Passing Messages while Sharing Memory

Marcos K. Aguilera, Naama Ben-David, Irina Calciu, Rachid Guerraoui, Erez Petrank, and Sam Toueg
Two Models

Message Passing

Asynchronous processes and links;
Processes can crash

Data centers

Shared Memory

Multiprocessor computers
Two Models

Message Passing
- Consensus impossible with asynchrony + crash
- Consensus with randomization and partial synchrony
- Distributed graph algorithms

Shared Memory
- Consensus impossible with asynchrony + crash
- Consensus with randomization and atomic primitives
- Concurrent data structures

This work: The M&M model

Computers in data center

Processes in one machine
What do we gain by combining the two models?
Equivalence

ABD’95:

“Message passing and shared memory are equivalent!”

\[ \text{computationally} \]

“The models can solve the same set of problems”

What about tolerance to process crashes?

What about synchrony requirements?

What about efficient algorithms?
Our Results

Combining models allows *increased robustness*

- Introduce the *message-and-memory (M&M)* model

- **Fault tolerance**: solve *consensus* with up to *n-1* failures, lower bound, depending on topology of graph

- **Synchrony**: solve eventual *leader election* with one timely process, no timely edges. Matching synchrony lower bound
The M&M model

- **Asynchronous** network of \( n \) processes with up to \( f \) crash failures
- Fully-connected message passing network: \textit{nodes}=procs, \textit{edges}=links

- **Each node owns a piece of memory**
- Shared memory graph, \( G_{SM} = (V, E) \), defined on nodes
- Nodes \( u \) and \( v \) can access each other’s memory iff \( (u,v) \in E \)
- Processes may crash, but their \textit{memory remains accessible}
Our Results

Combining models allows increased robustness

• Introduce the message-and-memory (M&M) model

• **Fault tolerance**: solve consensus with up to \( n-1 \) failures, lower bound, depending on topology of graph

• **Synchrony**: solve eventual leader election with one timely process, no timely edges. Matching synchrony lower bound
Fault Tolerance: Consensus

All processes must agree on the same value

*Message Passing:* Cannot solve consensus with more than \( \lceil n/2 \rceil - 1 \) failures

*Shared Memory:* Can solve consensus with up to \( n-1 \) failures

Goal: Tolerate \( f > n/2 \) failures when solving consensus in M&M network
Fault Tolerance: Take 1

Idea: Connect all nodes over shared memory!

Now we can run any shared memory algorithm

Fault tolerance for consensus up from $\lceil n/2 \rceil - 1$ to $n-1$

... max degree is $n-1$

In real systems, infeasible to share memory with many processes

Goal: Keep max degree of $G_{SM}$ low

Can we do better?
M&M Consensus

Idea: Use shared memory to speak for your neighbors

Instead of sending just your message, agree with each neighbor using *shared memory consensus*, then send a *list of messages*.
How Much Do We Gain?

Depends on the shared memory graph $G_{SM}$

- More specifically, the number of *neighbors of correct* processes

Adversary chooses the set of correct processes

Want graphs with the following property:

*All sets of at least n-f processes have many neighbors*
Expander Graphs

Well studied class of graphs

Let $G=(V, E)$ be an undirected graph.

1. The **vertex boundary** of a set $S \subset V$ is $\delta S = \{\text{Neighbors of } S\} \setminus S$.
2. The **vertex expansion ratio** of $G$, denoted $h(G)$, is defined as:

$$h(G) = \min_{S \text{ s.t. } |S| \leq |V|/2} \frac{|\delta S|}{|S|}$$

"G has high expansion!"

≡

"Every subset of the vertices has many neighbors!"
Putting it Together

- **Set of live processes** is $S \subset V$
- Adversary will pick $S$ to be the set with the least expansion

$G_{SM}$ *with high vertex expansion can tolerate more failures*

**Theorem**: If $G_{SM}$ has vertex expansion ratio $h$, then we can tolerate

$$f < \left(1 - \frac{1}{2 \cdot (1 + h)} \right) \cdot n$$

failures
Our Results

Combining models allows increased robustness

- Introduce the message-and-memory (M&M) model
  
- **Fault tolerance**: solve consensus with up to $n-1$ failures, lower bound, depending on topology of graph
  
- **Synchrony**: solve eventual leader election with one timely process, no timely edges. Matching synchrony lower bound
Everyone chooses some process to be the leader.

Eventually, everyone picks the *same alive leader*, and *never changes their mind again*.

Steady State
Leader Election: Background

- Only studied in message passing
- **Synchrony** is needed to solve leader election

**Timeliness:**

> p is *timely* if there is a *lower bound on its execution speed* relative to all other processes

- Quality measures of an algorithm:
  - How much synchrony is needed?
  - How much work is done in the steady state?
Leader Election: Results

All previous algorithms require *timeliness from links*, and *at least one message sent* to each process forever.

M&M algorithm requiring only *one timely process*, and no timely links. Only *one memory location written and read* in the steady state.

Unlike fault tolerance, problem is hard even with fully connected GSM.
Leader Election: Results

M&M algorithm requiring only *one timely process*, and no timely links. Only *one memory location* written and read in the steady state.

**Optimality Theorem:** In an M&M network with only *one timely process*, any leader election algorithm has a run in which the *leader writes* infinitely often.
Summary & Future

• Message-and-memory *(M&M)* model

• **Consensus:** tolerate more crashes

• **Leader election:** reduce synchrony

*New exciting model, many new questions!*