There Is More Consensus in Egalitarian Parliaments

Iulian Moraru, David Andersen, Michael Kaminsky

Carnegie Mellon University

Intel Labs
Fault tolerance $\Rightarrow$ Redundancy
State Machine Replication
State Machine Replication

*Execute the same commands in the same order*
State Machine Replication

Execute the same commands in the same order

1.9 km
Turn right onto Lewis Run Rd

1.5 km
Turn left onto Clairton Blvd

54.1 km
Turn right to merge onto US-119 S

5.7 km
Continue onto Uniontown Byp S

21.6 km
The destination is on your right
State Machine Replication

Execute the same commands in the same order

Paxos
State Machine Replication

Execute the same commands in the same order

Paxos

- No external failure detector required
State Machine Replication

Execute the same commands in the same order

Paxos

- No external failure detector required
- Fast fail-over (high availability)
Paxos is important in clusters

Chubby, Boxwood, SMARTER, ZooKeeper

• Synchronization
• Resource discovery
• Data replication

High throughput
High availability
Paxos is important in the wide-area

Spanner, Megastore

- Bring data closer to clients
- Tolerate datacenter outages

Low latency
Paxos is important in the wide-area Spanner, Megastore

- Bring data closer to clients
- Tolerate datacenter outages

Low latency
Paxos is important in the wide-area

Spanner, Megastore

• Bring data closer to clients
• Tolerate datacenter outages

Low latency

130ms

85ms
Paxos overview
Paxos overview

• Agreement protocol
Paxos overview

• Agreement protocol

• Tolerates F failures with 2F+1 replicas (optimal)
  • No external failure detector required
Paxos overview

• Agreement protocol

• Tolerates F failures with 2F+1 replicas (optimal)
  • No external failure detector required

• Replicas can fail by crashing (non-Byzantine)
Paxos overview

• Agreement protocol

• Tolerates F failures with 2F+1 replicas (optimal)
  • No external failure detector required

• Replicas can fail by crashing (non-Byzantine)

• Asynchronous communication
Using Paxos to order commands
Using Paxos to order commands

C

A

B

1  2  3  4  ...

Friday, November 8, 13
Using Paxos to order commands

A

B

C

1  2  3  4  ...

...
Using Paxos to order commands
Using Paxos to order commands

vote C
ack B

C

vote A
ack B

A

vote B

B

1  2  3  4  ...

Friday, November 8, 13
Using Paxos to order commands

C

A

B

1 2 3 4 ...

Friday, November 8, 13
Using Paxos to order commands

C

A

1
2
3
4
...

B
Using Paxos to order commands

C

1  2  3  4  ...

B  A  

Friday, November 8, 13
Using Paxos to order commands
Using Paxos to order commands
Using Paxos to order commands
Using Paxos to order commands

- Choose commands *independently* for each slot
Using Paxos to order commands

- Choose commands *independently* for each slot
- At least 2 RTTs per slot:
Using Paxos to order commands

- Choose commands *independently* for each slot
- At least 2 RTTs per slot:
  1. Take ownership of a slot
Using Paxos to order commands

- Choose commands *independently* for each slot
- At least 2 RTTs per slot:
  1. Take ownership of a slot
  2. Propose command
Multi-Paxos
Multi-Paxos

1 2 3 4 ...

1 2 3 4
Multi-Paxos

ABCD

1 2 3 4 ...

---

Friday, November 8, 13
Multi-Paxos

BCD

1 2 3 4 ...

A
Multi-Paxos
Multi-Paxos
Multi-Paxos
Multi-Paxos

- 1 RTT to commit
Multi-Paxos

- 1 RTT to commit
- Bottleneck for performance and availability
Can we have it all?
Can we have it all?

- High throughput, low latency
Can we have it all?

• High throughput, low latency
• Constant availability
Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas
Can we have it all?

• High throughput, low latency
• Constant availability
• Distribute load evenly across all replicas
• Use fastest replicas
Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas
- Use fastest replicas
- Use closest (lowest latency) replicas
Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas
- Use fastest replicas
- Use closest (lowest latency) replicas
Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas
- Use fastest replicas
- Use closest (lowest latency) replicas
Can we have it all?

