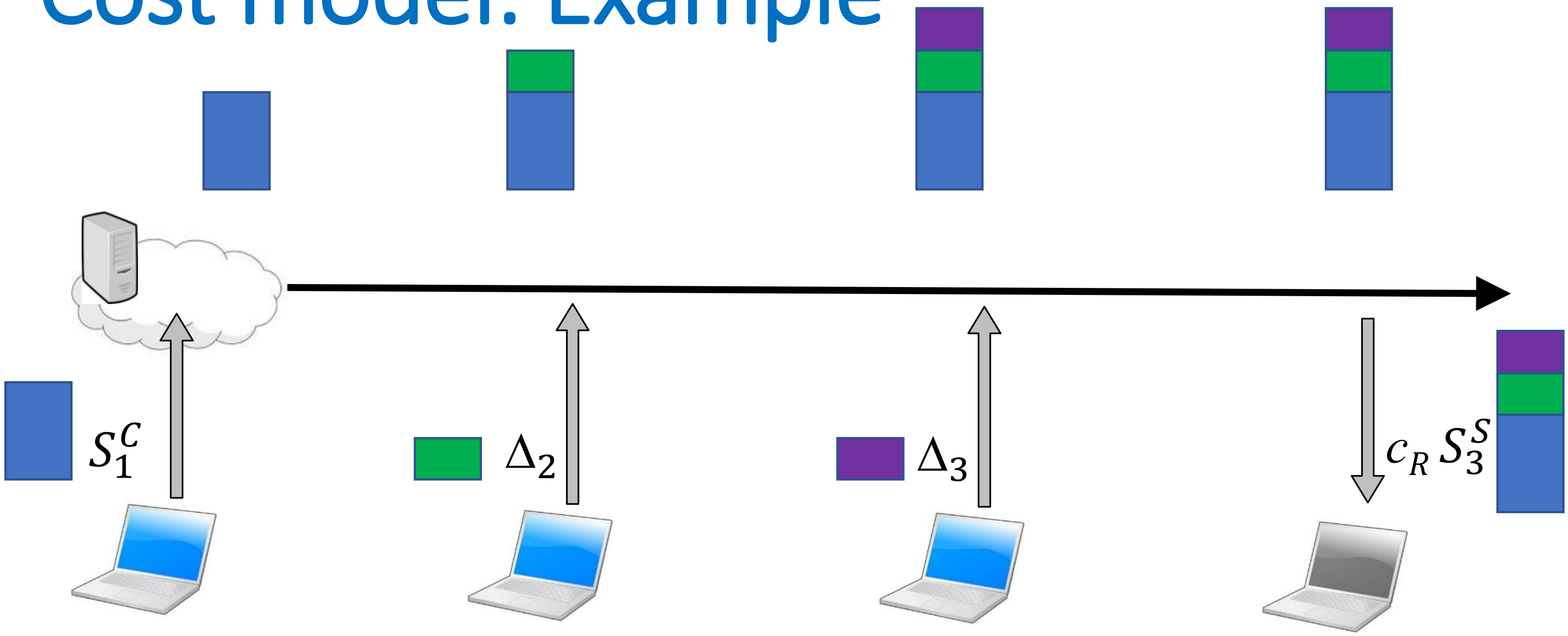
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively

Cost model: Example



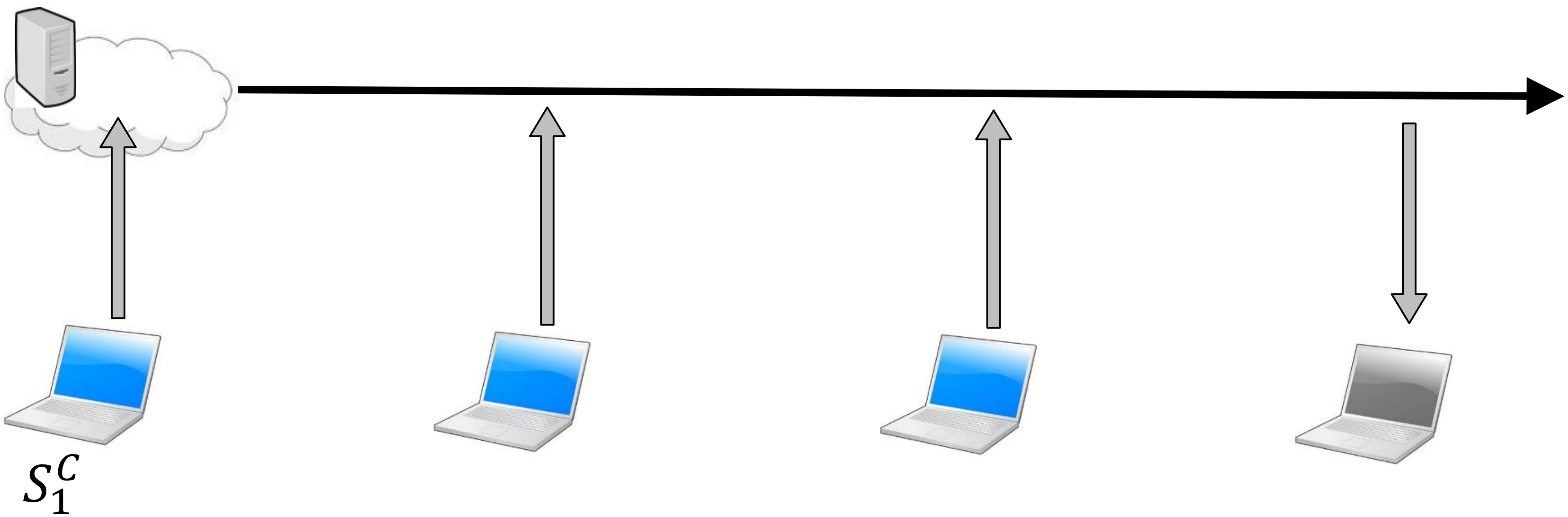
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively

Cost model: Example



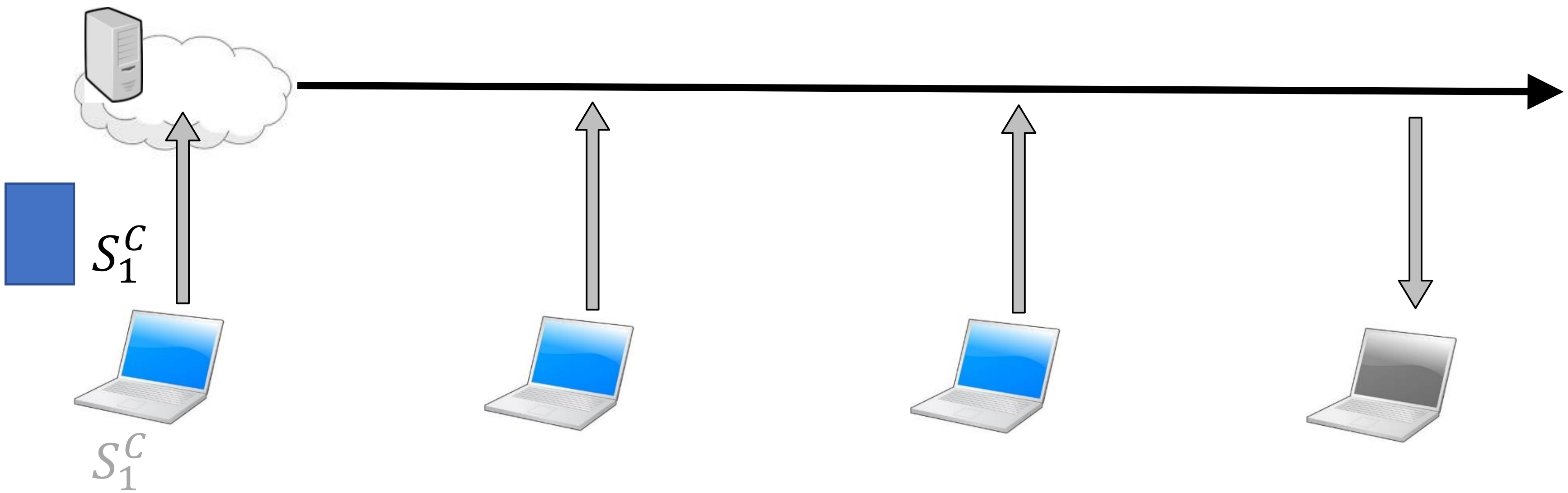
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively

Cost model: Example



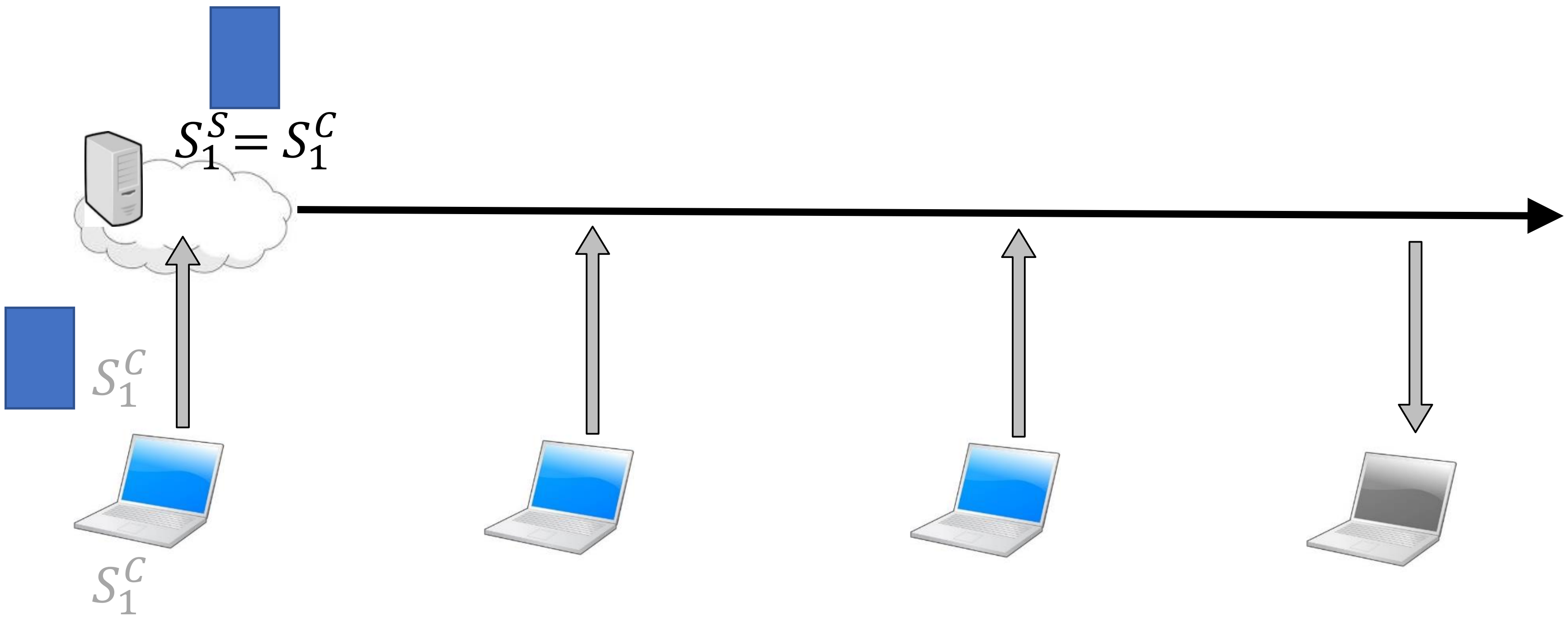
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively
  - S_i^C – file size (e.g., after compression) seen on the client

