Information-Theoretic Lower Bounds for Shared Memory Emulation

Zhiying Wang
University of California, Irvine
Joint with Viveck R. Cadambe (PSU), Nancy Lynch (MIT)
Distributed Algorithms

• Algorithms that are supposed to work in distributed networks, or on multiprocessors
• ~30 years research

• Applications
  – Machine learning
  – Networking
  – Multiprocessor programming
Timing

• Synchronous
  – Processes communicate every round
  – Requires “clock”

• Asynchronous: take action at any time
Changing Information in Distributed System

- Modern key value stores - Amazon Dynamo DB, Couch DB, Apache Cassandra DB, Google Spanner, Voldemort DB ...
- Used for transactions, reservation systems, multi-player gaming, social networks, news feeds, distributed computing tasks...
- High frequency trading
- Theoretical: relates to shared memory, connects two fundamental communication models of asynchronous systems
A Motivating Example

- **N** servers
- Tolerate **f** server failures
- A message generated every time slot
- **Channel:** delay between 0 and **T-1**
- **Encode:** a function of its own messages
- **Decode:** from any **N-f** servers get the latest common message or something later
A Motivating Example

- \( N \) servers
- Tolerate \( f \) server failures
- A message generated every time slot
- **Channel:** delay between 0 and \( T-1 \), say \( T=2 \)
- **Encode:** a function of its own messages
- **Decode:** from any \( N-f \) servers get the latest common message or something later
- At time slot \( t \)

<table>
<thead>
<tr>
<th>Server 1</th>
<th>Server 2</th>
<th>Server 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1,..., ( t-1 ), ( t )</td>
<td>0, 1,..., ( t-1 ), ( t )</td>
<td>0, 1,..., ( t-1 ), ( t )</td>
</tr>
</tbody>
</table>
A Motivating Example

- $T=2$
- At time $t$, only need to consider $t-1$ and $t$
- Call them message 1 and message 2
- In general, $T$ represents the concurrency level
- Q: What is the storage cost?
Solution 1: Replication

• Storage size = $\log|V|$
Solution 2: Static Coding

- Recover the message from any 3 pieces
- Every server only stores $\log|V| \times \frac{1}{3}$!
- Examples: Facebook, Windows Azure...

Original message

| x | y | z |

Coded message

| x | y | z | x+9y+4z | x+2y+3z | x+3y+7z | x+8y+z |

9
Solution 2: Static Coding

- However, need to keep history (Msg 1)
- (Worst-case) storage size = \( \log|V| \times \frac{2}{3} \)

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\
\end{array}
\]

\( N=7, \ N-f=3, \ T=2 \)
Our Solution: Multi-Version Code

- Storage size = $\log|V| \times \frac{1}{2}$

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\
\end{array}
\]

N=7, N-f=3, T=2

More Generally

- Messages generated (write invoked) at any time point
- Channel delays can be arbitrary and unbounded
- Can design write and read protocols
- Multiple write and read clients
Timing

• Synchronous
  – Processes communicate every round
  – Requires “clock”

• Asynchronous: take action at any time
  – Message passing model
  – Shared memory model
Message Passing Model

reader

writer

shared memory

servers

message

message

message

message

message

message

message

reader
Shared Memory Model

reader

writer

shared memory

reader

message

message

message
Emulate Shared Memory In Message Passing Model

Any algorithm for shared memory model => an algorithm for message passing model.
In This Talk

Information-theoretic lower bounds on the storage cost of the shared memory emulation algorithms

• Problem settings
• First lower bound for any algorithm
• Second lower bound for a restricted class of algorithms
Emulation of Shared Memory in Message-passing Networks

• Setup
  – Distributed, unreliable, asynchronous
  – Processes: \( N \) servers, any number of write and read clients
  – Failures: up to \( f \) servers, arbitrary number of clients can crash
  – Concurrent access allowed

• Requirements
  – Atomicity
  – Liveness
Atomicity

[Lamport 86]
Liveness
Challenges

• Asynchronous communication
  – A process has no global view
  – Atomicity requires a global ordering

• Unbounded delay
  – A very long delay?
  – A failure?

• Failures
What should the client do?

• Write to 1 server?
What should the client do?

• Write to all servers?
What should the client do?

- Write to or read from a “quorum”
  - A quorum = any subset of size $N-f$

$N=3, f=1$
What should the client do?

• Write to or read from a “quorum”
  – A quorum = any subset of size $N-f$

$N=3$, $f=1$
Other Concerns

• Add “tags” to messages
• Readers “write back”
Replication Based Algorithm

• Server stores the latest message it received
  – According to the tag
• Can recover what has been written
  – When the write and read quorums have an overlap

[ABD, Attiya, Bar-Noy, Dolev, 1990]
Improve by Coding

• Coding based algorithms
  – [HGR 04], [AJL 05], [CT 05, 06], [DGL 08], [DKLMSV 13], [ACDV 14], [CLMP 14]
  – (Worst-case) Storage cost is proportional to number of concurrent active writes at any time point, $v$
    – Per-server storage cost $\geq v \frac{\log_2 |\mathcal{V}|}{N-f}$

• Replication based algorithms [ABD]
  – Per-server storage cost $= \log_2 |\mathcal{V}|$
Storage Cost Independent of \( v \)?

