NCCloud: Applying Network Coding for the Storage Repair in a Cloud-of-Clouds

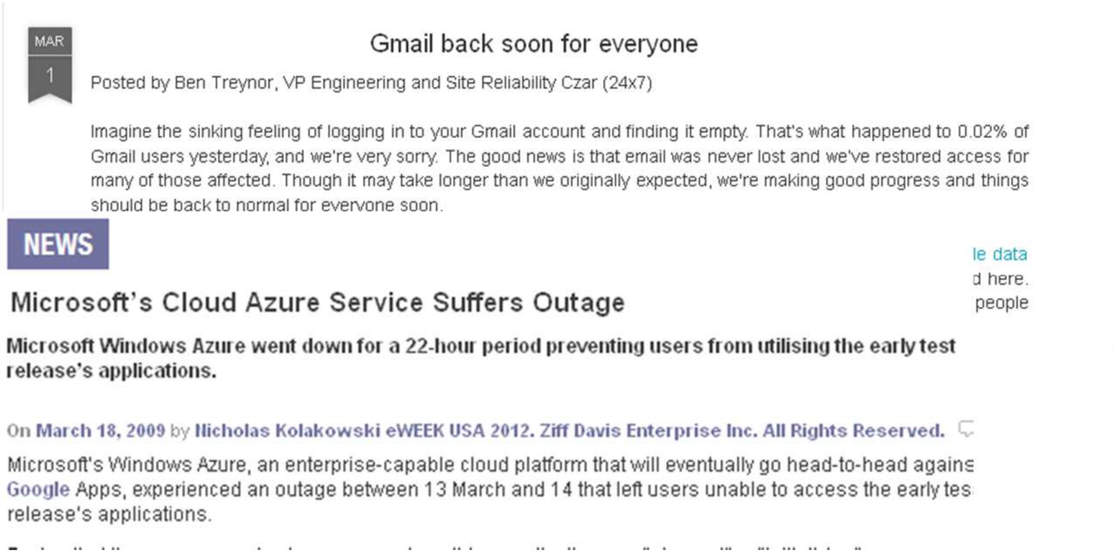
Yuchong Hu[1], Henry C. H. Chen[1],
Patrick P. C. Lee[1], Yang Tang[2]

[1]The Chinese University of Hong Kong
[2]Columbia University

FAST'12

Cloud Storage

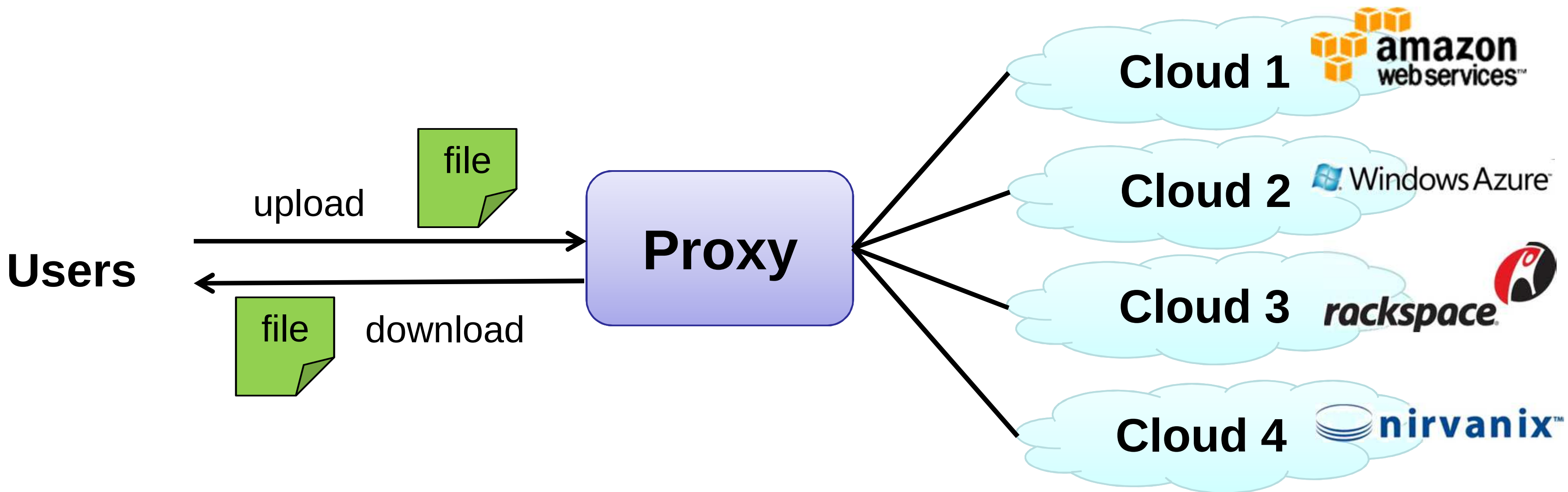
- ➢ Cloud storage is an emerging service model for remote backup and data synchronization
- ➢ Single-cloud storage raises concerns:
  - Cloud outage



  - Vendor lock-ins [Abu-Libdeh et al., SOCC'10]
    - Costly to switch cloud providers