Cost model: Example



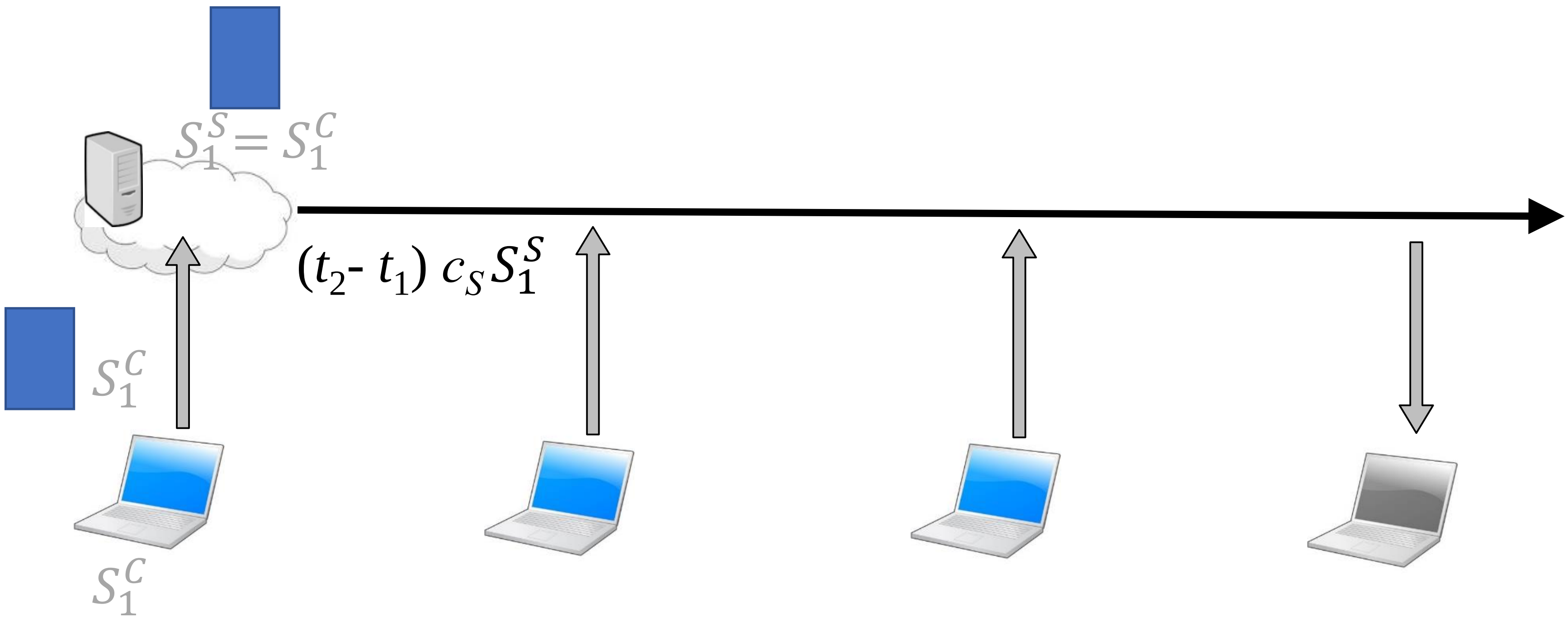
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively
  - S_i^C – file size (e.g., after compression) seen on the client

Cost model: Example



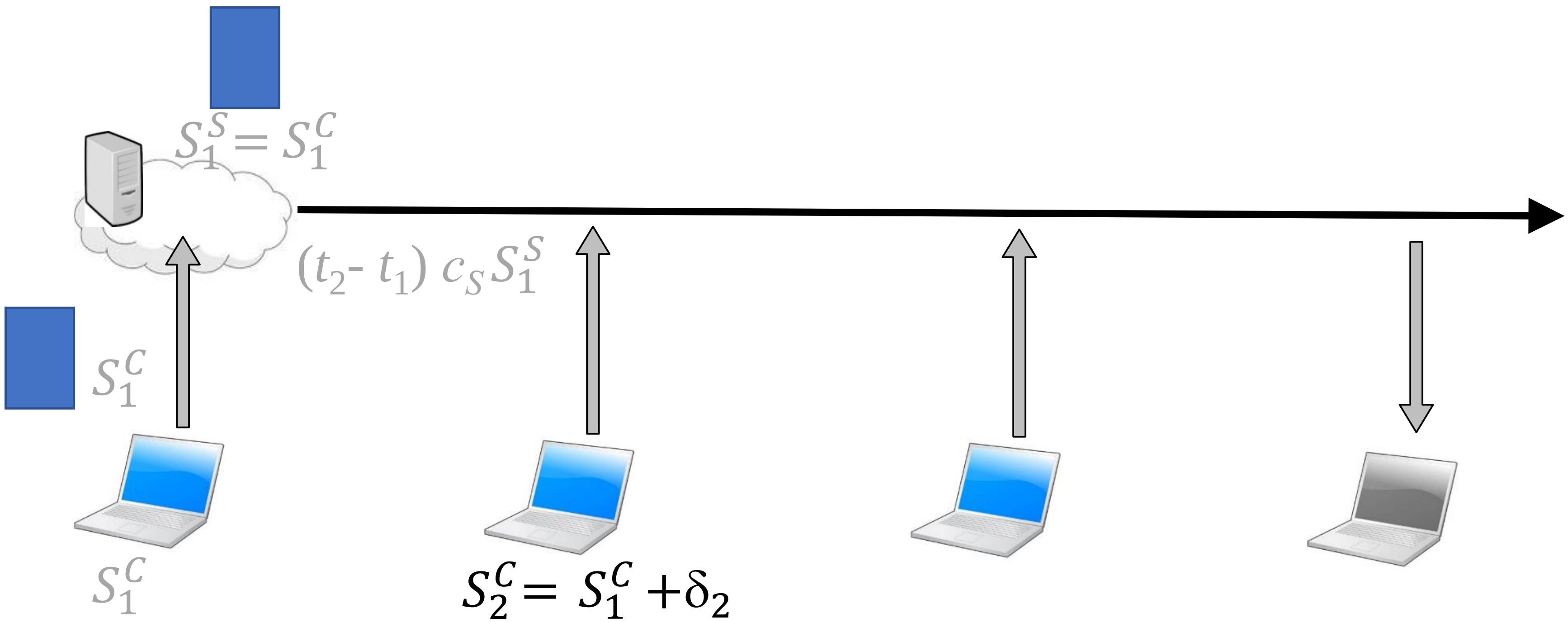
- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively
  - S_i^C – file size (e.g., after compression) seen on the client
  - S_i^S – size of the change log as seen on the server

Cost model: Example



- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively
  - S_i^C – file size (e.g., after compression) seen on the client
  - S_i^S – size of the change log as seen on the server

