Just-Right Consistency

As available as possible
As consistent as necessary
Correct by design

Marc Shapiro, Sorbonne-Université—LIP6 & Inria
Annette Bieniusa, U. Kaiserslautern
Nuno Preguiça, U. Nova Lisboa
Christopher Meiklejohn, U. Catholique de Louvain
Valter Balegas, U. Nova Lisboa

Geo-distributed DB

Fault tolerance
Fast reads
Replicate updates

CAP: CP ∩ AP = ∅

Strong Consistency

Azure Cosmos DB
Synchronous updates
Consistent under Partition; not Available
Application is correct
Slow, expensive

Eventual Consistency

Google Spanner
Asynchronous updates
Available under Partition; but not Consistent
Faster, less expensive
Concurrency anomalies
**FMK Fælles Medicinkort**

- Dr Alice
- Patient Bob
- Byrum pharma

Create: (...)
Add-med: (...)
Get-med: (...)
Process: (...)

Dr Alice
Aalborg Hospital
Patient: Mr Bob
Pharmacy: Byrum

Causatin: 2 boxes → 1

**Some invariants**

Dr Alice
Aalborg Hospital
Patient: Mr Bob
Pharmacy: Byrum

Causatin: 2 boxes → 1

Transactol: 1 box

**Availability + invariants?**

- Dr Alice
- Patient Bob
- Byrum pharma

Create: (...)
Add-med: (...)
Get-med: (...)
Process: (...)

Dr Alice
Aalborg Hospital
Patient: Mr Bob
Pharmacy: Byrum

Causatin: 2 boxes → 1

Transactol: 1 box

CP is overkill!
EC does not maintain!

**Approach**

Consistency: one size does not fit all
Correct: maintain invariants
- But often unknown!

Methodology:
- Preserve sequential patterns
- Synchronise only when strictly necessary for application

Best possible availability and performance
CRDTs

Concurrent, asynchronous updates
- Standard register model: assignments $\Rightarrow$ CP
- AP $\Rightarrow$ concurrent updates + merge

CRDT: register, counter, set, map, sequence
- Plug-in replacement for sequential type


Better than EC CP is overkill!

Pattern 1: relative order

Pattern 2: joint update

Create-p before add-med
- “Bob points to Rx $\Rightarrow$ Rx valid”
- General case: LHS $\Rightarrow$ RHS
- pattern: RHS!, LHS!

Deliver in the right order: Causal Consistency
AP-compatible

EC does not maintain!
CP is overkill!

EC does not maintain!
CP is overkill!
**All-or-nothing**

`create-p` updates doctor, patient & pharmacy record
Transmit joint updates together
+ Read from same snapshot
AP-compatible

**Pattern 3: precondition**

EC does not maintain!
CP is overkill!

CAP-sensitive invariants

pre-condition

```
process-p (... , nb)
|
if cnt ≥ nb  // precondition at source
   cnt ← nb  // at every replica
}{ // cnt ≥ 0
  Precondition stable w.r.t. concurrent add-med
  Concurrency OK
```

Stable precondition

```
process-p (... , nb)
|
if cnt ≥ nb  // precondition at source
   cnt ← nb  // at every replica
}{ // cnt ≥ 0
  Precondition stable w.r.t. concurrent add-med
  Concurrency OK
```
CAP-sensitive invariants

process-p (..., nb) {
  if cnt \geq nb // precondition at source
  cnt -= nb // at every replica
} // cnt \geq 0

Precondition not stable w.r.t. concurrent process-p
• Forbid concurrency? Synchro, CP.
• Or remove invariant? AP, degraded semantics

Not stable precondition

process-p (..., nb) {
  if cnt \geq nb // precondition at source
  cnt -= nb // at every replica
} // cnt \geq 0

Precondition not stable w.r.t. concurrent process-p
• Forbid concurrency
• Or, give up on invariant

Summary: Just-Right Consistency

Tailor consistency to application invariants
• (possibly unknown)
Baseline: Correct app, 1-copy, one op. at a time
Three patterns:
• Ordered updates \Rightarrow Causal Consistency, AP
• Joint updates \Rightarrow All-or-nothing, AP
• CAP-sensitive: precondition
  ‣ Stable \Rightarrow concurrent OK, AP.
  ‣ Otherwise, concurrency control, CP

CRDT data model
• Register, counter, set, map, sequence
• Extends sequential semantics

Transactional Causal Consistency (TCC)
• Strongest AP model
• Supports Joint Updates, Relative Order

CISE static analysis tools
Open source, well engineered