hochschule mannheim

Understanding Eventual Consistency

MSI Presentation SS2014

Horst Schneider, Patrick Beedgen
Hochschule Mannheim

June 17th, 2014

Introduction

„...the storage system guarantees that if no new updates are made to the object, eventually all accesses will return the last updated valuee“
–W. Vogels (2009)

Introduction

Interpretations of Eventual Consistency

Interpretation 1:

> *"When you read data[...], the response might not reflect the results of a recently completed write operation. The response might include some stale data. Consistency across all copies of the data is usually reached within a second; so if you repeat your read request after a short time, the response returns the latest data."*

Interpretation 2:

> *"This sort of system we term "single writer eventual consistency". So what are its properties?*
> *(1) A client could read stale data. (2) The client could see out-of-order write operations. [...] So this is our weakest form of consistency - eventually consistent with out of order reads in the short term."*

Introduction

Interpretations of Eventual Consistency

DynamoDB Documentation

> *"When you read data[...], the response might not reflect the results of a recently completed write operation. The response might include* **some stale data***. Consistency across all copies of the data is* **usually reached within a second***; so if you repeat your read request after a short time, the response returns the latest data."*

MongoDB Documentation

> *"This sort of system we term "single writer eventual consistency". So what are its properties?*
> *(1)A client could read stale data.*
> *(2)The client could see out-of-order write operations.[...]*
> *So this is our weakest form of consistency - eventually consistent with* **out of order reads** *in the short term."*

The Problem

- Disparate and low-level formalisms
  consistency model is tied to system implementation
- Weak guarantees
  in realistic scenarios updates **never** *stop*
- Conflict resolution policies
  resolution of conflicts in multiple replicas
- Combinations of different consistency levels
  strong consistency may be needed at certain times

⇒ Some sort of formalism is needed to define semantics of Eventual Consistency

Agenda

1. Replicated Data Types

Anfang Hauptteil Horst

Replicated Data Types

- A replicated database stores **objects** $\mathrm{Obj} = \{x, y, \dots\}$

Replicated Data Types

- A replicated database stores **objects** $\mathrm{Obj} = \{x, y, \dots\}$
- Every object $x \in \mathrm{Obj}$ has
  - a **value** $\in \mathrm{Val}$

Replicated Data Types

- A replicated database stores **objects** $\mathrm{Obj} = \{x, y, \dots\}$
- Every object $x \in \mathrm{Obj}$ has
  - a **value** $\in \mathrm{Val}$
  - a **type** $\mathsf{type}(x)$

Replicated Data Types

- A replicated database stores **objects** $\mathrm{Obj} = \{x, y, \dots\}$
- Every object $x \in \mathrm{Obj}$ has
  - a **value** $\in \mathrm{Val}$
  - a **type** type(x)
  - **operations** $\mathrm{Op}_{\mathrm{type}(x)}$ that a client can perform on it

Replicated Data Types

- A replicated database stores **objects** $\mathrm{Obj} = \{x, y, \dots\}$
- Every object $x \in \mathrm{Obj}$ has
  - a **value** $\in \mathrm{Val}$
  - a **type** type(x)
  - **operations** $\mathrm{Op}_{\mathrm{type}(x)}$ that a client can perform on it
- Two examples: Int Register **intreg**, Counter **ctr**

$$\mathrm{Op}_{\mathrm{ctr}} = \{\mathrm{rd}, \mathrm{inc}\}$$
$$\mathrm{Op}_{\mathrm{intreg}} = \{\mathrm{rd}, \mathrm{wr}(k) | k \in \mathbb{Z}\}$$

Replicated Data Types

Sequential Data Type Specification

In a *strongly consistent system*, the semantics of a data type can be described by a function

$$S_\tau \ : \mathrm{Op}_\tau^+ \rightarrow \mathrm{Val}$$

Replicated Data Types

Sequential Data Type Specification

In a *strongly consistent system*, the semantics of a data type can be described by a function

$$S_\tau \ : \mathrm{Op}_\tau^+ \to \mathrm{Val}$$

Examples:

$$S_{\mathrm{ctr}}(\sigma \mathrm{rd}) = (\text{number of inc operations in } \sigma);$$