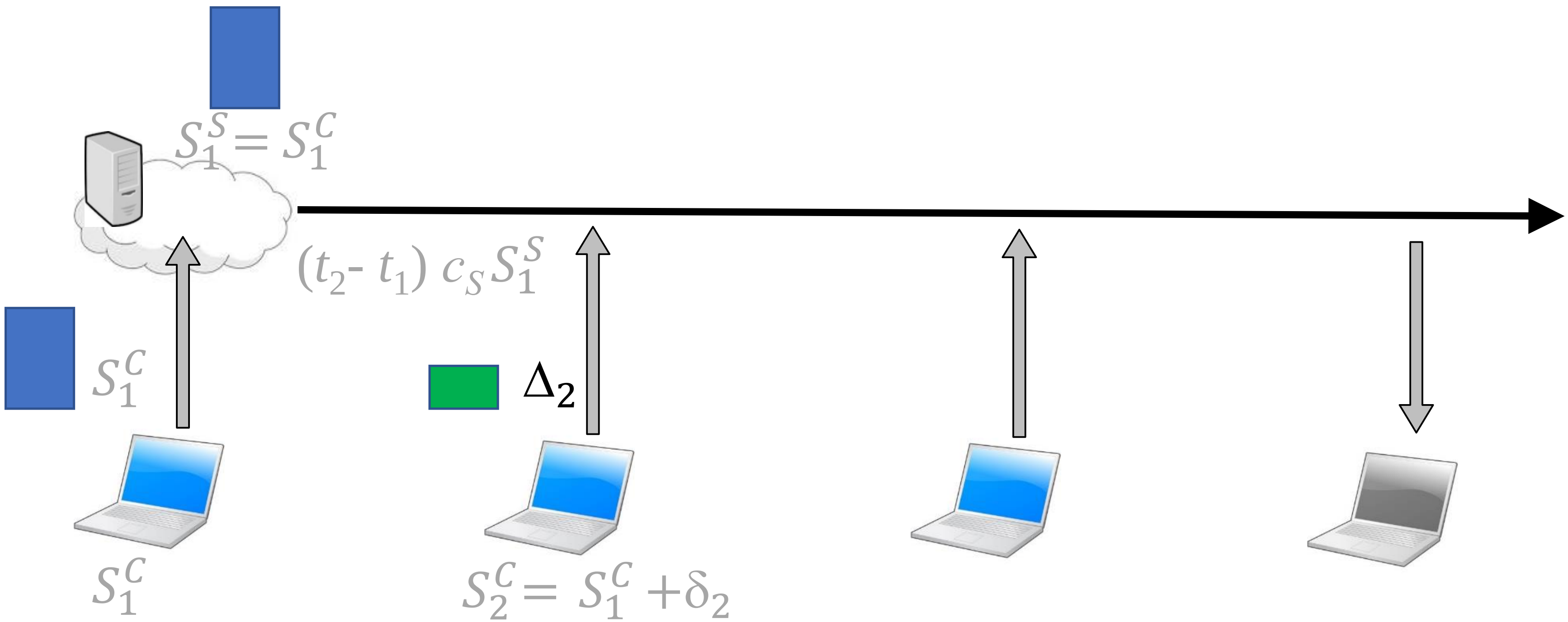
Cost model: Example



- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively
  - S_i^C – file size (e.g., after compression) seen on the client
  - S_i^S – size of the change log as seen on the server
  - δ_i – file size changes seen on the client

Assume $0 \leq \delta_i$

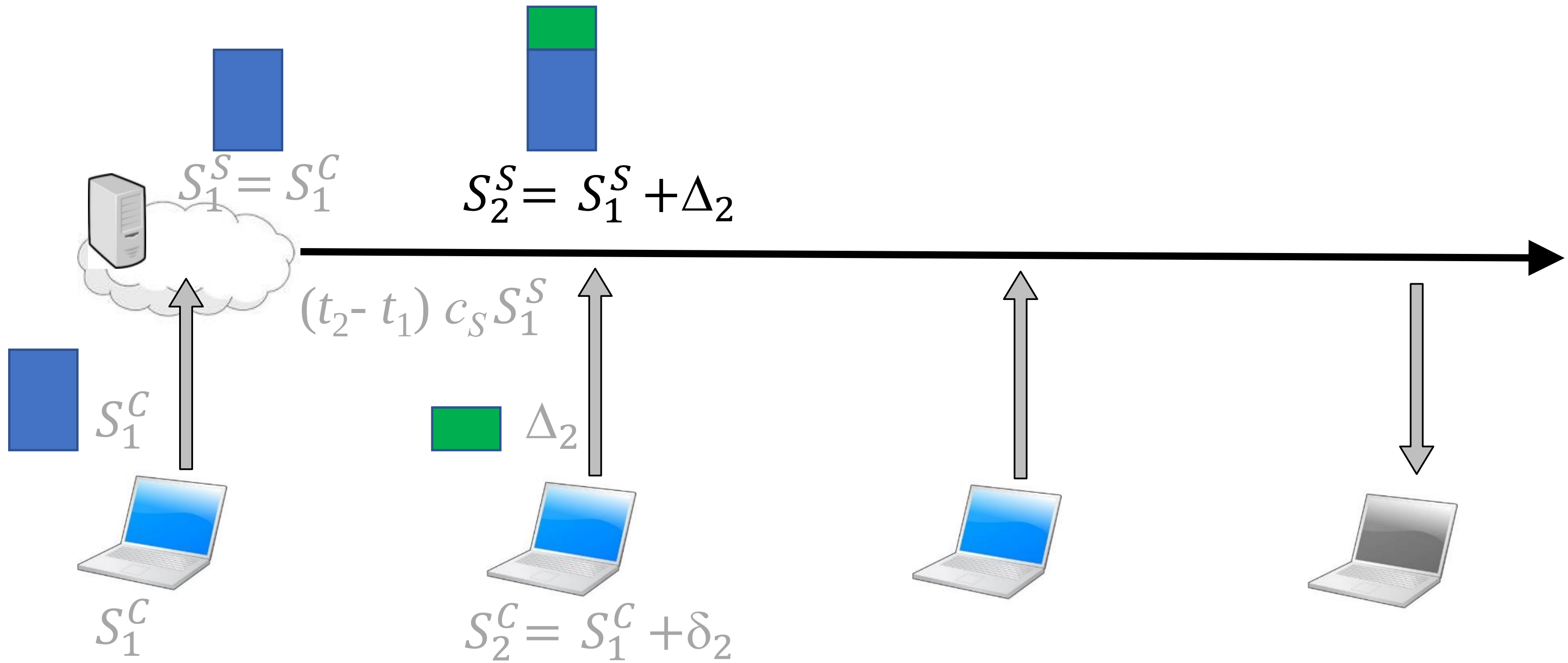
Cost model: Example



- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively
  - S_i^C – file size (e.g., after compression) seen on the client
  - S_i^S – size of the change log as seen on the server
  - δ_i – file size changes seen on the client
  - Δ_i – size of delta encoding submitted to server

$\boxed{\text{Assume } 0 \le \delta_i \le \Delta_i}$

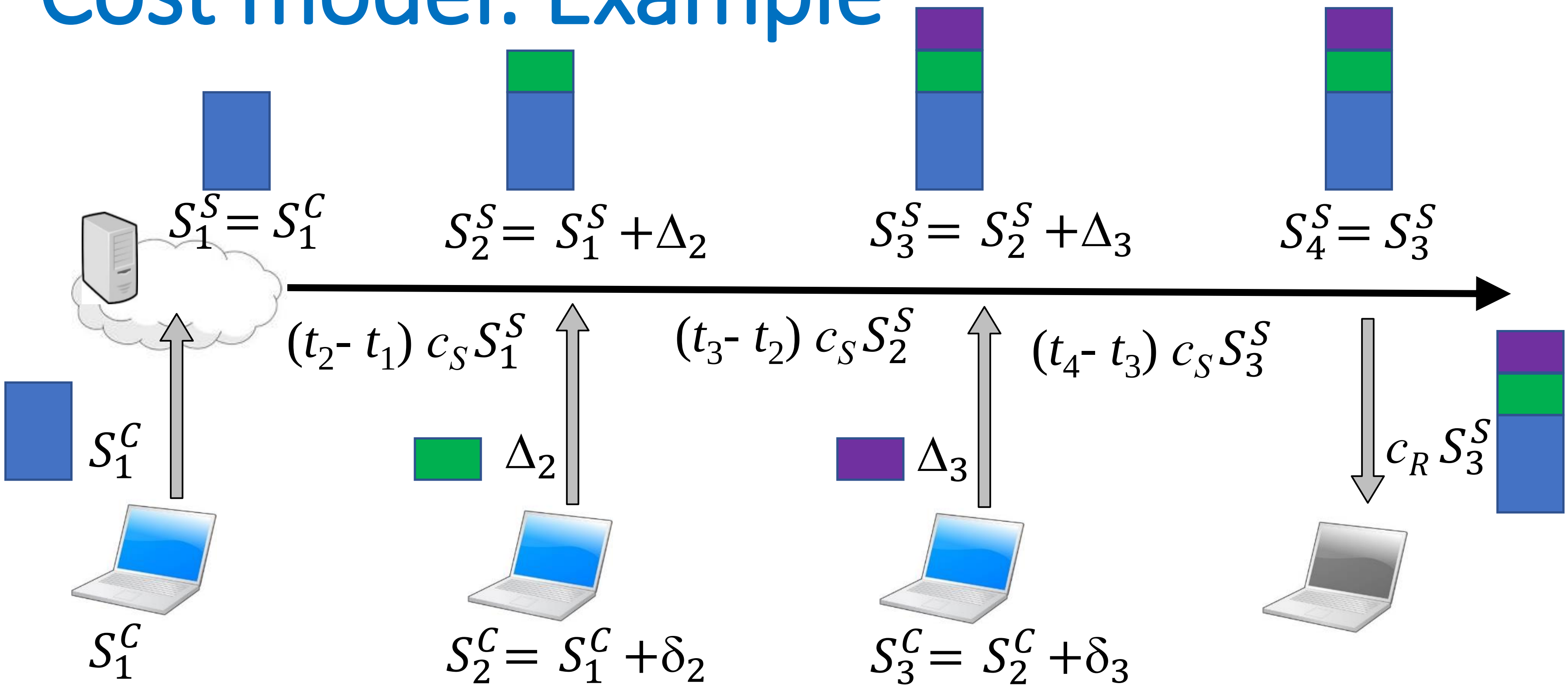
Cost model: Example



- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively
  - S_i^C – file size (e.g., after compression) seen on the client
  - S_i^S – size of the change log as seen on the server
  - δ_i – file size changes seen on the client
  - Δ_i – size of delta encoding submitted to server

$\boxed{\text{Assume } 0 \le \delta_i \le \Delta_i}$

Cost model: Example



- Normalized upload cost 1 (per unit data); rest relative to this
- Arbitrary event sequence $\mathcal{E}$, consisting of $N = |\mathcal{W}| + |\mathcal{R}|$ events, where $\mathcal{W}$ and $\mathcal{R}$ are the set of writes and reads, respectively
  - S_i^C – file size (e.g., after compression) seen on the client
  - S_i^S – size of the change log as seen on the server
  - δ_i – file size changes seen on the client
  - Δ_i – size of delta encoding submitted to server