Writer 1 deliver some pieces
Msg 1 = \((x,y,z)\)

Writer 2 deliver some pieces
Msg 2 = \((x',y',z')\)

Both writers deliver all pieces

Two writers decide to store msg 1
Works even if writer 1 fails
Baseline Lower Bound

Theorem. \( \text{Per-server storage-cost} \geq \frac{1}{N-f} \log_2 |\mathcal{V}| \)

- Analogous to Singleton bound in coding theory

\[ x \quad y \quad z \]
\[ x \quad y \quad z \quad x+y+z \]

\( N = 4, \ f=1 \)
Summary of Results

ABD algorithm

Baseline lower bound

Erasure coding based algorithms

First lower bound

Number of concurrent writes

Storage Cost

[Cadambe, W, Lynch, 2016]
Summary of Results

Storage Cost

ABD algorithm

Erasure coding based algorithms

Second lower bound*

Baseline lower bound

First lower bound

Number of concurrent writes

[Cadambe, W, Lynch, 2016]
First Lower Bound

Theorem. \( \text{Per-server storage-cost}(|\mathcal{V}|) \geq \frac{2}{N-f+2} \log_2 |\mathcal{V}| - o(1) \)

Holds even for single-writer single-reader case
First Lower Bound

Theorem. \( \text{Per-server storage-cost}(|\mathcal{V}|) \geq \frac{2}{N-2f} \log_2 |\mathcal{V}| - o(1) \)
First Lower Bound

Theorem.  \( \text{Per-server storage-cost}(|\mathcal{V}|) \geq \frac{2}{N-f+2} \log_2 |\mathcal{V}| - o(1) \)

Prove by a counting argument.
Both values can be returned from the two points.
These points differ in at most 2 server states.
A total of \(N-f+2\) server states.
Second Lower Bound

Theorem. \[ \text{Per-server storage-cost}(|\mathcal{V}|) \geq \frac{\nu^*}{N - f + \nu^* - 1} \log_2 |\mathcal{V}| - o(\log(|\mathcal{V}|)) \]

\[ \nu^* = \min(\nu, f), \]

under certain assumptions

Informally, our assumptions prevent interactions
Writers send the value only once
Assumptions on Write Protocol

• Writer actions are “black-box actions”
  – oblivious to the actual value

• Write protocol operates in phases

• For every write operation, there is at most one phase where value-dependent messages are sent

• Most existing algorithms fit above assumptions
Second Lower Bound

Theorem. \[ \text{Per-server storage-cost}(|\mathcal{V}|) \geq \frac{\nu^*}{N_{f+\nu^*-1}} \log_2 |\mathcal{V}| - o(\log(|\mathcal{V}|)) \]

\[ \nu^* = \min(\nu, f), \]

Prove for the case of \( \nu < f \).
One Time Point

• v writers. Every writer has one value

• If one writer delivers value-dependent msgs to all alive servers, and any value-independent msgs can be delivered,
  – Liveness: that write operation should return
  – Atomicity: reader should return something
Bad-case Server States

- Lemma. There exist the following “worst-case” assignments of $N-f+v-1$ server states such that they contain enough information about all the $v$ messages.
- Consequence of liveness and atomicity.
- In the bad-case server states, the average storage cost per server is at least $\frac{v}{(N-f+v-1)}$.

![Diagram showing the distribution of messages across servers]
Construct the Bad-case Execution

- Want to create an **execution** of the algorithm that has a **time point** corresponding to the bad-case server states
- Let writers act until the phase to send value-dependent messages
- Let writers send the value-dependent messages, **keep them in the channel**
- Let channel deliver according to the worst-case server states
Construct the Worst-case Execution

- Want to create an execution of the algorithm that has a **time point** corresponding to the bad-case server states
Second Lower Bound

Theorem. \( \text{Per-server storage-cost}(|V|) \geq \frac{\nu^*}{N_f - f + \nu^* - 1} \log_2 |V| - o(\log(|V|)) \)

\[ \nu^* = \min(\nu, f), \]

Proved for the case of \( \nu < f \).
Open Question: Beat the $O(\min(v,f))$ Bound?

• Second bound
  – Assumption: write once

• Similar bound in [Spiegelman, Cassuto, Chockler, Keidar, 2016]
  – Assumption: no coding across messages

• In order to get storage cost $< O(\min(v,f))$
  – Need to write multiple times
  – And need to code across messages

• Maybe not possible to beat!
Other Future Directions

• Improve performance of distributed algorithms with coding
  – Dynamic network configuration
  – Distributed optimization
Thank you!