Multiple-Cloud Storage

- Solution: **multiple-cloud storage**
  - Deploy a proxy between users and multiple clouds
  - Stripe data across multiple clouds

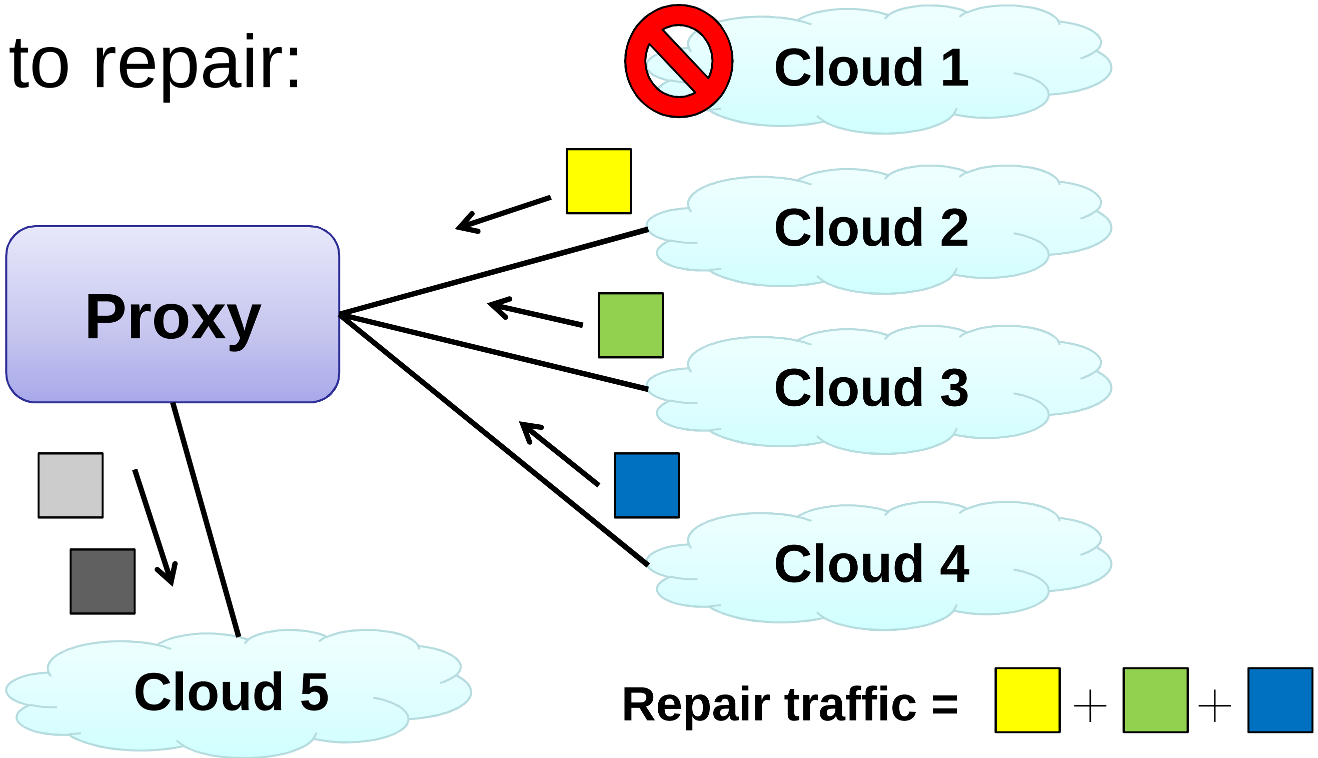


(n,k) MDS code: Any k out of n storage nodes (clouds) can rebuild original file.

