Dynamic Exploration of Behaviors for Cloud Programs

Erdal Mutlu
Koc University
Replication

• Scalability, availability and reliability arise from replication
Consistency

• Replicas must be kept consistent
  – Modification on one replica triggers modifications on all other replicas
  – Propagating each modification to each replica can degrade performance

• Solution: Make Consistency Weaker
Weak Consistency Models

• Programs can behave differently for different consistency models

\[
\begin{align*}
\text{x, y: intreg} \\
\text{x.wr(all)} & \quad i=y.rd \\
\text{x.wr(noboss)} & \quad j=x.rd \\
\text{y.wr(photo)} & \quad \\
\end{align*}
\]

• The programmer-visible non-determinism in such systems is different from shared memory non-determinism
CHESS

• CHESS is a tool for
  – Systematically enumerating thread interleavings
  – Reliably reproducing concurrent executions
CHESS Guarantees

- Every program run takes a different thread interleaving

Thread 1
- x = 1;
- y = 1;

Thread 2
- x = 2;
- y = 2;
CHESS Guarantees

- Reproduce the interleaving for every run

<table>
<thead>
<tr>
<th>1.4: read DATA_READ 513</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5: TASK_FORK 2</td>
</tr>
<tr>
<td>1.6: TASK_RESUME 2</td>
</tr>
<tr>
<td>1.7: Monitor.Enter LOCK_ACQUIRE 514</td>
</tr>
<tr>
<td>1.8: read DATA_READ 512</td>
</tr>
<tr>
<td>1.9: write DATA_WRITE 512</td>
</tr>
<tr>
<td>1.10: Monitor.Exit LOCK_RELEASE 514</td>
</tr>
<tr>
<td>1.11: Thread.Join(-1) TASK_JOIN2 (BLOCKS)</td>
</tr>
<tr>
<td>2.1: TASK_BEGIN 2</td>
</tr>
<tr>
<td>2.2: Monitor.Enter LOCK_ACQUIRE 514</td>
</tr>
<tr>
<td>2.3: read DATA_READ 512</td>
</tr>
<tr>
<td>2.4: Monitor.Exit LOCK_RELEASE 514</td>
</tr>
<tr>
<td>2.5: Monitor.Enter LOCK_ACQUIRE 514</td>
</tr>
<tr>
<td>2.6: write DATA_WRITE 512</td>
</tr>
<tr>
<td>2.7: Monitor.Exit LOCK_RELEASE 514</td>
</tr>
<tr>
<td>2.8: TASK_END 2</td>
</tr>
<tr>
<td>1.11: Thread.Join(-1) TASK_JOIN2</td>
</tr>
<tr>
<td>1.12: Monitor.Enter LOCK_ACQUIRE 514</td>
</tr>
<tr>
<td>1.13: read DATA_READ 512</td>
</tr>
</tbody>
</table>
This Project

• Current state of the art:
  – To experiment with programs for the cloud, must deploy code on the cloud. \cite{1,2}
  – Cloud emulation tools do not provide systematic exploration and replay capabilities\cite{3}

• Our goal: Enable systematic pre-deployment
  – Behavior exploration
  – Debugging

• Build a CHESS-like tool for programs written for cloud systems using weak consistency models

• Challenges:
  – Build tools and algorithms to explore precisely and efficiently \textit{programmer-visible} non-determinism
Motivation

- Cloud Computing
Motivation

Is the data CONSISTENT??
CAP Theorem \cite{1, 2}

• CAP Theorem: cannot satisfy all three of these guarantees
  – Consistency
    • all nodes see the same data at the same time
  – Availability
    • a guarantee that every request receives a response about whether it was successful or failed
  – Partition tolerance
    • the system continues to operate despite arbitrary message loss or failure of part of the system
Flavors of Weak Consistency

• Types of Consistency
  – Strong Consistency
  – Casual Consistency
    • Cross-Object\(^1\)
    • Per-Object\(^2\)
  – Session Guarantees\(^3\)
  – Eventual Consistency\(^4\)

\(^1\)Cross-Object, \(^2\)Per-Object, \(^3\)Session Guarantees, \(^4\)Eventual Consistency
Programming with Weak Consistency

• **Eventual Consistency**
  – Updates are not conditional on very latest state
  – Examples: Ratings, Shopping Cart, Comments, Settings, Chat, Grocery List, Playlist, Calendar, Mailbox, Contacts ...
  – Implementations: Riak\(^1\)
  – **But: how to program on this model?**
Programming with Weak Consistency

• New data types
  – Cloud types\(^1\)
  – Conflict-free replicated data types (CRDT)\(^2\)

• New programming languages
  – Bloom/Bloom\(^L\) \(^3,4\)

• New programming models
  – Orleans\(^5\)
Correctness

• Programming became harder with
  – Weaker consistency models
  – New data types
  – New programming languages
  – New programming models

• Crucial to have verification tools for weak consistency models
  – Consistency model formalization
  – New data type formalization
  – Message reordering
Dynamic Verification for Weak Consistency Models

- **Goal:** Systematic and efficient exploration of programmer-visible non-determinism exposed by weak consistency models
Specification of Replicated Data Types

