LOCKS WITH INTENT

Srinath Setty, Chunzhi Su,* Jacob R. Lorch, Lidong Zhou, Hao Chen,§ Parveen Patel, and Jinglei Ren

Microsoft Research
*The University of Texas at Austin
§Shanghai Jiao Tong University
Cloud application

Simple APIs that hide cloud storage’s distributed machinery
APPLICATION FAILURE

- Failure
  - Application processes or VMs can fail
  - Network can drop/reorder messages
- Such failures can introduce inconsistencies to application’s state
- Example: Consistency between application’s data and indexes
  - Invariants should hold even with: Concurrent operations on cloud storage state & Concurrent operations on the client
- A significant burden on application developers
APPLICATION FAILURE

Does failing at line $i$ violate any invariant?

Moreover, the developer must reason about concurrent executions.
HIGHLIGHTS OF OLIVE

● Powerful new primitives: intents and locks with intent
  ● Exactly-once execution semantics
  ● Mutual exclusion; locked objects associated with intents
  ● Eventual progress
● New mechanisms to implement this abstraction
  ● Distributed atomic affinity logging (DAAL)
  ● Intent collector
● Built several real-world, fault-tolerant cloud services
  ● Live re-partitioning of tables
  ● Snapshotting service
  ● ACID transactions
  ● …
Overview of Olive

- Application code
  - Intents
  - Locks with intents
    - Exactly-once protocol
    - Lock primitive
    - Intent collector
  - Storage model: Create, Read, Update, Delete, UpdateIfUnchanged, AtomicBatchUpdate, Scan
    - Unreliable network
  - Simple APIs that hide distributed machinery
    - Cloud storage systems
      - (Amazon DynamoDB, Azure table store, …)
  - Unmodified storage
Overview of Olive

Application code

Intents
- Exactly-once protocol

Locks with intents
- Lock primitive

- Intent collector

Storage model: Create, Read, Update, Delete, UpdateIfUnchanged, AtomicBatchUpdate, Scan

Unreliable network

Simple APIs that hide distributed machinery
- Cloud storage systems
  - (Amazon DynamoDB, Azure table store, …)

Unmodified storage
OVERVIEW OF OLIVE

Application code

- Intents
- Locks with intents

Exactly-once protocol
- Lock primitive
- Intent collector

Storage model: Create, Read, Update, Delete, UpdateIfUnchanged, AtomicBatchUpdate, Scan

Unreliable network

Simple APIs that hide distributed machinery

Cloud storage systems
- (Amazon DynamoDB, Azure table store, …)

Unmodified storage
INTENT

- An arbitrary snippet of code:
  - Cloud storage operations
  - Local computation (loops, recursion, control flow, ...)
- Goal of exactly-once execution: Code should run as if it is executed by a single, failure-free client
OVERVIEW OF OLIVE

Application code

Intents

Locks with intents

Exactly-once protocol

Lock primitive

Intent collector

Storage model: Create, Read, Update, Delete, UpdateIfUnchanged, AtomicBatchUpdate, Scan

Unreliable network

Simple APIs that hide distributed machinery

Cloud storage systems
(Amazon DynamoDB, Azure table store, …)

Unmodified storage
TRACKING AND EXECUTING

- Intent id
- executionLog (DAAL): Leverage AtomicBatchUpdate for objects in the same shard or partition
  - A global cloud table for recording read operations, and
  - DAAL entries spread throughout
- locks with intent: Locks are owned by intents, not client VMs -> any client can unlock an object by executing the associated intent
- Intent Collector: ensure liveness
EVALUATION

- Do Olive’s abstractions simplify building fault-tolerant applications?
- Yes. Olive’s library is 2,000 lines of code.

<table>
<thead>
<tr>
<th>Service</th>
<th>Without Olive</th>
<th>With Olive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snapshots</td>
<td>987</td>
<td>665</td>
</tr>
<tr>
<td>OCC-transactions</td>
<td>2,201</td>
<td>408</td>
</tr>
<tr>
<td>Live table re-partitioning</td>
<td>2,116</td>
<td>474</td>
</tr>
</tbody>
</table>
How do Olive-based artifacts perform relative to alternatives?

Olive is competitive with the baseline for most operations.
OLIVE’S KEY TAKEAWAYS

- We propose two new primitives: Intents and locks with intent, which guarantee exactly-once semantics, mutual exclusion, and eventual progress
- We propose new mechanisms: DAAL and an intent collector
Q & A