e.g., RAID-5: $k = n - 1$; RAID-6: $k = n - 2$

Repairing a Failed Cloud

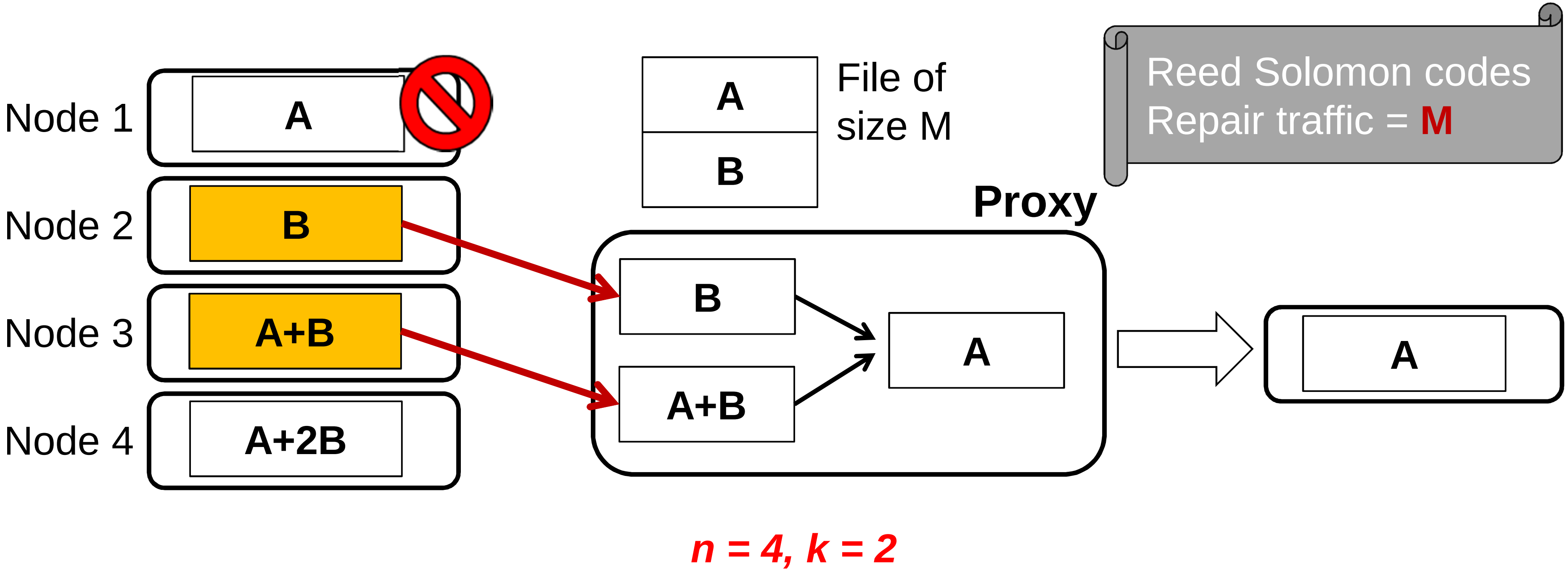
➢ How to repair:



➢ Goal: ***minimize repair traffic***

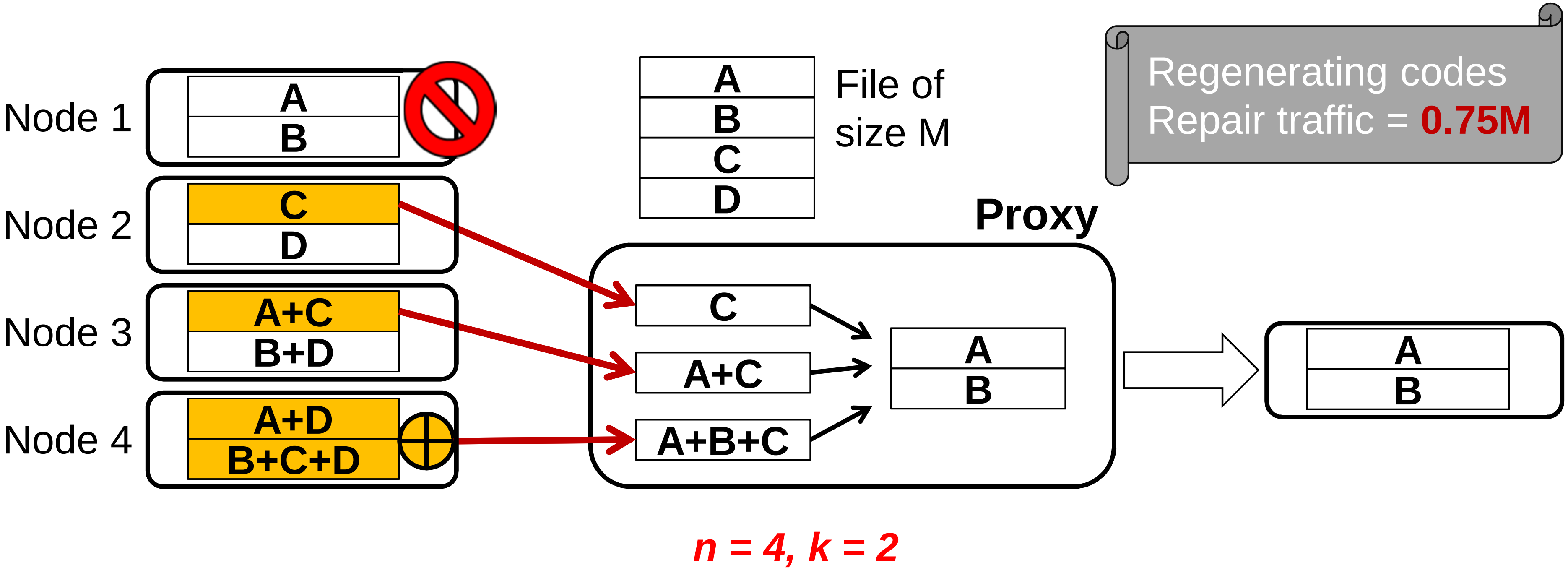
- Repair traffic: amount of data read from surviving clouds
- Hence minimize monetary cost due to data migration

Reed Solomon Codes



- Conventional repair:
  - Repair whole file and reconstruct data in new node

Regenerating Codes [Dimakis et al.'10]



- Repair in regenerating codes:
  - Downloads one chunk from each node (instead of whole file)
  - Repair traffic: save 25% for (n=4,k=2), while same storage size
  - Using network coding: encode chunks in storage nodes

