Just-Right Consistency

As available as possible
As consistent as necessary
Correct by design

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

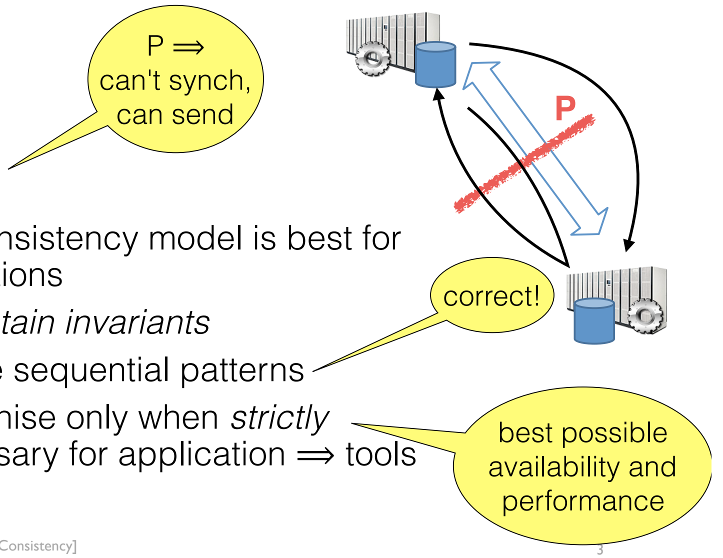
Bridging the CAP gap

CP ∩ AP = ∅

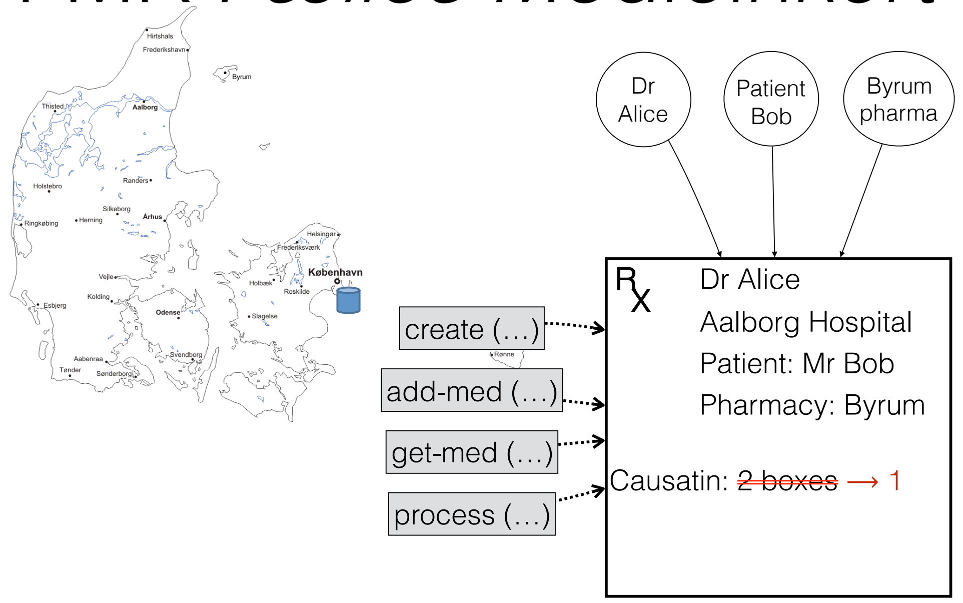
No single consistency model is best for all applications

Insight: *Maintain invariants*

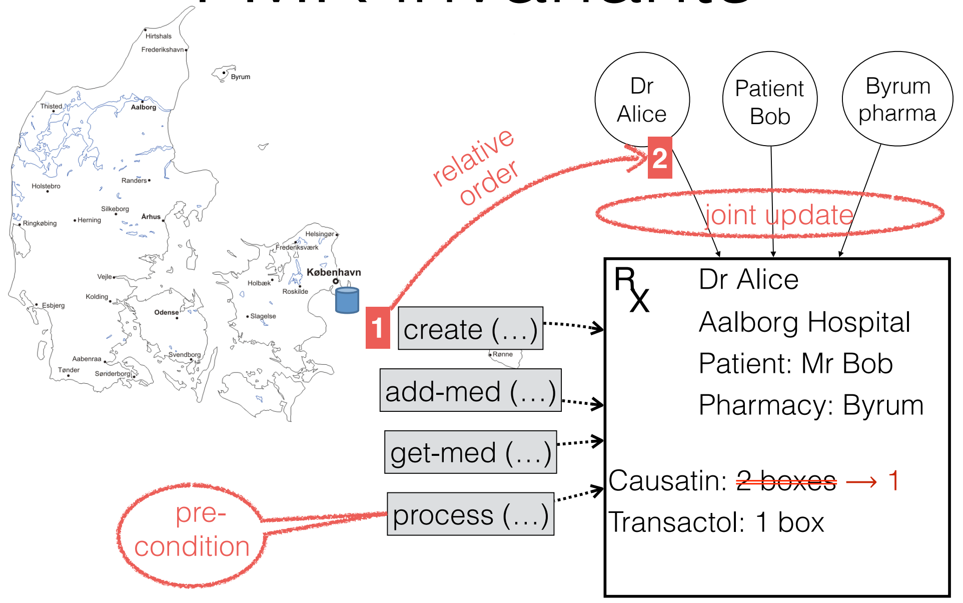
- Preserve sequential patterns
- Synchronise only when *strictly* necessary for application ⟹ tools


