Space Bounds for Reliable Storage: Fundamental Limits of Coding

Alexander Spiegelman  Yuval Cassuto  Gregory Chockler  Idit Keidar

ACM PODC 2016
Codes Meet Distributed Systems
Fault-Tolerant Distributed Storage
Fault-Tolerant Distributed Storage Model

- $\infty$ clients (all can fail)
- $f$ can fail (crash)
- $D$ bits
- $n$ servers
- Asynchronous
Storage Blow-Up

• Demands are growing exponentially
• Replication is costly
• Erasure coding can help
  [Goodson et al. DSN 2004]
  [Aguilera et al. DSN 2005]
  [Cachin and Tessaro DSN 2006]
  [Hendricks et al. SOSP 2007]
  [Dutta et al. DISC 2008]
  [Cadambe et al. NCA 2014]
• But ... within limits
k-of-n Erasure Codes

\[ D \xrightarrow{\text{encode}} D/k \xrightarrow{} n \]
k-of-n Erasure Codes

\[ D \xrightarrow{\text{encode}} \frac{D}{k} \xrightarrow{\text{decode}} k \]

\[ D \xrightarrow{\text{encode}} n \]
Why Codes?

To tolerate one failure

• With replication
• With erasure codes
Distributed Storage: Space Bounds

Replication

\[ O(D \cdot f) \text{ bits} \]

Lower bound

\[ \Omega(D \cdot \min(f, c)) \text{ bits} \]

Best-of-both algorithm

\[ O(D \cdot \min(f, c)) \text{ bits} \]

Coding

\[ O(D \cdot c) \text{ bits with } c \text{ concurrent writes} \]
Reliable Storage Example

• \( n = 2f + k \)

\[ \begin{align*}
  n &= 4 \\
  f &= 1 \\
  k &= 2
\end{align*} \]
Write
Write

Generate timestamp
Write

Can’t wait forever
Write

Wait for $n-f$ replies
Read
Read
Read

Wait for $n-f$ replies
What About Concurrency?
Write
Write
Write
Write

Overwrite?
Write

Overwrite?
Suppose yes, if timestamp is bigger
Write
Write

Read
Write

Read
No written value can be restored!
What About Replication?
Write

No problem!

Read
Back to Coding ...
Overwrite? No!
Overwrite Yellow?
Read cannot be restored!
What can be overwritten? Nothing!
Ok, so the standard algorithm can use loads of storage

But...
Inherent
Inherent

For asynchronous algorithms that store coded data
Distributed Storage: Space Bounds

Lower bound \( \Omega(D \cdot \min(f, c)) \)

Best-of-both algorithm \( O(D \cdot \min(f, c)) \)

Replication: \( O(D \cdot f) \)

Coding: \( O(D \cdot c) \) with \( c \) concurrent writes
But First: More About The Model

• Black-box encoding

• Arbitrary encoding scheme

• The storage holds:
  – Coded blocks
  – Unbounded data-independent meta-data
Observation

• Every data bit in the storage can be associated with a unique write operation
  – Given our storage model (black-box encoding)
  – Formalities in the paper
• Storage is measured in bits

do not count

meta-data

count
Theorem

• Every reliable storage solution
  – Ensuring progress (lock-freedom) & consistency (read returns last or concurrent write)
  – Tolerating $f$ failures
  – Allowing $c$ concurrent writes
  – Storing values from a domain of size $2^D$

• Needs to store $\Omega(D \cdot \min(f,c))$ bits
  – At some point
Proof Steps

• Pigeonhole: need $D = \log_2 |V|$ bits associated with some write operation to read a value
Tracking Sets
Tracking Sets

$C^+(t)$

writes that store more than $D - \ell$ bits at time $t$
Tracking Sets

- $C^+(t)$
  - Servers that store at least $\ell$ bits at time $t$

- $F(t)$
  - Writes that store more than $D - \ell$ bits at time $t$

$D - \ell + 1$
Storage Size

\[ C^+(t) \]

\[ \text{Storage Size} = D - \ell + 1 + \ell / 2 \]
Storage Size

\[ C^+(t) \]

\[ \text{At least } (D-\ell+1) \cdot |C^+(t)| \text{ bits} \]
Storage Size

At least \((D - \ell + 1) \cdot |C^+(t)|\) bits

At least \(\ell \cdot |F(t)|\) bits

\[C^+(t)\]

\[F(t)\]
Adversary Ad

• We’ll define a particular adversary structure
  – Controls scheduling
  – Prevents progress
  – Blows up the storage
Defining Adversary Ad

- $C^+(t)$
- delay $C^+(t)$
- freeze $F(t)$
- $D - \ell + 1$
- $D$
Implications of Ad

• F only grows  
  – Servers in F are “frozen”

• Writes can move out of C⁺  
  – If their blocks are overwritten

• We will next show that no client can complete a write operation
Observation

Every set of n-f servers must store D bits of some pending write for a write to return
Observation

Otherwise
Observation

Read

No value can be restored!
Lemma

• With Ad, all servers in $N \setminus F(t)$ together store less than D bits for each write
  – Proof in the paper
Lemma Proof

With Ad, if $|F(t)| \leq f$, then there are $n - f$ servers from which no value of a pending write can be read.