Related Work

- Theoretical analysis
  - Regenerating codes [Dimakis et al. ’10] exploit the optimal trade-off between storage and repair traffic.
- Empirical studies
  - e.g., [Gkantsidis & Rodriguez ’05], [Dunimuco & Biersack ’09], [Martalo et al. ’11]
    - Evaluate random linear codes
    - Based on simulations
- Multiple cloud storage
  - e.g., HAIL [Bowers et al. ’09], RACS [Abu-Libdeh et al. ’10], DEPSKY [Bessani et al. ’11]
  - Based on erasure codes

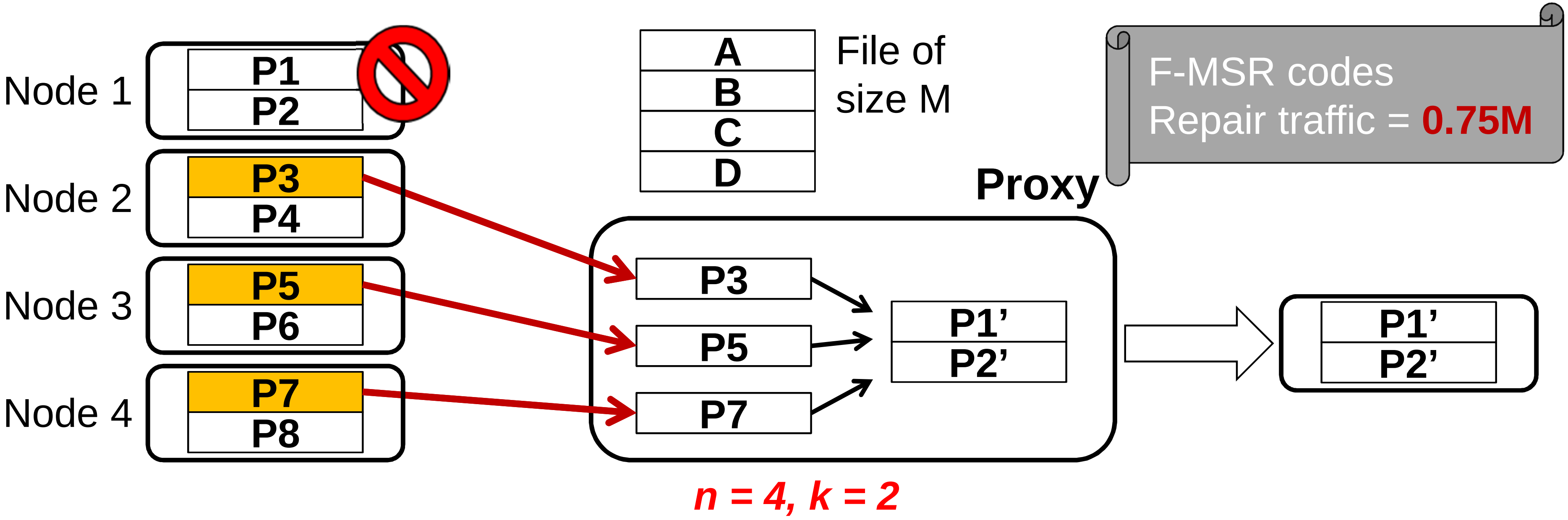
Challenges

- Implementation of regenerating codes in multiple cloud storage:
  - Can we eliminate encoding/decoding operations in storage nodes (clouds)?
    - Only standard read/write interfaces would suffice
  - Can we support basic upload/download operations with regenerating codes?
  - Can we support the repair function with regenerating codes?

Our Work

- Build **NCCloud**, a proxy-based storage system that applies regenerating codes in multiple-cloud storage
- Design goals:
  - Propose an implementable design of functional minimum-storage regenerating (F-MSR) code
  - Support basic read/write operations and the repair function
  - Preserve storage overhead as in MDS codes, while reducing repair traffic
- Implement and evaluate NCCloud in real storage setting
  - focus on double-fault tolerance (k = n-2)
  - focus on single-fault recovery
  - built on FUSE