P ⟹ can't synch, can send
P
correct!
best possible availability and performance

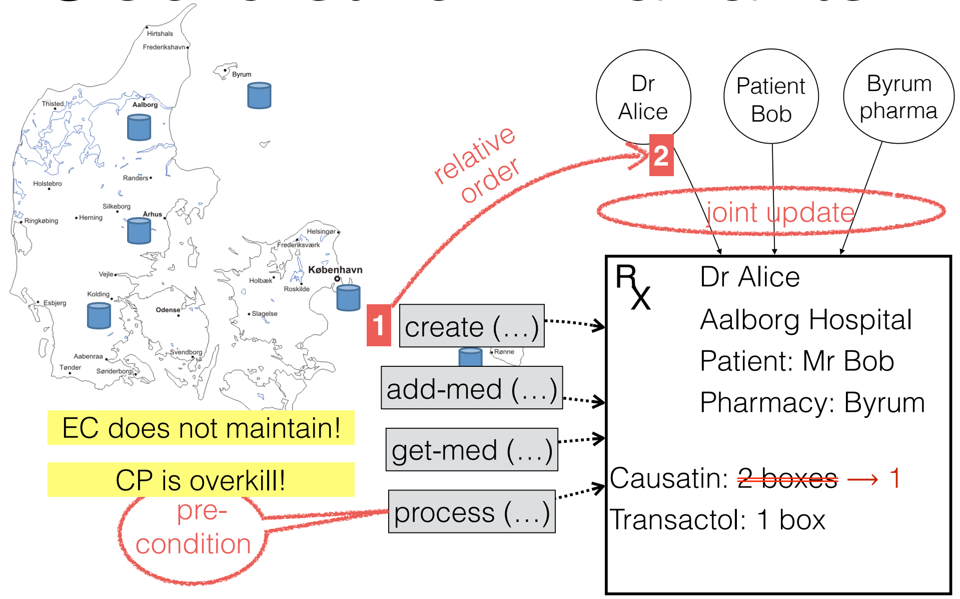
FMK *Fælles Medicinkort*



FMK invariants



Geo-distrib: invariants?



EC does not maintain!

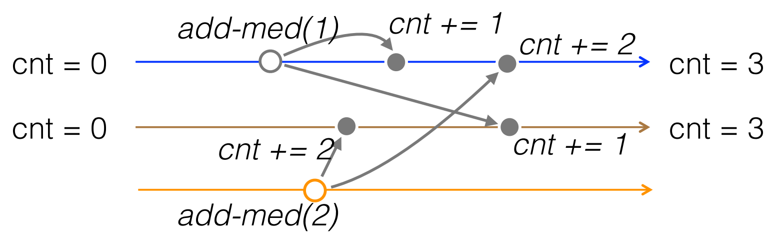
CP is overkill!

AP-compatible programming constructs

Available under partition
- ⟹ no synchronisation
- ⟹ asynchronous updates
- ⟹ fast response

AP-compatible:
- CRDT data model
- Relative-order pattern
- Joint-update pattern

AP data model: CRDTs

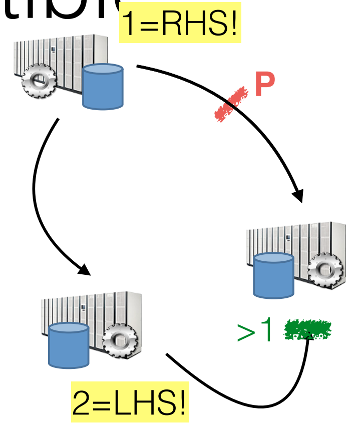


Concurrent, asynchronous updates
- Standard register model: assignments ⟹ CP
- AP ⟹ concurrent updates merged

CRDT: register, counter, set, map, sequence
- Extends sequential type
- Encapsulates convergent merge

without CC animation

Relative order is AP-Compatible



create-p before *add-pp*
- Referential integrity
  - *x* valid ∧ *x* points to *y* ⟹ *y* valid
- *admin-login-enabled* ⟹ *non-default-password*

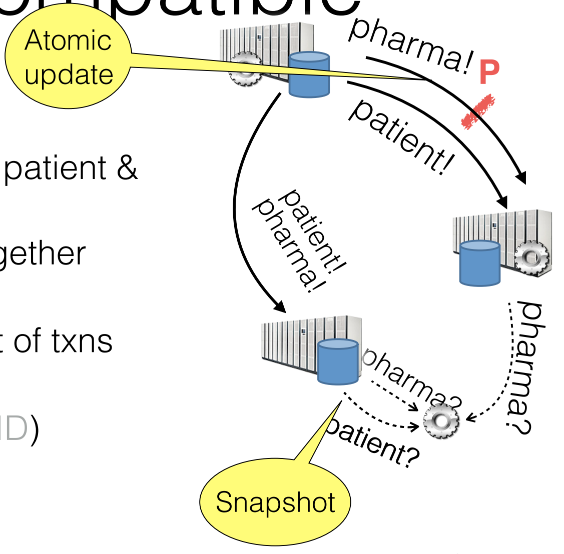
`RHS ≔ true; LHS ≔ true`

Transmit in the right order!

