Bolt-On Global Consistency for the Cloud

Zhe Wu, Edward Wijaya, Muhammed Uluyol, Harsha V. Madhyastha

University of Michigan
Geo-distribution for Low Latency
Geo-distribution Requires Data Replication
Geo-distribution Requires Data Replication
Cloud Simplifies App Deployment
Cloud Simplifies App Deployment
Application Needs to Manage Replication

Isolated storage services
Application Needs to Manage Replication

Isolated storage services

[Diagram showing replication between different cloud services such as Amazon Web Services and Google Cloud Platform]
Application Needs to Manage Replication

Isolated storage services

No replication across cloud providers
Challenges for Data Replication in Cloud

Conflict?
Challenges for Data Replication in Cloud

**Paxos**
Megastore (CIDR'11)
Spanner (OSDI'12)
MDCC (Eurosys'13)
Tapir (SOSP'15) .....
Challenges for Data Replication in Cloud

Paxos

Propose writes
Challenges for Data Replication in Cloud

Paxos

Propose writes
Challenges for Data Replication in Cloud

Paxos

Propose writes

Paxos logic

Paxos logic

Paxos logic

Amazon Web Services

Cloud
Challenges for Data Replication in Cloud

Paxos

Propose writes

Paxos logic

PUT/GET

amazon web services

amazon web services

Cloud services
Challenges for Data Replication in Cloud

Paxos

Paxos logic

VMs managed by application

PUT/GET

Web services

Application managed by VMs
Challenges for Data Replication in Cloud

**Paxos**

1. High cost

VMs managed by application

PUT/GET
Challenges for Data Replication in Cloud

**Paxos**

1. High cost
2. Bottleneck

PUT/GET

VMs managed by application

---

18
Challenges for Data Replication in Cloud

**Paxos**
1. High cost
2. Bottleneck
Challenges for Data Replication in Cloud

**Paxos**
1. High cost
2. Bottleneck

**Paxos with limited interface**

- Disk Paxos (Distributed Computing'03)
- pPaxos (ATC'15)
Challenges for Data Replication in Cloud

**Paxos**
1. High cost
2. Bottleneck

**Paxos with limited interface**

*DiskPaxos, pPaxos*
Challenges for Data Replication in Cloud

**Paxos**
1. High cost
2. Bottleneck

**Paxos with limited interface**

**DiskPaxos, pPaxos**
1. Conflict-free write Paxos ops
Challenges for Data Replication in Cloud

**Paxos**
1. High cost
2. Bottleneck

**Paxos with limited interface**

**DiskPaxos, pPaxos**
1. Conflict-free write Paxos ops
2. Read from the log to replay Paxos logic
Challenges for Data Replication in Cloud

**Paxos**
1. High cost
2. Bottleneck

**Paxos with limited interface**
1. High latency

**DiskPaxos, pPaxos**
1. Conflict-free write Paxos ops
2. Read from the log to replay Paxos logic
Challenges for Data Replication in Cloud

**Paxos**
1. High cost
2. Bottleneck

**Paxos with limited interface**
1. High latency
2. High cost

**DiskPaxos, pPaxos**
1. Conflict-free write Paxos ops
2. Read from the log to replay Paxos logic
### Problems with Existing Solutions

<table>
<thead>
<tr>
<th></th>
<th>Low latency</th>
<th>Compatible with limited interface</th>
<th>Low cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional Paxos</td>
<td>✔️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Disk Paxos, pPaxos</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
</tbody>
</table>
Our Solution: **Consistent Replication In the Cloud**

<table>
<thead>
<tr>
<th></th>
<th>Low latency</th>
<th>Compatible with limited interface</th>
<th>Low cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional Paxos</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Disk Paxos, pPaxos</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td><strong>CRIC</strong></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
CRIC Overview

Key-value store (reads/writes)
CRIC Overview

Key-value store (reads/writes)
CRIC Overview

Key-value store
(reads/writes)

✅ Apps directly read/write data from/to cloud storage
CRIC Overview

Key-value store (reads/writes)

- Apps directly read/write data from/to cloud storage
- Low latency (1 RTT)
CPaxos In Action

Executing a write in traditional Paxos

- Proposer (App)
- Acceptor
- Storage

Steps:
- Prepare
- Accept
CPaxos In Action

Executing a write in traditional Paxos

- Proposer (App)
- Acceptor
- Storage

Prepare
Accept

Executing a write in CPaxos

- Proposer (App)
- Storage
CPaxos In Action

Executing a write in traditional Paxos

Proposer (App) → Acceptor → Storage

- Prepare
- Accept

Executing a write in CPaxos

Proposer (App) → Storage (Passive acceptor)
CPaxos In Action

Executing a write in traditional Paxos

- Proposer (App)
- Acceptor
- Storage

- Prepare
- Accept

Executing a write in CPaxos

- Proposer (App)
- Storage (Passive acceptor)

- Read Paxos state
- Update Paxos state

Run Paxos prepare logic

Executing a write in traditional Paxos

Executing a write in CPaxos
CPaxos In Action

Executing a write in traditional Paxos

Proposer (App) → Acceptor → Storage

Prepare

Accept

Executing a write in CPaxos

Proposer (App) → Storage (Passive acceptor)

Run Paxos prepare logic

Read Paxos state

Update Paxos state

Run Paxos accept logic

Update Paxos state and data

Executing a write in traditional Paxos

Proposer (App) → Acceptor → Storage

Prepare

Accept

Executing a write in CPaxos

Proposer (App) → Storage (Passive acceptor)

Run Paxos prepare logic

Read Paxos state

Update Paxos state

Run Paxos accept logic

Update Paxos state and data
CPaxos In Action

Executing a write in CPaxos

Proposer (App)
- Run Paxos prepare logic
- Run Paxos accept logic

Storage (Passive acceptor)
- Read Paxos state
- Update Paxos state
- Update Paxos state and data