F-MSR: Key Idea

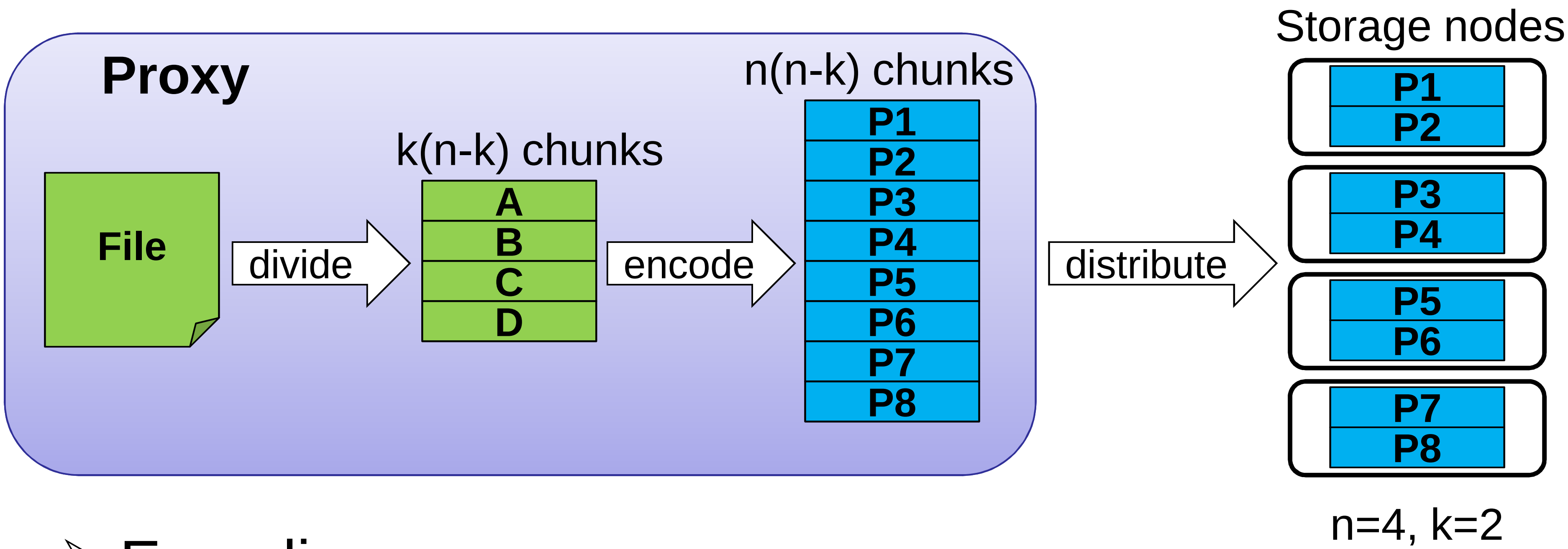


- Code chunk P_i = linear combination of original data chunks
- Repair in F-MSR:
  - Download one code chunk from each surviving node
  - Reconstruct new code chunks (via random linear combination) in new node

F-MSR: Key Idea

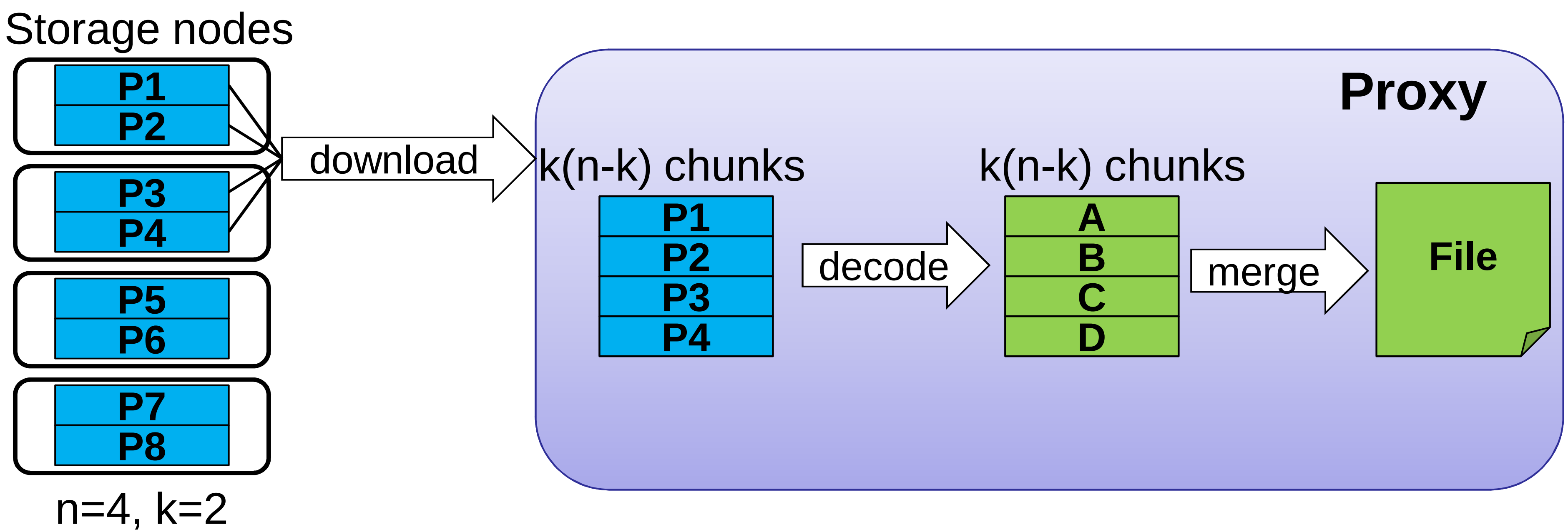
- F-MSR: **non-systematic**
  - Doesn't keep original data as in systematic codes
  - Stores only linearly combined code chunks
    - while maintaining MDS property
  - Suitable for rarely-read long-term archival
- With (non-systematic) F-MSR,
  - Eliminate need of encoding/decoding in clouds
  - Keep the benefits of network codes in storage repair
  - For k = n-2 (double-fault tolerance)
    - n = 4: repair traffic saved by 25%
    - For very large n: repair traffic saved by almost 50%