- High throughput, low latency
- Constant availability
- Distribute load evenly across all replicas
- Use fastest replicas
- Use closest (lowest latency) replicas

Egalitarian Paxos (EPaxos)
EPaxos is all about ordering
EPaxos is all about ordering

Previous strategies:
EPaxos is all about ordering

Previous strategies:

Contend for slots
EPaxos is all about ordering

Previous strategies:

Contend for slots

Paxos
EPaxos is all about ordering

Previous strategies:

Contend for slots

One replica decides

Paxos
EPaxos is all about ordering.

Previous strategies:

- Contend for slots
- One replica decides

Paxos

Multi-Paxos, Fast Paxos, Generalized Paxos
EPaxos is all about ordering

Previous strategies:

- Contend for slots
- One replica decides
- Take turns round-robin

Paxos

Multi-Paxos, Fast Paxos, Generalized Paxos
EPaxos is all about ordering

Previous strategies:

- Contend for slots
  - Paxos

- One replica decides
  - Multi-Paxos, Fast Paxos, Generalized Paxos

- Take turns round-robin
  - Mencius

Friday, November 8, 13
EPaxos intuition
**EPaxos intuition**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
EPaxos intuition

A

C

B

1 2 3 4 ...

Friday, November 8, 13
EPaxos intuition

![Diagram of EPaxos intuition with nodes A, B, C, and D and numbers 1, 2, 3, 4, and ...]
# EPaxos intuition

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>D</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
EPaxos intuition
EPaxos intuition

1  2  3  4  ...

A  C  D
B

Friday, November 8, 13
EPaxos intuition

1 2 3 4 ...

A

C D

B

Friday, November 8, 13
EPaxos intuition

After commit @ each replica
EPaxos intuition

After commit @ each replica

A ← B ← C ← D
EPaxos intuition

After commit @ each replica

- Load balance (every replica is a leader)
EPaxos intuition

After commit @ each replica

- Load balance (every replica is a leader)
- EPaxos can choose any quorum for each command
EPaxos commit protocol

R1

R2

R3

R4

R5
EPaxos commit protocol

PreAccept(A)

A → ∅  A → ∅
EPaxos commit protocol

PreAccept(A) → A
Commit A → ∅
EPaxos commit protocol

PreAccept(A) → Commit A → ∅
EPaxos commit protocol

PreAccept(A) \rightarrow ∅

PreAccept(B) \rightarrow ∅

Commit A \rightarrow ∅
EPaxos commit protocol

PreAccept(A) ➜ A ➜ Commit A ➜ ∅ ➜ B ➜ PreAccept(B) ➜ B ➜ ∅
EPaxos commit protocol

PreAccept(A)
A $\rightarrow$ $\emptyset$

Commit A $\rightarrow$ $\emptyset$
B $\rightarrow$ \{A\}
B $\rightarrow$ $\emptyset$

PreAccept(B)

R1
R2
R3
R4
R5
EPaxos commit protocol

PreAccept(A) ➝ A

Commit A ➝ ∅

PreAccept(B) ➝ B

Accept(B) ➝ B

R1

R2

R3

R4

R5

B ➝ {A}

B ➝ {A}

PreAccept(B) ➝ B

Accept(B) ➝ B

B ➝ ∅

B ➝ ∅
EPaxos commit protocol

PreAccept(A)  \[\xrightarrow{\text{Commit}}\] A → ∅

PreAccept(B)  \[\xrightarrow{\text{Accept}}\] B → {A}

Commit B → {A}
EPaxos commit protocol

P1

\[\text{PreAccept}(A)\]  \[\text{Commit } A \rightarrow \emptyset\]

P2

\[A \rightarrow \emptyset\]

P3

\[B \rightarrow \{A\}\]

P4

\[B \rightarrow \emptyset\]

\[B \rightarrow \{A\}\]

\[\text{ACK}\]

P5

\[\text{PreAccept}(B)\]

\[\text{Accept}(B)\]  \[\text{Commit } B \rightarrow \{A\}\]