• Capture all programmer-visible non-determinism in two relations
  – Visibility Relation
    • Events visible to a context
      – $C \rightarrow_{\text{vis}} D$
  – Arbitration Relation
    • Order events
      – $A \rightarrow_{\text{ar}} B$
Examples

• Counter (ctr)

\[ \mathcal{F}_{\text{ctr}}(\text{inc}, V, \text{vis}, \text{ar}) = \bot; \]
\[ \mathcal{F}_{\text{ctr}}(\text{rd}, V, \text{vis}, \text{ar}) = (\text{the number of inc operations in } V). \]

• Integer Register (Last-Write Wins)

– Var: Ordered sequence of visible events with arbitration relation

\[ \mathcal{F}_{\text{intreg}}(f, V, \text{vis}, \text{ar}) = S_{\text{intreg}}(V^{\text{ar}}f) \]
\[ S_{\text{intreg}}(\sigma \text{ rd}) = k, \text{ if wr}(0) \sigma = \sigma_1 \text{ wr}(k) \sigma_2 \text{ and } \]
\[ \sigma_2 \text{ does not contain wr operations;} \]
Motivation

$x, y : $ integer registers
Spec: Last-Write-Wins

Replica 1
- $x.wr(all)$;
- $x.wr(noboss)$;
- $y.wr(photo)$;

Replica 2
- $y.rd()$;
- $x.rd()$;

4/19/2013
Weak Consistency Models

- Axiomatic specification of system behaviors
  - Formalize programmer-visible non-determinism

**Well-formedness Axioms**

SOWF: so is the union of transitive, irreflexive and total orders on actions by each session

\[
\forall a, b. \ a \xrightarrow{\text{vis}} b \implies \text{obj}(a) = \text{obj}(b)
\]

VISWF: \(\forall a, b. \ a \xrightarrow{\text{vis}} b \implies \text{obj}(a) = \text{obj}(b)\)

ARWF: \(\forall a, b. \ a \xrightarrow{\text{ar}} b \implies \text{obj}(a) = \text{obj}(b)\),
ar is transitive and irreflexive, and \(\text{ar}_{\text{vis}^{-1}(a)}\) is a total order for all \(a \in A\)

**Data Type Axiom**

RVAL: \(\forall a \in A. \ \text{rval}(a) = \mathcal{F}_{\text{type}(a)}(\text{ctxt}(a))\)

**Auxiliary Relations**

- Per-object session order: \(\text{soo} = (\text{so} \cap \text{sameobj})\)
- Per-object causality order: \(\text{hbo} = (\text{soo} \cup \text{vis})^+\)
- Causality order: \(\text{hb} = (\text{so} \cup \text{vis})^+\)

**Causality Axioms**

- POCV (Per-Object Causal Visibility): \(\text{hbo} \subseteq \text{vis}\)
- POCA (Per-Object Causal Arbitration): \(\text{hbo} \subseteq \text{ar}\)
- COCV (Cross-Object Causal Visibility): \((\text{hb} \cap \text{sameobj}) \subseteq \text{vis}\)
- COCA (Cross-Object Causal Arbitration): \(\text{hb} \cup \text{ar}\) is acyclic
Motivation

x, y : integer registers
Spec: Last-Write-Wins

Replica 1
x.wr(all);
x.wr(noboss);
y.wr(photo);
ar

Replica 2
y.rd();
x.rd();

Auxiliary relations
Per-object session order: sso = (so ∩ sameobj)
Per-object causality order: hbo = (sso ∪ vis)⁺
Causality order: hb = (so ∪ vis)⁺

Causality axioms
POCV (Per-Object Causal Visibility): hbo ⊆ vis
POCA (Per-Object Causal Arbitration): hbo ⊆ ar
COCV (Cross-Object Causal Visibility): (hb ∩ sameobj) ⊆ vis
COCA (Cross-Object Causal Arbitration): hb ∪ ar is ayclic
Dynamic Verification for Weak Consistency Models

What we need?

- Explore consistency model exposed non-determinism
  - Message Interleavings
  - Model Axioms

What CHESS have?

- Just reasons about thread interleaving non-determinism

Auxiliary relations

\[
S' = \text{Per-object session order: soo } = (so \cap \text{sameobj})
\]

\[
V = \text{Per-object causality order: hbo } = (so \cup \text{vis})
\]

\[
A = \text{Causality order: hb } = (so \cup \text{vis})^+
\]

Causality axioms

POCV (Per-Object Causal Visibility): hbo \subseteq vis

POCA (Per-Object Causal Arbitration): hbo \subseteq ar

COCV (Cross-Object Causal Visibility): (hb \cap \text{sameobj}) \subseteq vis

COCA (Cross-Object Causal Arbitration): hb \cup ar \text{ is acyclic}

Transform the cloud program to an equivalent multi-threaded program

Soundness Proof!!!
Dynamic Verification for Weak Consistency Models

Outgoing Message Queue

Replica 1

x.wr(all);

x.wr(noboss);

y.wr(photo);

Incoming Message Queue

Replica N

Thread N

Thread 4+1

Thread 4+2

Thread 4+3

Waits
Initial Results

• For cross-object consistency model
  – Implemented the model over .NET with threads and tested with Chess
  – Trace
Future Work

• Tool for systematically testing the application for different message ordering between replications for different consistency models
• Take into account different weak memory data type specifications
• Compare different consistency model for a given program for finding the weakest