NCCloud: Upload



- Encoding process:
  - $\mathbf{P_i} = \mathbf{ECV_i} \times [\mathbf{A,B,C,D}]^T$
    - **ECV$_i$:** encoding coefficient vector of **P$_i$**
    - Arithmetic operations in GF(2^8)
  - $\mathbf{EM} = [\mathbf{ECV_1}, \mathbf{ECV_2}, \ldots, \mathbf{ECV_n}]^T$
    - **EM**: encoding matrix is replicated to all nodes as metadata

NCCloud: Download

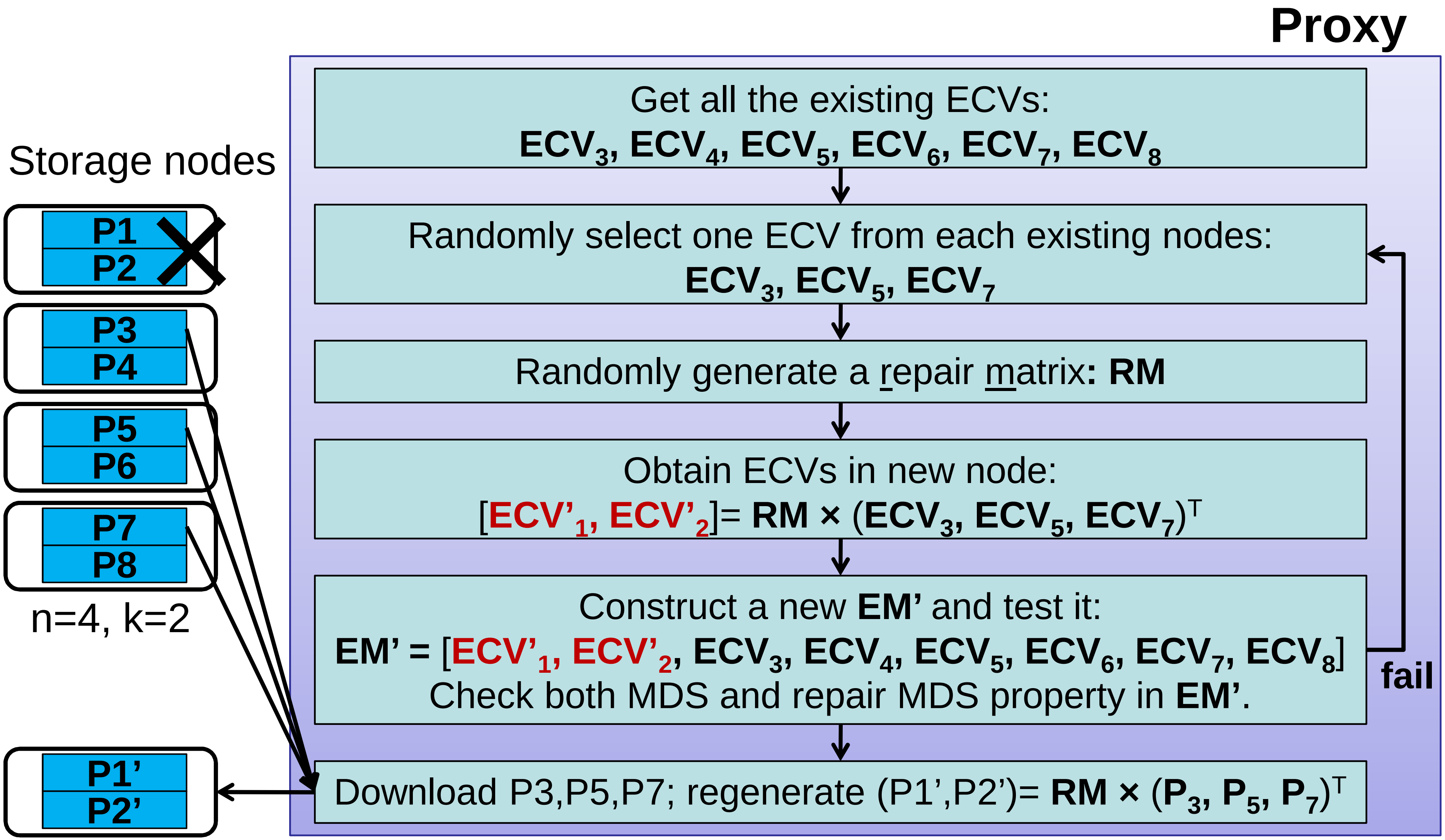


- Decoding process:
  - $[\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}]^T = \mathbf{EM}^{-1} \times [\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4]^T$
    - Download all the chunks from any k of n clouds
    - Multiply inverted encoding matrix with downloaded chunks

NCCloud: Iterative Repair

- Repair: generate random linear combinations of chunks
- How to keep iterative single-failure repairs sustainable?
  - i.e., how to ensure new code chunks don't break MDS property?
- Solution: **two-phase checking**
  - MDS property check
    - Current repair maintains MDS property
  - Repair MDS property check
    - Next repair for any possible failure maintains MDS property
- Simulations show the importance of two-phase checking over MDS property check only
  - See paper for details