Proposer
Executing a write in CPaxos

Proposer (App)

Storage (Passive acceptor)

Run Paxos prepare logic

Run Paxos accept

Read Paxos state

Update Paxos state

Leverage cloud supported **conditional-PUT** (available in **all** cloud storage services)
CPaxos In Action

Executing a write in traditional Paxos

- **Proposer (App)**
- **Acceptor**
- **Storage**

  - **Prepare**
  - **Accept**

  **2 RTTs**

Executing a write in CPaxos

- **Proposer (App)**
- **Storage (Passive acceptor)**

  - **Prepare logic**
  - **Accept logic**
  - **Read Paxos state**
  - **Update Paxos state**
  - **Update Paxos state and data**

  **3 RTTs**
CPaxos In Action

Executing a write in traditional Paxos:
- Proposer (App)
  - Prepare
  - Accept

Executing a write in CPaxos:
- Proposer (App)
  - Prepare logic
  - Accept logic
- Accept
- Prepare
- Storage (Passive acceptor)
  - Read Paxos state
  - Update Paxos state
  - Update Paxos state and data

Can be omitted when:
1. Write follows a read
2. Object creation
CPaxos In Action

Executing a write in traditional Paxos

Proposer (App)  Accept (App)

Preparing

Executing a write in CPaxos

Proposer (App)

Storage (Passive acceptor)

Read Paxos state

Prepare logic

Update Paxos state and data

Can be omitted when:
1. Write follows a read
2. Object creation

Leverage Fast Paxos to execute reads and writes in one round
Tradeoff: High Latency under Conflict

Propose 0

Propose 1
Tradeoff: High Latency under Conflict
Tradeoff: High Latency under Conflict
Tradeoff: High Latency under Conflict

Higher proposal will succeed in traditional Paxos
Tradeoff: High Latency under Conflict

Reason for conflict: variance in latency to different data centers
Optimization: Staggered Requests

Time

Cloud Storage
Propose 0

Cloud Storage

Cloud Storage
Propose 1
Optimization: Staggered Requests

![Diagram showing staggered requests between a device and cloud storage with proposed time intervals labeled as Propose 0 and Propose 1.]

- Time
- Cloud Storage
- Propose 0
- Propose 1
Optimization: Staggered Requests
Optimization: Staggered Requests
Optimization: Staggered Requests

Propose 0

Propose 1

Detect conflict faster
Optimization: Staggered Requests

Observation: **low network latency variance between cloud DCs**

Detect conflict faster
CRIC Optimizations

- Reduce **latency under conflict**
  - Staggered Requests

- Reduce **reader-write-back**
  - Asynchronous commit notification

- Reduce **storage and data transfer cost**
  - Separates data and Paxos log
  - Aggressive garbage collection in Accept phase
  - Store data digest in Paxos log
**CRIC Optimizations**

- Reduce **latency under conflict**
  - Staggered Requests

- Reduce **reader-write-back**
  - Asynchronous commit notification

- Reduce **storage and data transfer cost**

**Cost-effective**

Only **one version** of the data is stored in each replica data center
Evaluation

- Deploy CRIC in 5 Azure data centers and run YCSB workload

- Comparison systems:
  - active acceptor Fast Paxos
  - passive acceptor pPaxos
Evaluation

- Deploy CRIC in **5 Azure data centers** and run **YCSB workload**

- Comparison systems:
  - active acceptor **Fast Paxos**
  - passive acceptor **pPaxos**

- How does CRIC compare with respect to cost and performance?
Evaluation

- Deploy CRIC in **5 Azure data centers** and run **YCSB workload**

- Comparison systems:
  - active acceptor **Fast Paxos**
  - passive acceptor **pPaxos**

- How does CRIC compare with respect to cost and performance?

- How effective are staggered requests?
CRIC Enables Low Cost

The chart illustrates the normalized cost for different block sizes (256B, 1KB, 4KB) and read/write ratios (R/W=1, R/W=10). The cost is broken down into network, request, and VM components.
CRIC Enables Low Cost

Normalized Cost

CRIC

Fast Paxos

pPaxos

R/W=1  R/W=10  R/W=1  R/W=10  R/W=1  R/W=10

256B  1KB  4KB

Network  Request  VM
CRIC Enables Low Cost

Eliminate need for relay VMs
CRIC Enables Low Cost

Reduce I/O and data transfers

Fast Paxos

CRIC

pPaxos

Normalized Cost

Network
Request
VM

R/W=1  R/W=10  R/W=1  R/W=10  R/W=1  R/W=10

256B  1KB  4KB
CRIC Enables Low Cost

CRIC can reduce cost by 20% ~ 50%
... without Sacrificing Performance

![Graph showing median latency under low conflict for Read and Write operations with FastPaxos, pPaxos, and CRIC compared to 63 ms.](image-url)
... without Sacrificing Performance

Same performance as FastPaxos
... without Sacrificing Performance

Better write performance than pPaxos

Median latency (ms) Under low conflict

- FastPaxos
- pPaxos
- CRIC

Read    Write
Staggered Requests Lower Latency Under Conflict

Median latency for successful writes (ms)

# of client servers per DC

Without staggered

With staggered

Increasing conflict rate

Without staggered

With staggered

Increasing conflict rate
Staggered Requests Lower Latency Under Conflict

Median latency for successful writes (ms)

# of client servers per DC

Without staggered

With staggered

Lower latency for same conflict rate

Increasing conflict rate
Conclusions

- **Consistent Replication In the Cloud**
  - Compatible with cloud storage interface
  - One round read/write in common case
  - Low cost

Thank you

towuzhe@gmail.com