Assume $0 \le \delta_i \le \Delta_i$

Baseline policies

- Non-CSE

$$\sum_{i\in\mathcal{W}} \Delta_i + c_R \sum_{i\in\mathcal{R}} S_i^c.$$

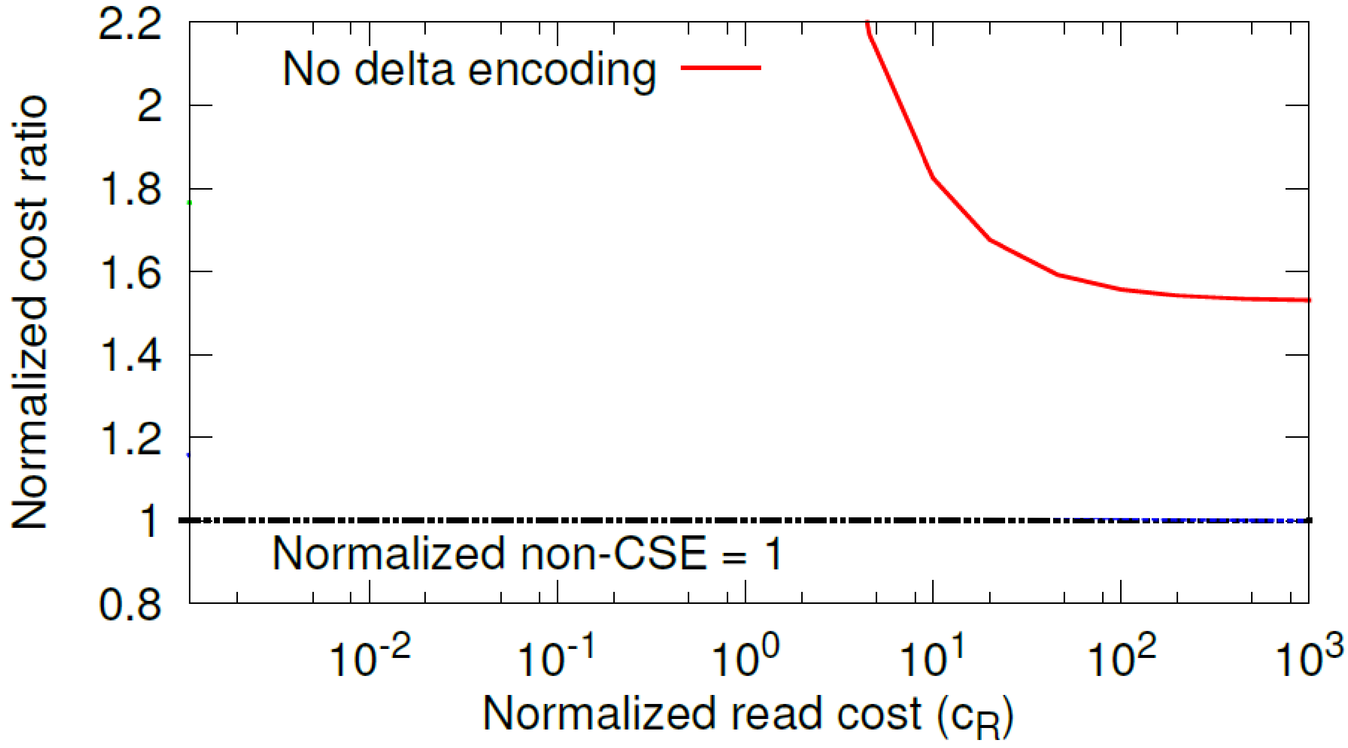
- No delta coding

$$\sum_{i\in\mathcal{W}} S_i^c + c_R \sum_{i\in\mathcal{R}} S_i^c.$$

- Note: Not using delta coding can be arbitrary worse

$$\frac{\sum_{i\in\mathcal{W}} S_i^c + c_R \sum_{i\in\mathcal{R}} S_i^c.}{\sum_{i\in\mathcal{W}} \Delta_i + c_R \sum_{i\in\mathcal{R}} S_i^c.} \xrightarrow[c_R \to 0]{} \frac{\sum_{i\in\mathcal{W}} S_i^c}{\sum_{i\in\mathcal{W}} \Delta_i} \xrightarrow[\Delta/S \to 0]{} \infty$$

Trace-based comparison

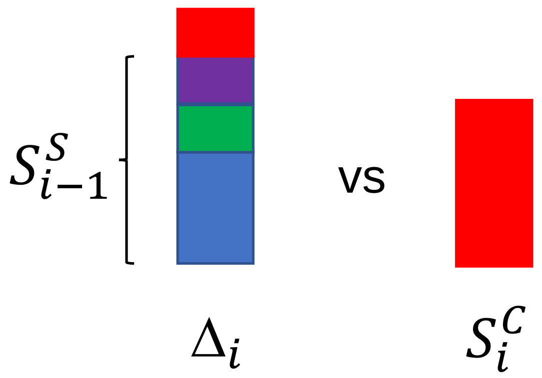


- No delta coding can perform very poorly

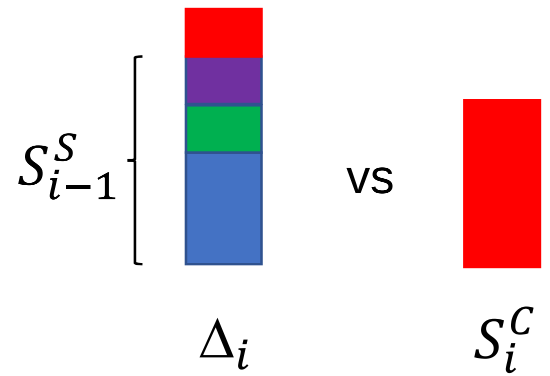
Binary system policies

At each stage either

- Upload new base copy at cost S_i^C
- Append another delta coding Δ_i



Binary system policies



At each stage either

- Upload new base copy at cost S_i^C
- Append another delta coding Δ_i

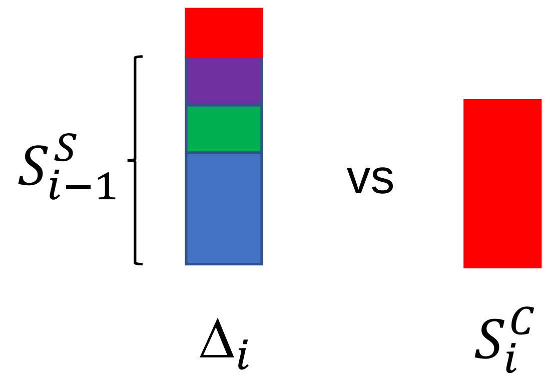
Optimal offline policy (not achievable in practice!)

- Given a sequence $\mathcal{E}$, consider all possible choices and pick one with lowest cost
- Solve using dynamic programing

Binary system policies

At each stage either

- Upload new base copy at cost S_i^C
- Append another delta coding Δ_i



Optimal offline policy (not achievable in practice!)

- Given a sequence $\mathcal{E}$, consider all possible choices and pick one with lowest cost
- Solve using dynamic programing

Threshold-based policy

- Replace base file at cost S_i^C at write event whenever $2S_i^C \leq S_{i-1}^S + \Delta_i$
- Theorem + Proof (in paper): The above policy has a cost ratio to the non-CSE within a factor 2

Binary system policies

At each stage either

- Upload new base copy at cost S_i^C
- Append another delta coding Δ_i

Optimal offline policy (not achievable in practice!)

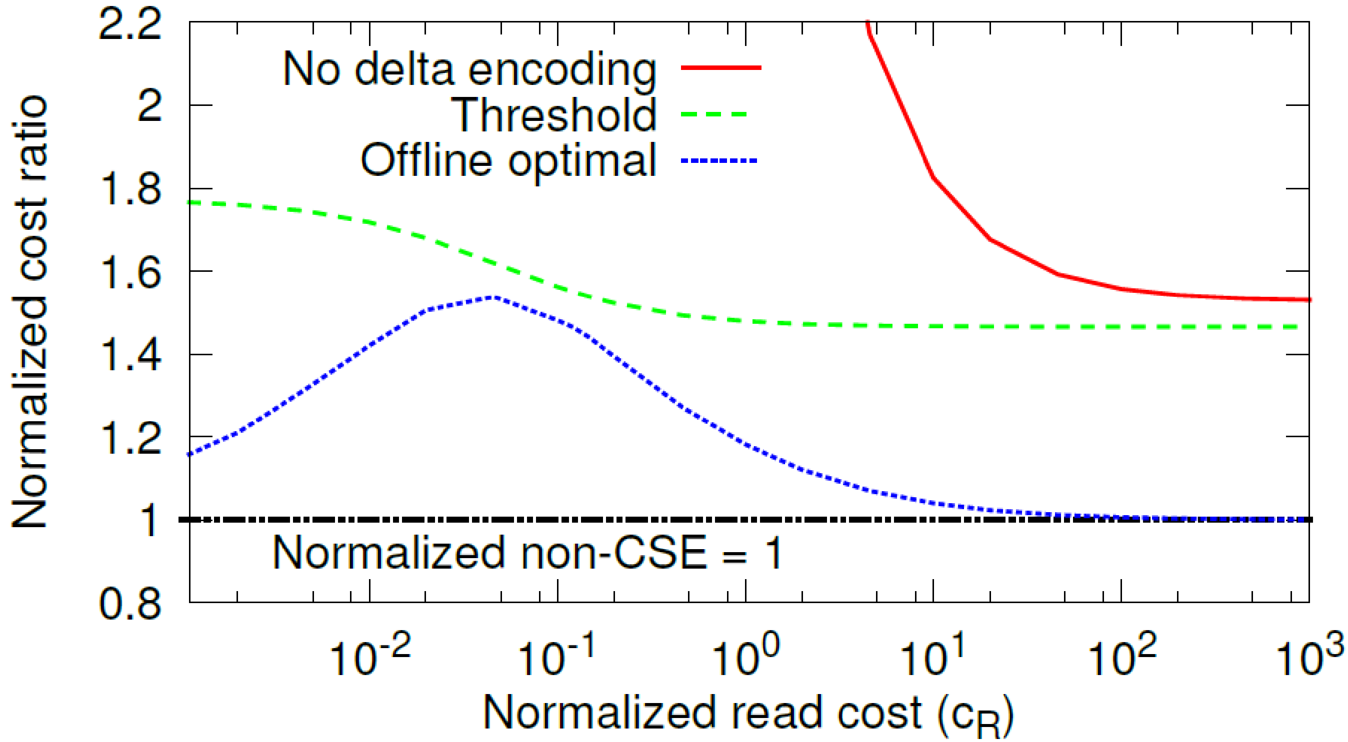
- Given a sequence $\mathcal{E}$, consider all possible choices and pick one with lowest cost
- Solve using dynamic programing