NCCloud: Iterative Repair



Cost Analysis

	S3	RS	Azure
Storage (per GB)	$0.14	$0.15	$0.15
Data transfer in (per GB)	free	free	free
Data transfer out (per GB)	$0.12	$0.18	$0.15
PUT,POST (per 10K requests)	$0.10	free	$0.01
GET (per 10K requests)	$0.01	free	$0.01

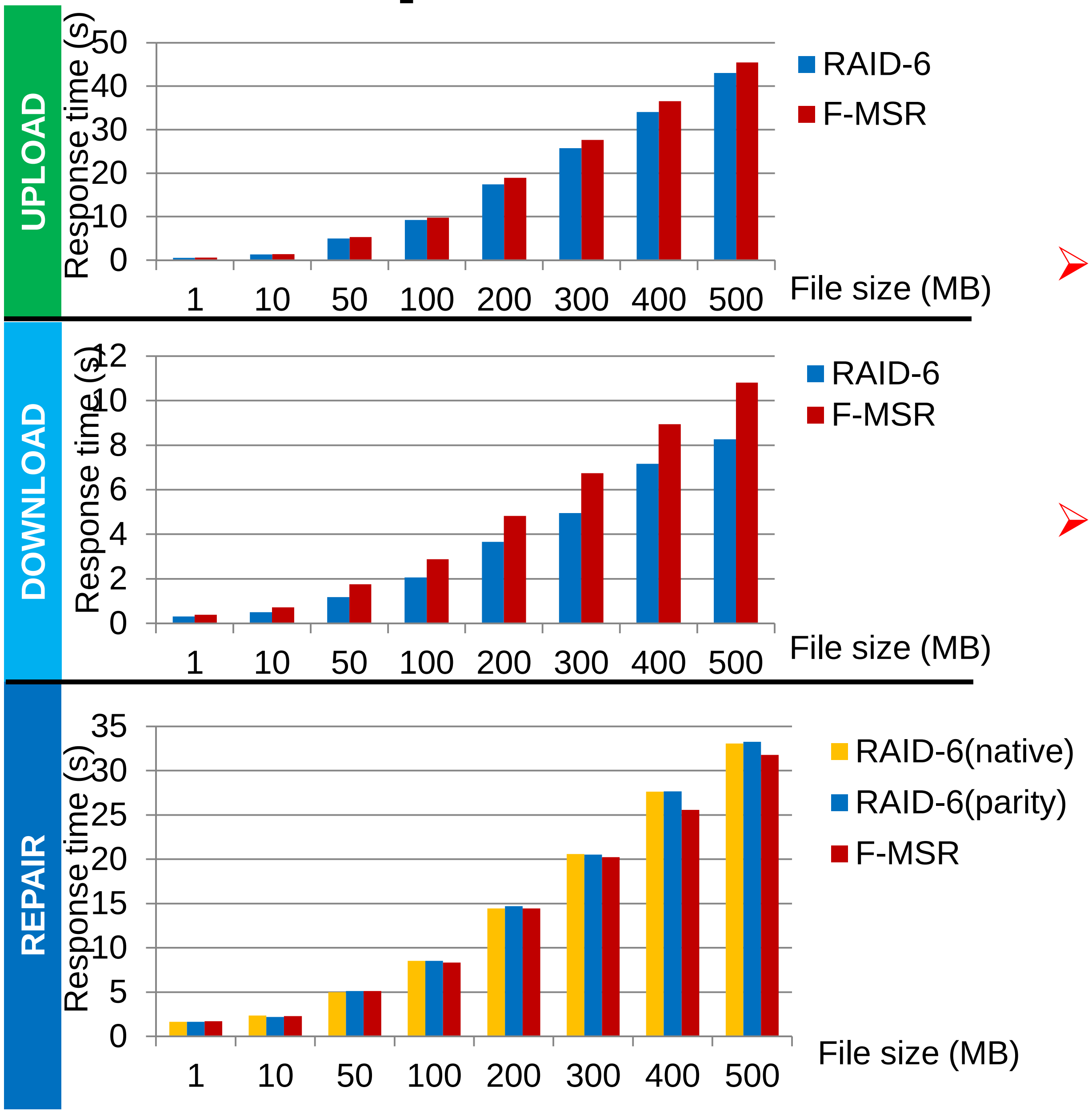
Monthly price plan as of Sep 2011

- Repair traffic cost
  - F-MSR saves 25% (for n = 4) compared to conventional repair
- Metadata of F-MSR
  - Metadata size = 160B; file size = several MBs
- Overhead due to GET requests during repair
  - Assuming S3 plan in Sep 2011, n = 4, k = 2, file size = 4MB
  - Conventional repair: 0.427%
  - F-MSR repair: 0.854%