AP-compatible: Causal Consistency

without animations

Joint update is AP-Compatible



create-p updates doctor, patient & pharmacy record

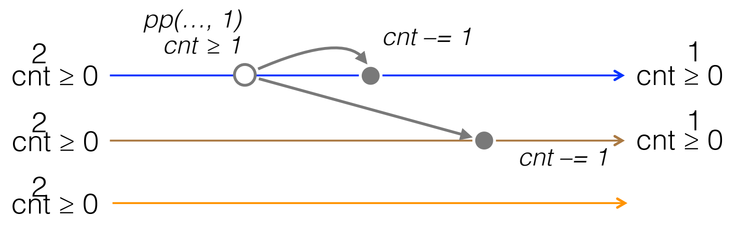
Transmit joint updates together
- write-atomic

+ Read from common set of txns
- snapshot property

= All-or-Nothing (A of ACID)

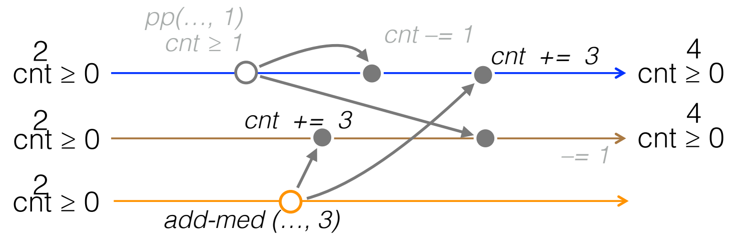
AP-compatible

CAP-sensitive invariants



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

CAP-sensitive invariants

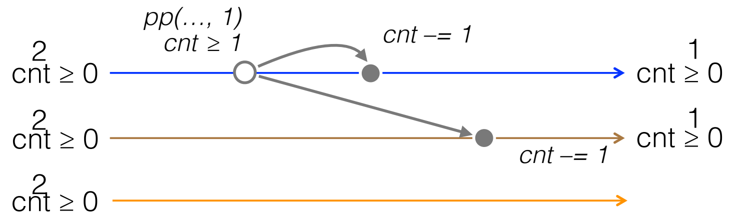


```
process-p (…, nb) {
  if $cnt \geq nb$      // precondition at source
    $cnt -= nb$         // at every replica
} // $cnt \geq 0$
```

Precondition *stable* w.r.t. concurrent *add-med*

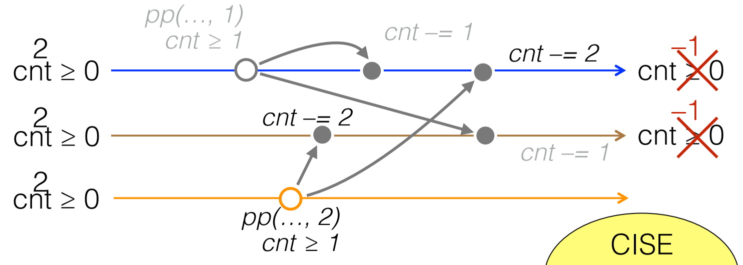
Concurrency OK

CAP-sensitive invariants



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

CAP-sensitive invariants



CISE Static Analysis

```
process-p (…, nb) {
  if cnt ≥ nb     // precondition at source
    cnt −= nb       // at every replica
} // cnt ≥ 0
```

Precondition *not stable* w.r.t. concurrent *process-p*

- Forbid concurrency? Synchro, CP.
- Or remove invariant? AP, degraded semantics

CISE tools

Static analysis of any application:

- Operations, invariants
- Does each individual op maintain invariant?
- Do concurrent updates converge?
- Is precondition of *u* stable w.r.t. concurrent *v*?
  If not:
  - Change specification (~~invariant~~)
  - or Synchronise
  - Designer decision, per pair *(u, v)*

Ex: medication count= *inc*||*inc*, *inc*||*dec*, ~~*dec*||*dec*~~

Just-Right Consistency

Methodology for provably ensuring
As Available as Possible, Consistent Enough

TCC ⟹ AP-compatible invariants

CAP-sensitive invariants: Bounded Ctr, CISE

AntidoteDB:

⟹ AP-compatible

- CRDTs
- Causal Consistency
- Transactions
- Bounded Counter

mostly AP

CP when necessary

CISE verification
& co-design

(+ related tools)

AntidoteDB.eu

CRDT data model

- Register, counter, set, map, sequence
- Extend sequential semantics
- AP compatible

Transactional Causal Consistency (TCC)

- Strongest AP-compatible model
- Joint Updates / Transactional
- Partial Order / Causal Consistency

Open source, well engineered

Community of users