Threshold-based policy

- Replace base file at cost S_i^C at write event whenever $2S_i^C \leq S_{i-1}^S + \Delta_i$
- Theorem + Proof (in paper): The above policy has a cost ratio to the non-CSE within a factor 2

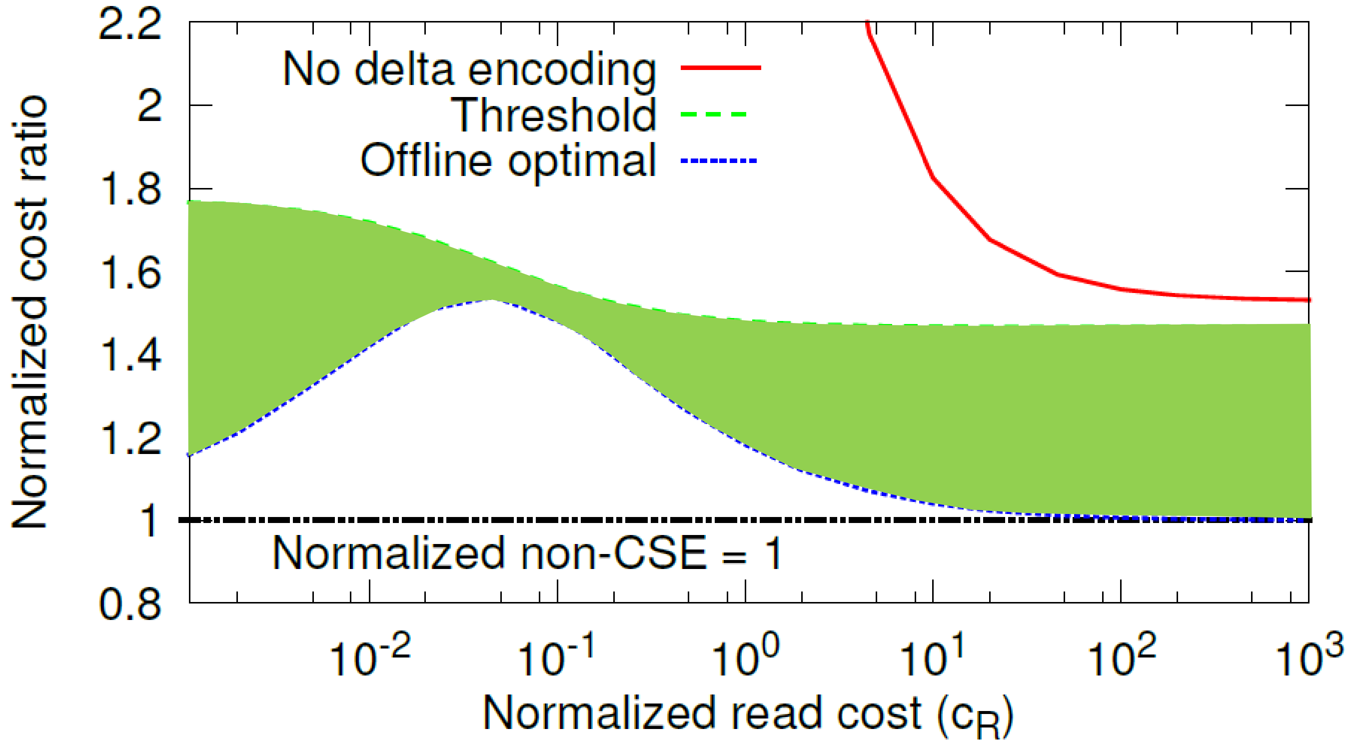
Also, in paper: Results extended to general case (with $c_S > 0$)

Trace-based comparison



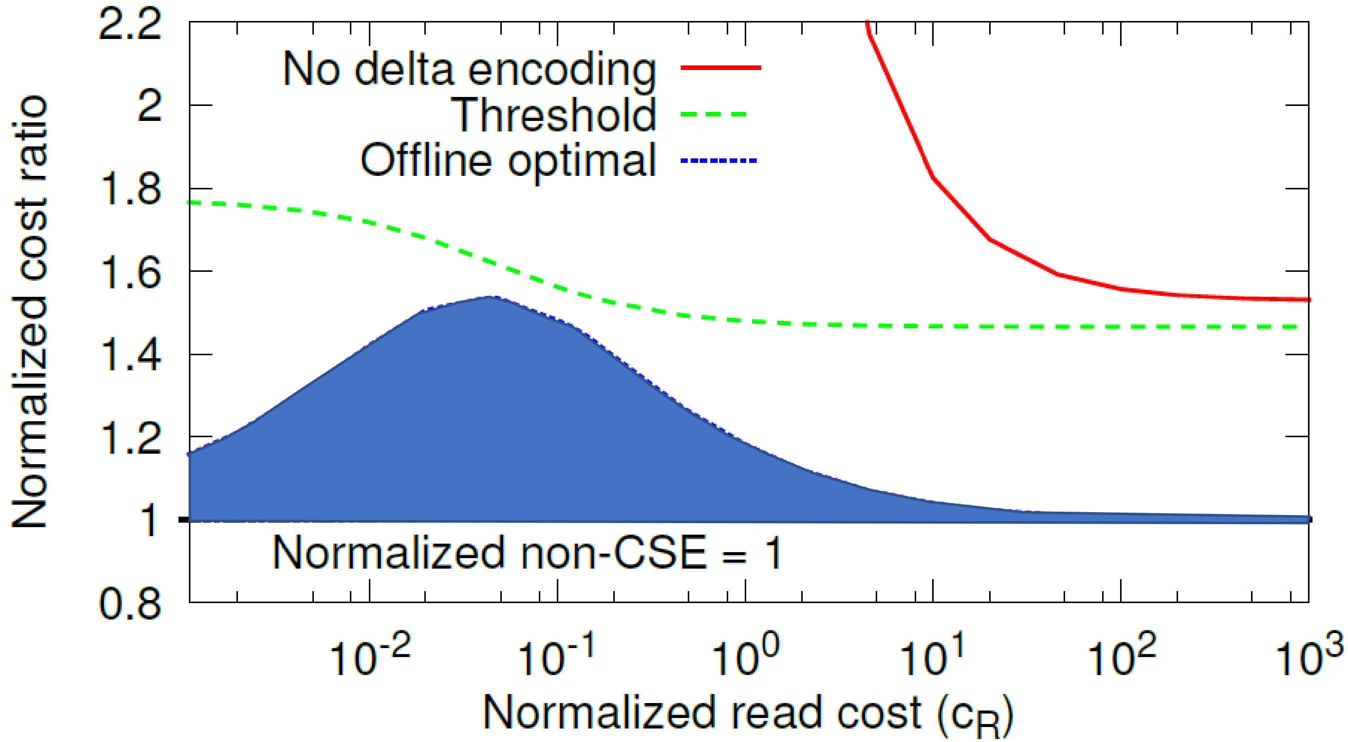
- No delta coding can perform very poorly
- On average threshold policy typically perform better (e.g., within 1.5 of offline optimal)

Trace-based comparison



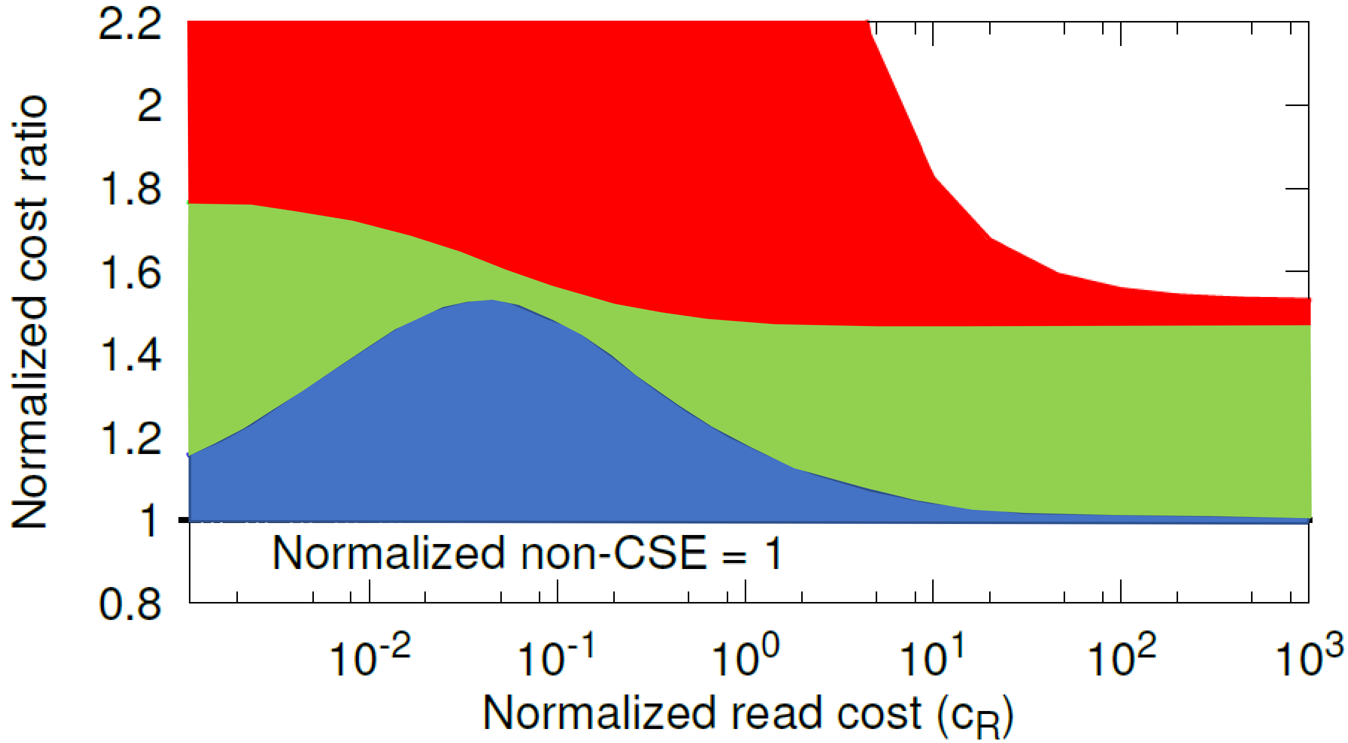
- No delta coding can perform very poorly
- On average threshold policy typically perform better (e.g., within 1.5 of offline optimal)