Friday, November 8, 13
EPaxos commit protocol

R1

PreAccept(A) → A → ∅

Commit A → ∅

PreAccept(C) → C → {A}

R2

B → {A}

B → ∅

R3

B → {A}

R4

B → ∅

B → {A}

R5

PreAccept(B) → B → ∅

Accept(B) → B → {A}

Commit B → {A}

Friday, November 8, 13
EPaxos commit protocol

R1
- PreAccept(A)
- A $\rightarrow \emptyset$
- Commit A $\rightarrow \emptyset$
- PreAccept(C)
- C $\rightarrow \{A\}$
- C $\rightarrow \{A, B\}$

R2
- A $\rightarrow \emptyset$
- A $\rightarrow \emptyset$

R3
- B $\rightarrow \{A\}$

R4
- B $\rightarrow \emptyset$
- B $\rightarrow \emptyset$
- B $\rightarrow \{A\}$
- ACK

R5
- PreAccept(B)
- Accept(B)
- Commit B $\rightarrow \{A\}$
EPaxos commit protocol

1. PreAccept(A) $\rightarrow \emptyset$
2. PreAccept(B) $\rightarrow \emptyset$
3. PreAccept(C) $\rightarrow \emptyset$
4. Accept(B) $\rightarrow \{A\}$
5. Commit A $\rightarrow \emptyset$
6. Commit C $\rightarrow \{A, B\}$
7. Commit C $\rightarrow \{A\}$
8. Commit B $\rightarrow \{A\}$
EPaxos commit protocol

R1
PreAccept(A) Commit A → ∅ PreAccept(C) Commit C → {A, B}

R2
A → ∅ A → ∅ C → {A} C → {A, B}

R3
B → {A} B → {A}

R4
B → ∅ B → ∅ B → {A} ACK

R5
PreAccept(B) Accept(B) Commit B → {A}

Friday, November 8, 13
Order only interfering commands

• 1 RTT
  • Non-concurrent commands
  • OR non-interfering commands

• 2 RTTs
  • Concurrent AND interfering
Interference is application-specific
Interference is application-specific

KV store

• Infer from operation key
Interference is application-specific

**KV store**

- Infer from operation key

**Google App Engine**

- Programmer-specified
Interference is application-specific

**KV store**
- Infer from operation key

**Google App Engine**
- Programmer-specified

**Relational databases**
- Most transactions are simple, can be analyzed
- Few remaining transactions interfere w/ everything
 Execution

strongly-connected component
strongly-connected component
Execution

strongly-connected component
strongly-connected component
strongly-connected component

Approximate sequence # order  (Lamport clock)
EPaxos properties
EPaxos properties

*Linearizability*: If A~B, and A committed before B proposed then A will be executed before B.
**EPaxos properties**

*Linearizability:* If $A \sim B$, and $A$ committed before $B$ proposed then $A$ will be executed before $B$.

*Fast-path quorum:* $F + \lceil F / 2 \rceil$

- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1
**EPaxos properties**

*Linearizability:* If $A \sim B$, and $A$ committed before $B$ proposed, then $A$ will be executed before $B$.

*Fast-path quorum:* $F + \lceil F / 2 \rceil$
- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1
**EPaxos properties**

*Linearizability*: If $A \sim B$, and $A$ committed before $B$ proposed
then $A$ will be executed before $B$.

*Fast-path quorum*: $F + \lceil F / 2 \rceil$
- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1
**Linearizability**: If A~B, and A committed before B proposed then A will be executed before B.

**Fast-path quorum**: \[ F + \left\lceil \frac{F}{2} \right\rceil \]
- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1
EPaxos properties

**Linearizability:** If \( A \sim B \), and \( A \) committed before \( B \) proposed, then \( A \) will be executed before \( B \).

**Fast-path quorum:** \( \lceil F / 2 \rceil \)

- Optimal for 3 and 5 replicas
- Better than Fast / Generalized Paxos by 1
Results
Optimal wide-area commit latency
Optimal wide-area commit latency
Optimal wide-area commit latency
Optimal wide-area commit latency
Optimal wide-area commit latency
Optimal wide-area commit latency
Optimal wide-area commit latency

Median Commit Latency [ms]

EPaxos
Mencius
Generalized Paxos
Multi-Paxos (CA leader)
### Optimal wide-area commit latency

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Median Commit Latency [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPaxos</td>
<td>85ms</td>
</tr>
<tr>
<td>Mencius</td>
<td>90ms</td>
</tr>
<tr>
<td>Generalized Paxos</td>
<td>130ms</td>
</tr>
<tr>
<td>Multi-Paxos (CA leader)</td>
<td>150ms</td>
</tr>
<tr>
<td></td>
<td>18ms</td>
</tr>
<tr>
<td></td>
<td>200ms</td>
</tr>
</tbody>
</table>
Optimal wide-area commit latency

Map showing median commit latencies for different geographies.