Replicated Data Types

Sequential Data Type Specification

In a *strongly consistent system*, the semantics of a data type can be described by a function

$$S_\tau \ : \mathrm{Op}_\tau^+ \to \mathrm{Val}$$

Examples:

$$\begin{aligned} S_{\mathrm{ctr}}(\sigma \mathrm{rd}) &= (\text{number of inc operations in } \sigma); \\ S_{\mathrm{intreg}}(\sigma \mathrm{rd}) &= k; \ \text{if } \mathrm{wr}(0)\sigma = \sigma_1 \mathrm{wr}(k) \sigma_2 \text{ and} \\ &\qquad \sigma_2 \text{ does not contain wr operations} \end{aligned}$$

Replicated Data Types

Sequential Data Type Specification

In a *strongly consistent system*, the semantics of a data type can be described by a function

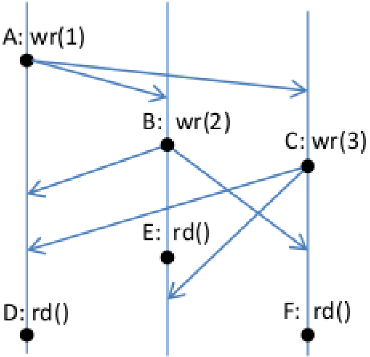
$$S_\tau \;: \mathrm{Op}_\tau^+ \to \mathrm{Val}$$

Examples:

$$\begin{aligned} S_{\mathrm{ctr}}(\sigma \mathrm{rd}) &= (\text{number of inc operations in } \sigma); \\ S_{\mathrm{intreg}}(\sigma \mathrm{rd}) &= k;\ \text{if } \mathrm{wr}(0)\sigma = \sigma_1 \mathrm{wr}(k) \sigma_2 \text{ and} \\ &\qquad\qquad \sigma_2 \text{ does not contain wr operations} \\ S_{\mathrm{intreg}}(\sigma\ \mathrm{wr}(k)) &= S_{\mathrm{ctr}}(\sigma\ \mathrm{inc}) = \bot; \end{aligned}$$