Trace-based comparison



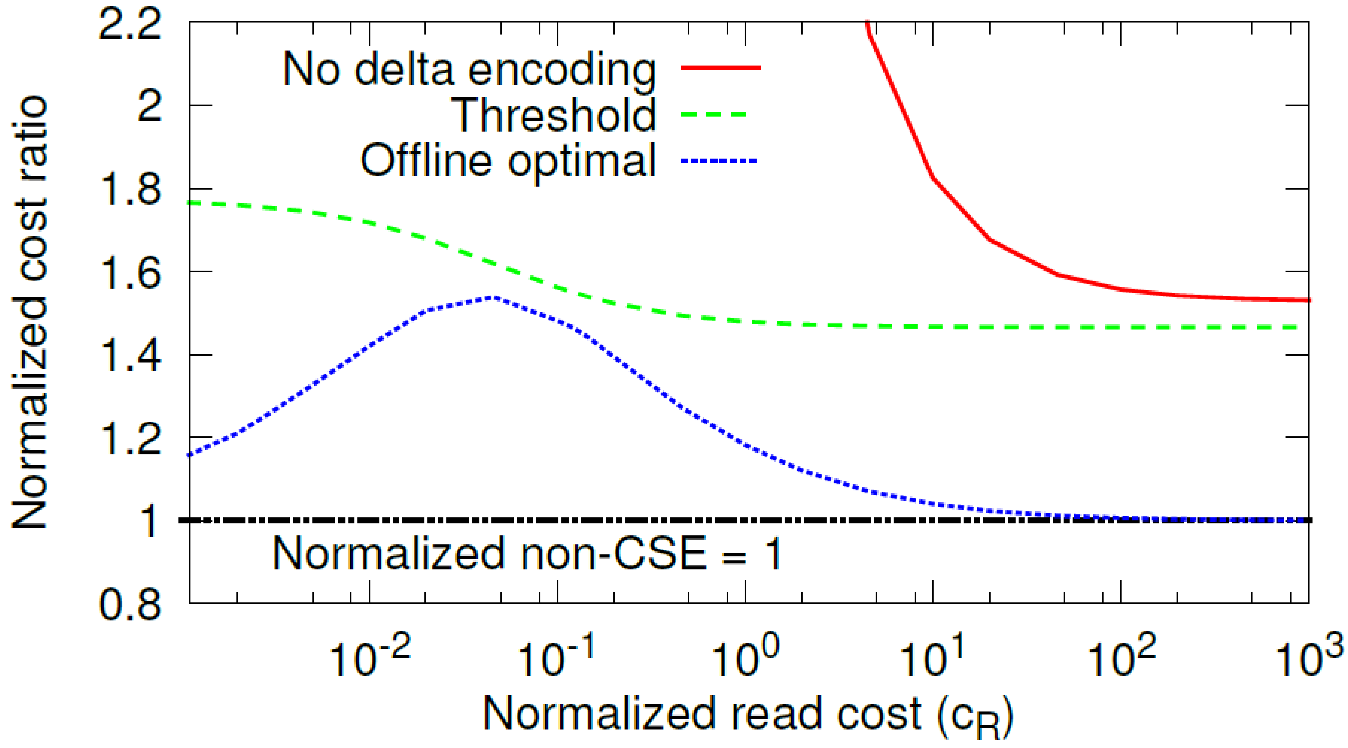
- No delta coding can perform very poorly
- On average threshold policy typically perform better (e.g., within 1.5 of offline optimal)
- Inherent penalty to using CSE (e.g., upwards 1.5 difference with optimal)

Trace-based comparison



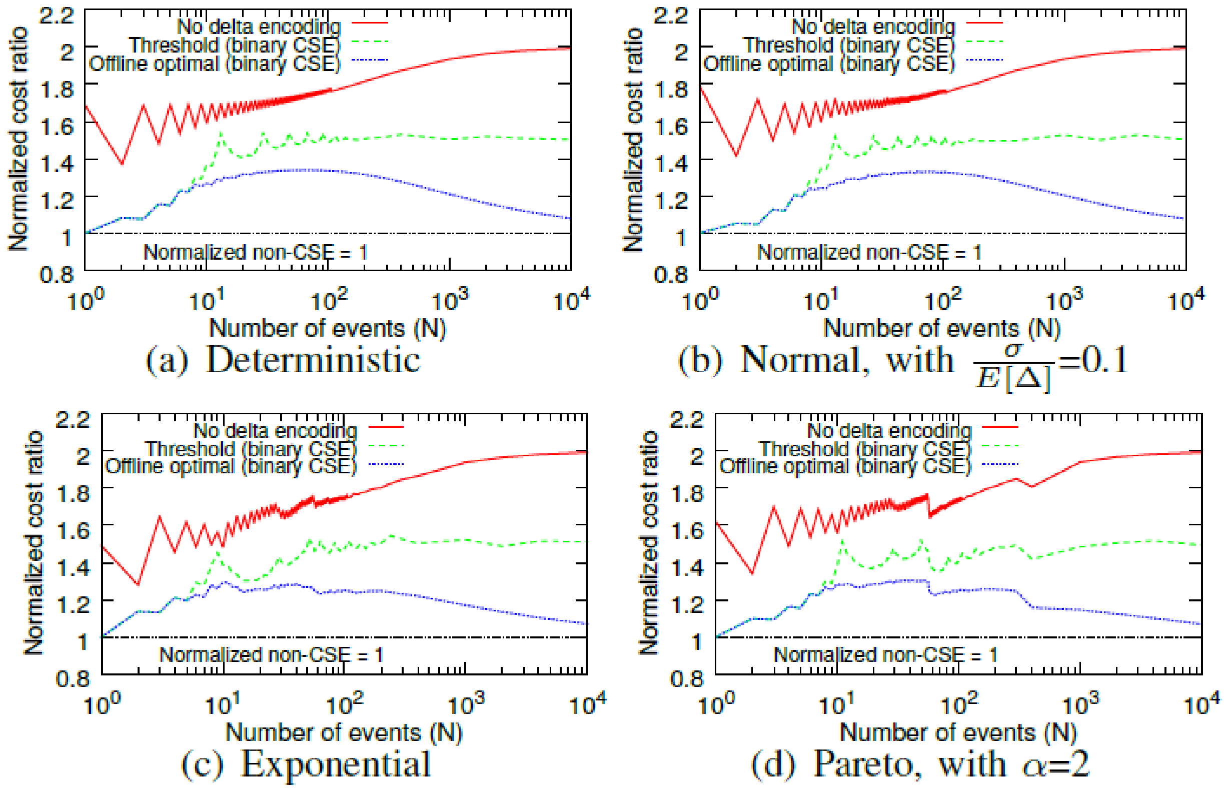
- No delta coding can perform very poorly
- On average threshold policy typically perform better (e.g., within 1.5 of offline optimal)
- Inherent penalty to using CSE (e.g., upwards 1.5 difference with optimal)

Trace-based comparison



- No delta coding can perform very poorly
- On average threshold policy typically perform better (e.g., within 1.5 of offline optimal)
- Inherent penalty to using CSE (e.g., upwards 1.5 difference with optimal)

Impact of size distributions



(a) Deterministic

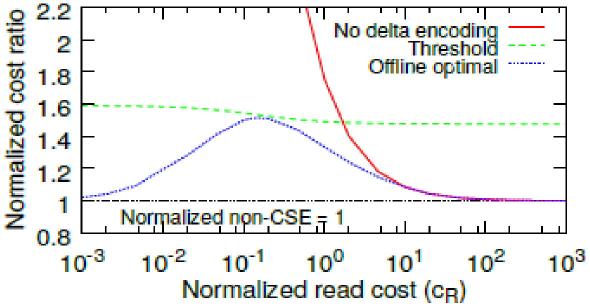
(b) Normal, with $\frac{\sigma}{E[\Delta]}$=0.1

(c) Exponential

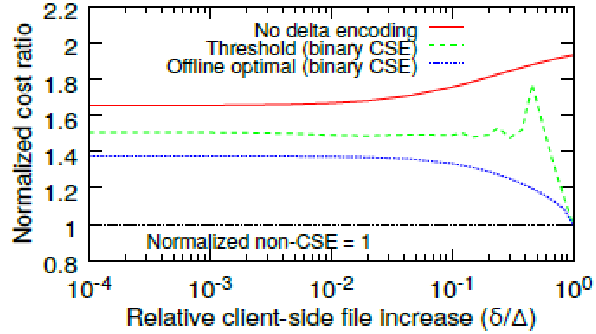
(d) Pareto, with α=2

- Results consistent for both **long-tailed** (Pareto) and **short-tailed** (exponential, normal, and deterministic) size **distributions**