- If RMW writes $\ell$ bits to $S$, $S$ is added to $F$ and $v$ cannot be restored without servers in $F$.
- Write($v$) is in $C$ and $\ell$ bits are missing.
- Write($v$) RMW to server $S$ responds.
- Assume $v$ can be read.

Otherwise, at least one bit is still missing.

Contradiction!
Corollary : Lemma + Observation

• With Ad, for any time t, if $|F(t)| \leq f$, then no write completes
Storage Size Recalled

\[ C^+(t) \]

At least \((D-\ell+1) \cdot |C^+(t)|\) bits

At least \(\ell \cdot |F(t)|\) bits

\[ D - \ell + 1 \]

\[ \ell/2 \]

\[ \ell/2 \]
Theorem Proof

• Build run r using $Ad$
  – Have c clients invoke writes

• Three possible cases:
  – $|F(t)| > f$ at some point in r
    $\Rightarrow$ storage cost $\ell(f+1)$ bits
  – $|C^+(t)| = c$ at some point in r
    $\Rightarrow$ storage cost $(D-\ell+1)c$ bits
  – None of the above
    $\Rightarrow$ by corollary, no write returns
    $\Rightarrow$ we will show this is impossible

• By setting $\ell = D/2$, we get $\Omega(D \cdot \min(f,c))$
Ad is Not Fair!

- Operations on servers in F(t) never take effect from time t onward
- Operations by clients that remain in C⁺ from some point onward never take effect
Constructing a Fair Run (Sketch)

• Run $r$ with Ad: assume $|F(t)| \leq f$ and $|C^+(t)| < c$
  – No write completes in $r$
• Build $r'$: kill all the servers in $F$ and clients permanently in $C^+$
• $r'$ is fair with at least one correct process
  – By progress, some write completes in $r'$
• $r$ and $r'$ are indistinguishable to all correct clients
  – Therefore, some write completes in $r$
Constructing a Fair Run (Sketch)

- Run \( r \) with Ad: assume \( |F(t)| \leq f \) and \( |C^+(t)| < c \)
  - No write completes in \( r \)

Contradiction

- Therefore, some write completes in \( r \)
Constructing a Fair Run (Sketch)

- Run $r$ with Ad: assume $|F(t)| \leq f$ and $|C^+(t)| < c$
  - No write completes in $r$

- Therefore, some write completes in $r$
Theorem Proof

• Build run $r$ using $Ad$
  – Have $c$ clients invoke writes
• Three Two possible cases:
  – $|F(t)| > f$ at some point in $r$
    ⇒ storage cost $\ell(f+1)$ bits
  – $|C^+(t)| = c$ at some point in $r$
    ⇒ storage cost $(D-\ell+1)c$ bits
  – None of the above
• By setting $\ell = D/2$, we get $\Omega(D \cdot \min(f,c))$
Distributed Storage: Space Bounds

Lower bound
\( \Omega(D \cdot \min(f, c)) \)

Best-of-both algorithm
\( O(D \cdot \min(f, c)) \)

Replication
\( O(D \cdot f) \) bits

Coding
\( O(D \cdot c) \) with \( c \) concurrent writes
Wrap Up

• We proved a fundamental limit of coding
  – $\Omega(D \cdot \min(f,c))$ bits
  – Replication is the best solution under high concurrency

Why not enjoy both worlds?
Adaptive Algorithm

Replication
storage cost: \(nD\)

Coding with \(k,n = O(f)\)
storage cost: \((c+1)(D/k)n\)

We combine both approaches
Storage cost:
\[
\min(2nD, (c+1)(D/k)n) = O(D \cdot \min(f, c))
\]
Adaptive Algorithm

Replication
storage cost: \( nD \)

Coding with \( k, n = O(f) \)
storage cost: \( (c+1)(D/k)n \)

Storage cost:
\[
\text{min}(2nD, (c+1)(D/k)n) = O(D \cdot \text{min}(f,c))
\]

Details in the paper
Related Work

- [Cadambe, Wang, Lynch, PODC 2016]
- Similar bound
  - Only for wait-free storage (ours is for lock-free)
  - Different (incomparable) black-box assumptions
  - Different proof technique

**Future:** find unique minimal set of assumptions
Thank You!