Experiments

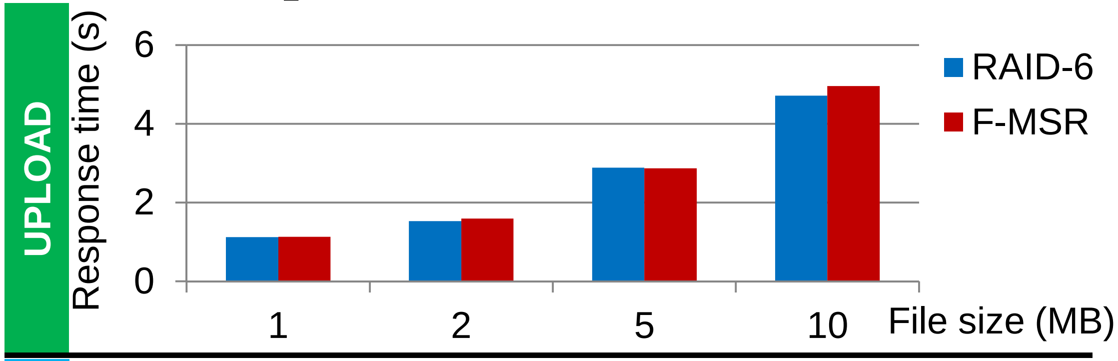
- NCCloud deployment
  - Single machine connected to a cloud-of-clouds
  - n = 4, k = 2
- Coding schemes
  - Reed-Solomon-based RAID-6 vs. F-MSR
- Metric
  - Response time
- Cloud environments:
  - Local cloud: OpenStack Swift
  - Commercial cloud: multiple containers in Azure

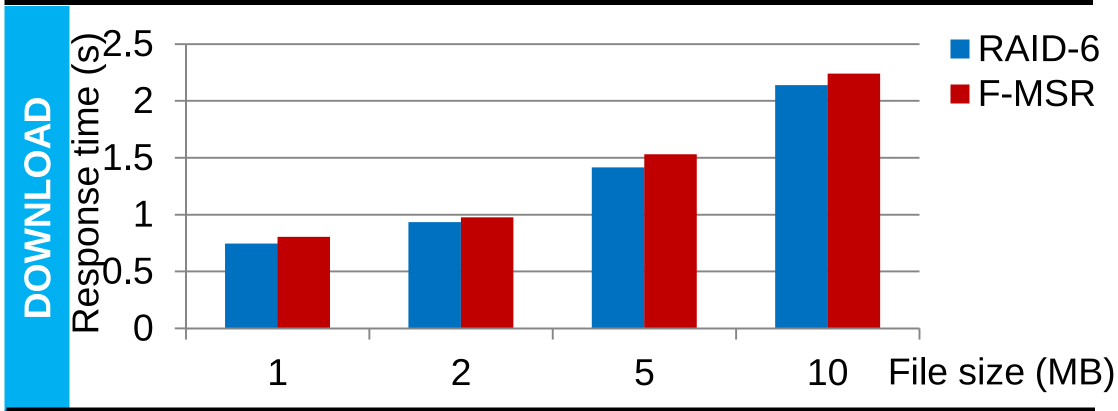
Response time: Local Cloud

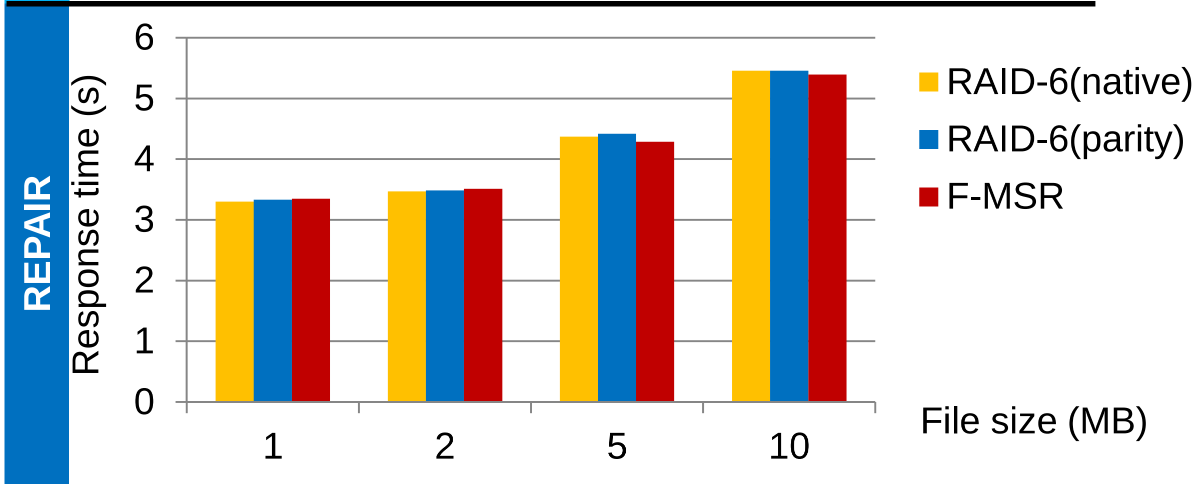


- F-MSR has higher response time due to encoding/decoding overhead
- F-MSR has slightly less response time in repair, due to less data download

Response time: Commercial Cloud







- No distinct response time difference, as network fluctuations play a bigger role in actual response time

Conclusions

- Propose an implementable design of **F-MSR**:
  - Preserve storage cost, but use less repair traffic
- Build **NCCloud**, which realizes F-MSR
- Source code:
  - **http://ansrlab.cse.cuhk.edu.hk/software/nccloud/**