- EPaxos
- Mencius
- Generalized Paxos
- Multi-Paxos (CA leader)

Legend:
- CA
- VA

Median Commit Latency [ms]

0 50 100 150 200 250
Optimal wide-area commit latency

Median Commit Latency [ms]

EPaxos
Mencius
Generalized Paxos
Multi-Paxos (CA leader)

CA
VA
OR

0 50 100 150 200 250

Median Commit Latency [ms]
Optimal wide-area commit latency

EPaxos

Mencius

Generalized Paxos

Multi-Paxos (CA leader)

Median Commit Latency [ms]
Optimal wide-area commit latency

Median Commit Latency [ms]

CA  VA  OR  JP  EU

EPaxos

Mencius

Generalized Paxos

Multi-Paxos (CA leader)
Optimal wide-area commit latency

**EPaxos: Optimal commit latency in wide-area for 3 and 5 replicas**

<table>
<thead>
<tr>
<th>EPaxos</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mencius</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generalized Paxos</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-Paxos (CA leader)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Median Commit Latency [ms]
Higher + more stable throughput

5 replicas

- Multi-Paxos
- Mencius
- EPaxos 0%
- EPaxos 2%
- EPaxos 100%

0 10000 20000 30000 40000 50000
Operations / sec

3 replicas

- Multi-Paxos
- Mencius
- EPaxos 0%
- EPaxos 2%
- EPaxos 100%

0 10000 20000 30000 40000 50000
Operations / sec

Friday, November 8, 13
Higher + more stable throughput

When one replica is slow
EPaxos: higher throughput w/ batching

5 ms batching, local area

Latency [ms]
log scale

Throughput (ops / sec)

Multi-Paxos
EPaxos 0%
EPaxos 100%
EPaxos: higher throughput w/ batching

5 ms batching, local area

Latency [ms] log scale

Throughput (ops / sec)

- Multi-Paxos
- EPaxos 0%
- EPaxos 100%
EPaxos: higher throughput w/ batching

5 ms batching, local area

Latency [ms] log scale

Throughput (ops/sec)

- Multi-Paxos
- EPaxos 0%
- EPaxos 100%
EPaxos: higher throughput w/ batching

5 ms batching, local area

Latency [ms] log scale

Throughput (ops / sec)

- Multi-Paxos
- EPaxos 0%
- EPaxos 100%
EPaxos: higher throughput w/ batching

5 ms batching, local area

Latency [ms] log scale

Throughput (ops / sec)

Multi-Paxos
EPaxos 0%
EPaxos 100%
Constant availability

![Graph showing commit throughput over time for Multi-Paxos and another system. The graph indicates that after a leader failure at time 15, both systems maintain a high commit throughput.](image-url)
Constant availability

Commit throughput [ops/sec]

Time [sec]

leader failure

delayed commits

replica failure

Multi-Paxos

Mencius
Constant availability

![Graph showing comparison between EPaxos, Mencius, and Multi-Paxos in terms of commit throughput and leader/failure resilience.](image)
EPaxos insights
EPaxos insights

Order commands explicitly
EPaxos insights

Order commands explicitly

High throughput
Order commands explicitly

High throughput

Stability
EPaxos insights

Order commands explicitly

- High throughput
- Stability
- Low latency
EPaxos insights

Order commands explicitly

Optimize only delays that matter (clients co-located w/ closest replica)

High throughput
Stability
Low latency
EPaxos insights

Order commands explicitly

Optimize only delays that matter
(clients co-located w/ closest replica)

Smaller quorums

High throughput
Stability
Low latency
EPaxos insights

Order commands explicitly

- Optimize only delays that matter (clients co-located w/ closest replica)

- Smaller quorums

  - High throughput
  - Stability
  - Low latency
Formal Proof

http://cs.cmu.edu/~imoraru/epaxos/tr.pdf

Open Source Release

http://github.com/efficient/epaxos