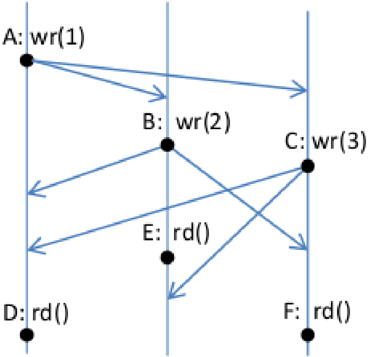
Replicated Data Types

Conflict Resolution Strategies



Replicated Data Types

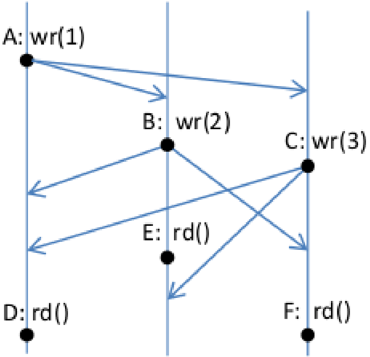
Conflict Resolution Strategies



1. Make concurrent operations commutative

Replicated Data Types

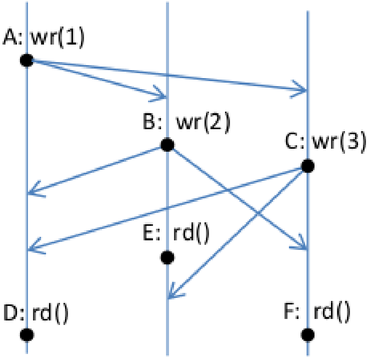
Conflict Resolution Strategies



1. Make concurrent operations commutative
2. Order concurrent operations

Replicated Data Types

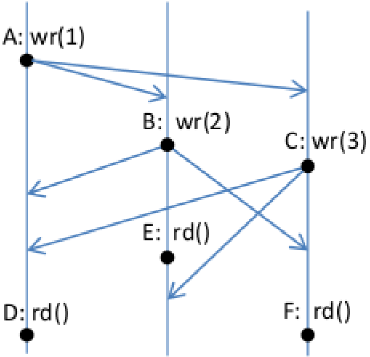
Conflict Resolution Strategies



1. Make concurrent operations commutative
2. Order concurrent operations
3. Flag conflicts (let the user decide)

Replicated Data Types

Conflict Resolution Strategies



1. Make concurrent operations commutative
2. Order concurrent operations
3. Flag conflicts (let the user decide)
4. Resolve conflicts semantically

Replicated Data Types

Replicated Data Type Specification

- S_τ is not strong enough to formalize these strategies

Replicated Data Types

Replicated Data Type Specification

- S_τ is not strong enough to formalize these strategies
- visibility and order of preceding operations have to be included

Replicated Data Types

Replicated Data Type Specification

- S_τ is not strong enough to formalize these strategies
- visibility and order of preceding operations have to be included
- F_τ: takes an **operation context** and returns a **value**

$$F_\tau(C) \in \mathrm{Val}$$

Replicated Data Types

Replicated Data Type Specification

- S_τ is not strong enough to formalize these strategies
- visibility and order of preceding operations have to be included
- F_τ: takes an **operation context** and returns a **value**

$$F_\tau(C) \in \mathrm{Val}$$

- operation context C adds **visibility** and **arbitration relations** to preceding operations:

$$C = (f, V, \mathrm{ar}, \mathrm{vis})$$

Replicated Data Types

Replicated Data Type Specification

- S_τ is not strong enough to formalize these strategies
- visibility and order of preceding operations have to be included
- F_τ: takes an **operation context** and returns a **value**

$$F_\tau(C) \in \mathrm{Val}$$

- operation context C adds **visibility** and **arbitration relations** to preceding operations:

$$C = (f, V, \mathrm{ar}, \mathrm{vis})$$
$$u \xrightarrow{\mathrm{vis}} v, \mathrm{vis} \subseteq V \times V$$

Replicated Data Types

Replicated Data Type Specification

- S_τ is not strong enough to formalize these strategies
- visibility and order of preceding operations have to be included
- F_τ: takes an **operation context** and returns a **value**

$$F_\tau(C) \in \mathrm{Val}$$

- operation context C adds **visibility** and **arbitration relations** to preceding operations:

$$C = (f, V, \mathrm{ar}, \mathrm{vis})$$
$$u \xrightarrow{\mathrm{vis}} v, \mathrm{vis} \subseteq V \times V$$
$$u \xrightarrow{\mathrm{ar}} v, \mathrm{ar} \subseteq V \times V$$

Ende Hauptteil Horst

Anfang Hauptteil Patrick

Axiomatic Specification Framework

Levels of Eventual Consistency

- With replicated data types we can define multiple forms of eventual consistency
  - Basic eventual consistency
  - Ordering guarantees
  - on-demand consistency strengthening
- Every form contains multiple axioms

Axiomatic Specification Framework

Client Interaction Model

- Clients often wish to perform multiple operations within some context
- bla

Axiomatic Specification Framework

Basic Eventual Consistency Axioms

- Axioms a database has to fulfill to be eventual consistent
- SOWF, ARWF, VISWF, RVAL, EVENTUAL, THINAIR

Axiomatic Specification Framework

Session guarantees

- Axioms that ensure that databases stay consistent within a single session with a client
- RYW, MR, WYRV, WFRA, MWV, MWA

Axiomatic Specification Framework

Causal Consistency Axioms

- POCV, POCA, COCV, COCA

Ende Hauptteil Patrick

Conclusion

- Which problems does the techreport solve?
- What is not solved by it?
- What do **we** think about it?