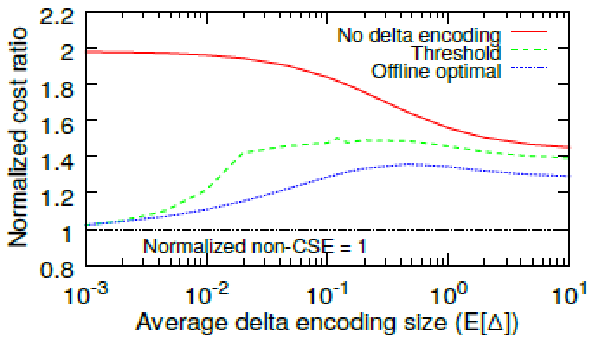
Comparison across parameters



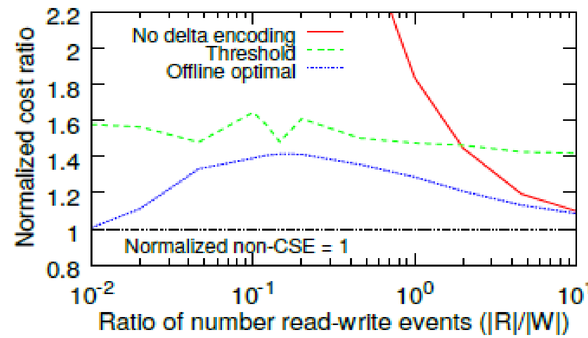
(a) Normalized read cost



(b) Relative client-side file increase



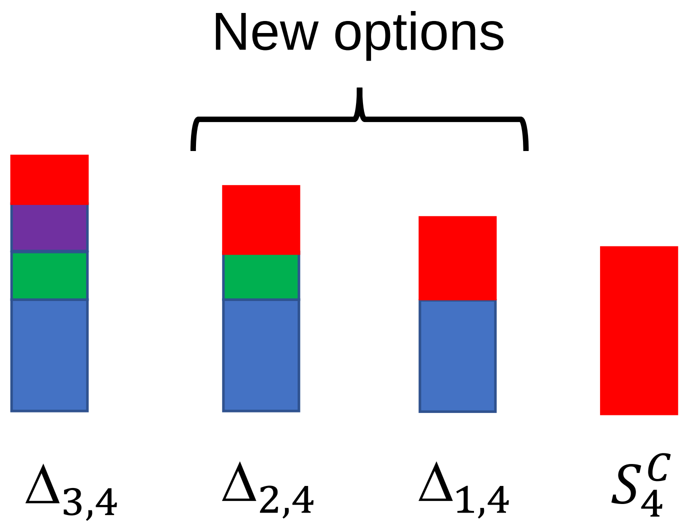
(c) Average delta encoding size



(d) Ratio of read-write events

- Results consistent for broad range of **workload parameters**

More advanced multi-step policies



$$S_4^S = \begin{cases} S_3^S + \Delta_{3,4} \\ S_2^S + \Delta_{2,4} \\ S_1^S + \Delta_{1,4} \\ S_4^C \end{cases}$$

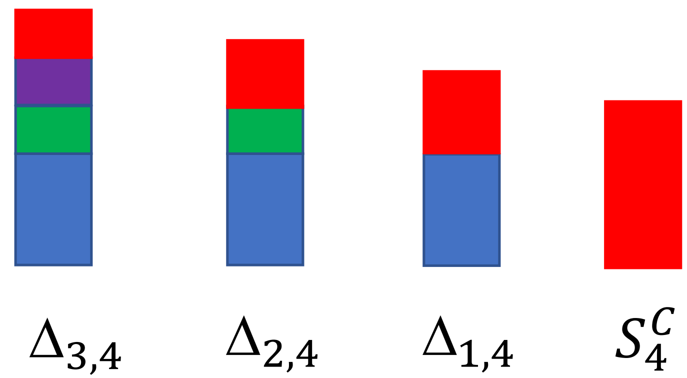
More advanced multi-step policies

- Offline optimal not feasible to calculate
  - Complexity lower bounded by $\Omega(3^N)$
  - [conjecture] Complexity can be upper bounded by $O(4^N)$
- Greedy policy
  - Upload delta change $\Delta_{i(j)*,j}$ that minimizes

    $$\arg\min_{i \in \log} \Delta_{i,j} + f c_R(S_i^s + \Delta_{i,j}),$$

  - Greedy also have optional "threshold extension"

    $$S_{i(j)*}^s + \Delta_{i(j)^*,j} \geq 2 S_j^c.$$



$$S_4^S = \begin{cases} S_3^S + \Delta_{3,4} \\ S_2^S + \Delta_{2,4} \\ S_1^S + \Delta_{1,4} \\ S_4^C \end{cases}$$

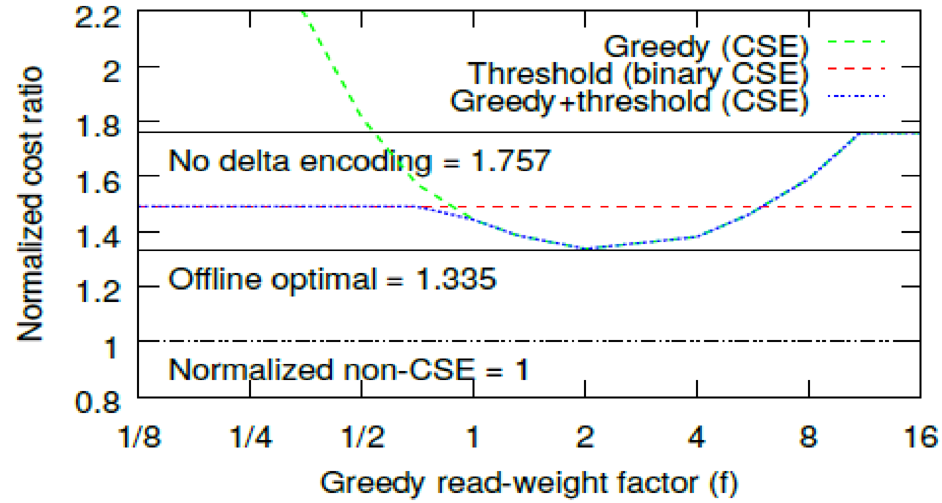
More advanced multi-step policies

- Offline optimal not feasible to calculate
  - Complexity lower bounded by $\Omega(3^N)$
  - [conjecture] Complexity can be upper bounded by $O(4^N)$
- Greedy policy
  - Upload delta change $\Delta_{i(j)*,j}$ that minimizes

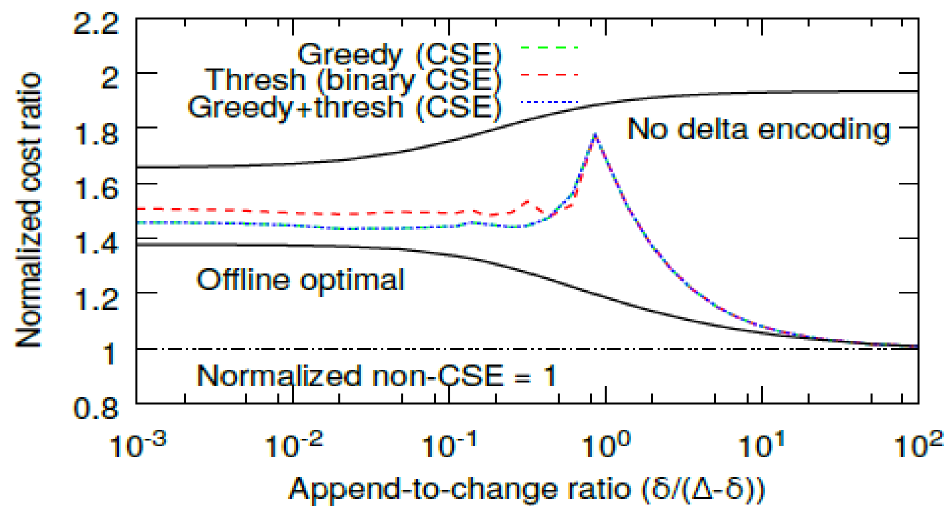
$$\arg\min_{i \in \log} \Delta_{i,j} + f c_R(S_i^s + \Delta_{i,j}),$$

  - Greedy also have optional “threshold extension”

$$S_{i(j)*}^s + \Delta_{i(j)*,j} \geq 2 S_j^c.$$



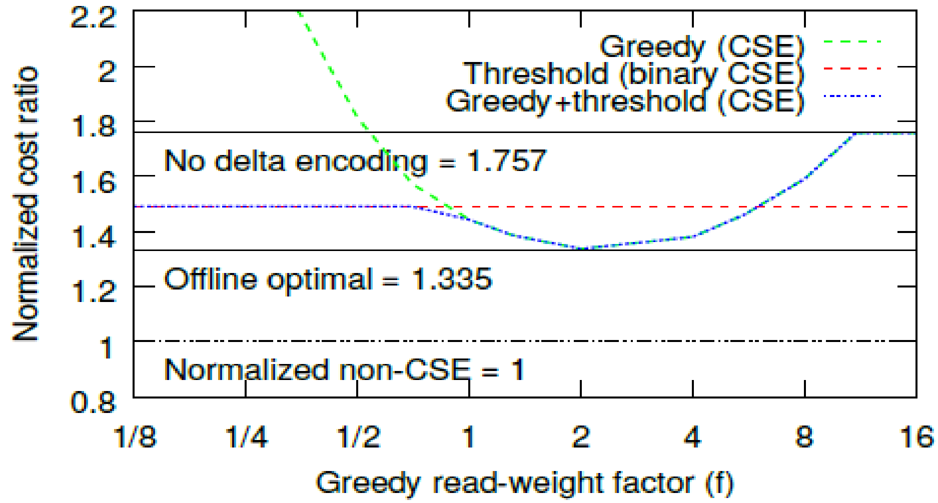
(a) Greedy read-weight factor



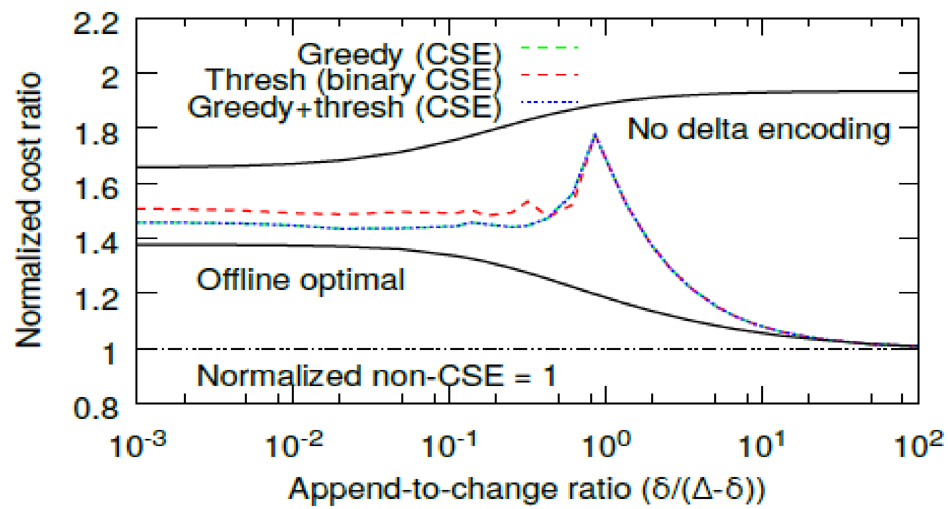
(b) Append-to-change ratio

More advanced multi-step policies

- Simple binary threshold policy still does well



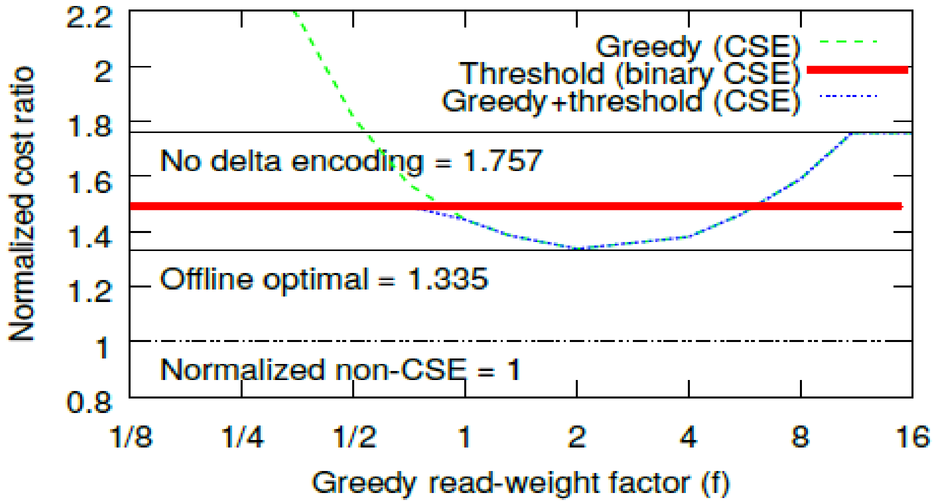
(a) Greedy read-weight factor



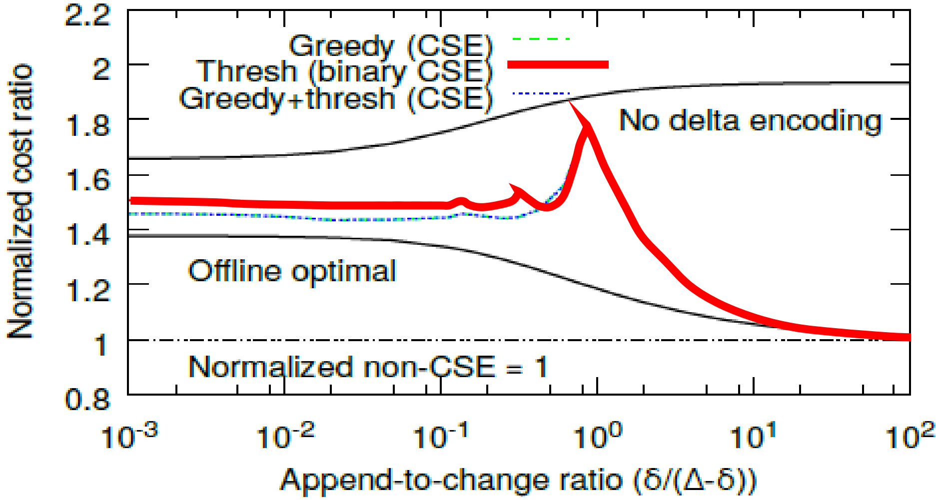
(b) Append-to-change ratio

More advanced multi-step policies

- Simple binary threshold policy still does well

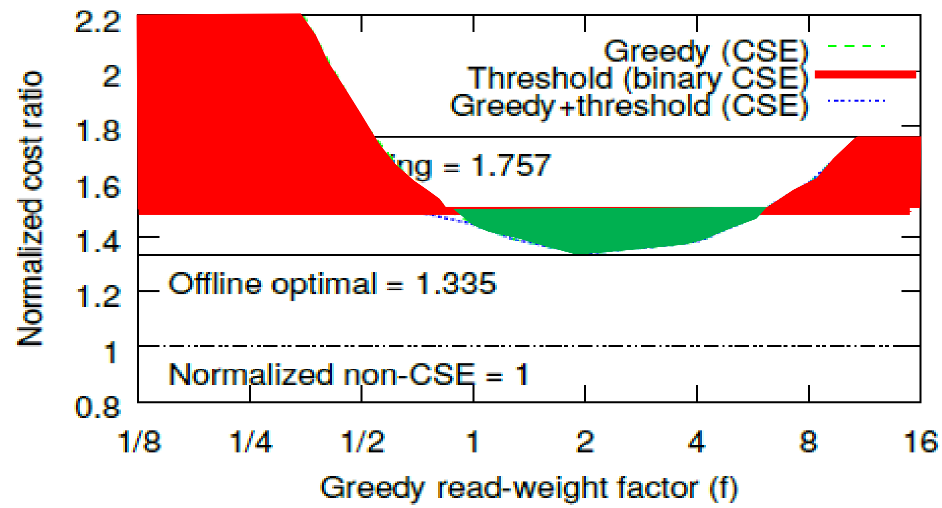


(a) Greedy read-weight factor

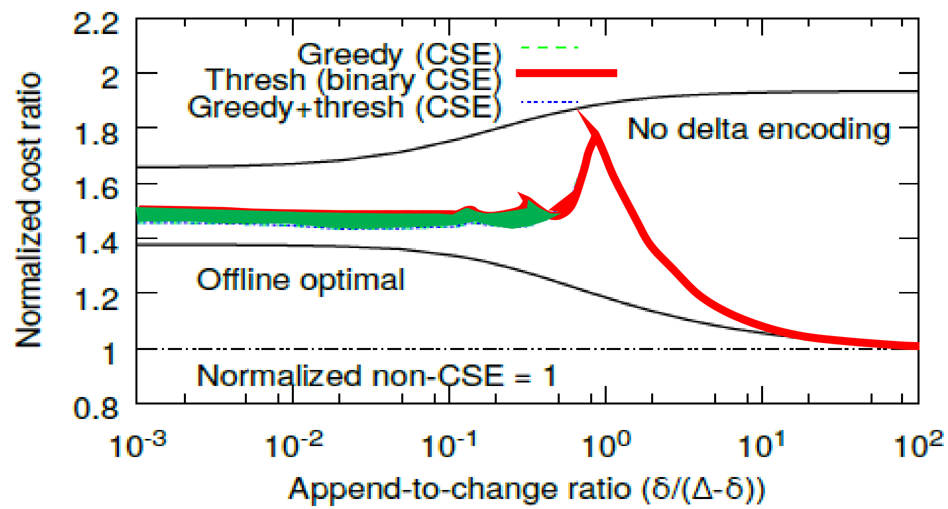


(b) Append-to-change ratio

More advanced multi-step policies



(a) Greedy read-weight factor



(b) Append-to-change ratio

- Simple binary threshold policy still does well
- Greedy can do slightly better in a few cases, but also worse

Conclusions

Conclusions

Targeted experiments and a model-based analysis to

1. demonstrate the delta encoding problem associated with CSE
2. characterize the practical overheads associated with delta encoding
3. determine the potential room for further improvements.

Our experiments demonstrate

- overheads due to CSEs not being able to decode delta encoding messages
- significant differences in the effectiveness in how delta encoding is implemented
- much room for improvements

A simple cost model is then developed that captures multi-device scenarios

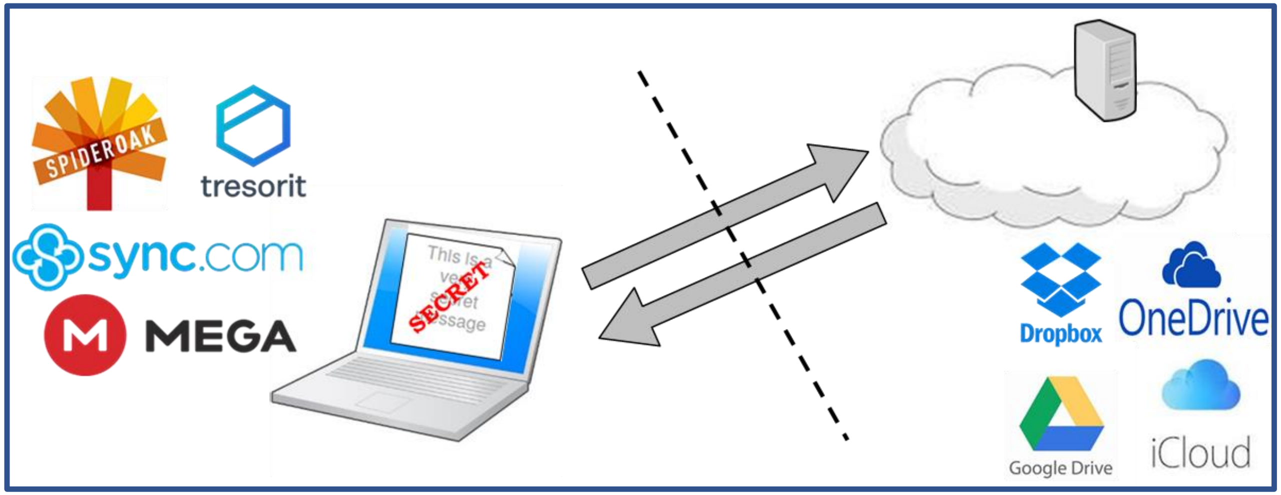
- worst-case bounds of the delta encoding penalty associated with CSEs
- characterization of the CSE overheads observed

Overall, the results show that

- costs of CSEs can be worst-case bounded by a factor 2 of the best non-CSEs
- with average differences significantly smaller for wide range of other workloads

Results demonstrate significant cost saving opportunities not yet used by current CSEs

Thanks for listening!



Delta Encoding Overhead Analysis of Cloud Storage Systems using Client-side Encryption

Eric Henziger and Niklas Carlsson

LINKÖPING UNIVERSITY

Niklas Carlsson (niklas.carlsson@liu.se)