Bigsets are awesome

My Helsing connection

Russell Brown 09/10/2025

What Even is a Bigset?

And what has it to do with anything?

CRDTs

CRDT Sets

Real Life vs Papers

Bigsets (2016)

Bigsets->DSON

Bigsets-rs (2025)

Definition

A conflict-free replicated data type (CRDT) is an abstract data type, with a well defined interface, designed to be replicated at multiple processes and exhibiting the following properties: (i) any replica can be modified without coordinating with another replicas; (ii) when any two replicas have received the same set of updates, they reach the same state, deterministically, by adopting mathematically sound rules to guarantee state convergence.

https://arxiv.org/abs/1805.06358

Conflict-free Replicated Data Types. Nuno Preguiça,Carlos Baquero,Marc Shapiro

SOSP 2007 Dynamo

Amazon Shopping Cart

from Noun Project

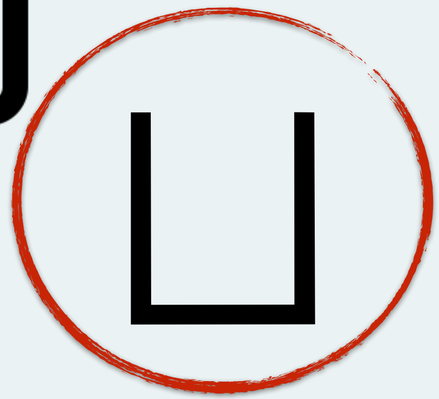
Created by Amy Schwartz
from the Noun Project

Created by Dara Ullrich
from Noun Project

Created by Amy Schwartz
from the Noun Project

Created by Amy Schwartz
from the Noun Project

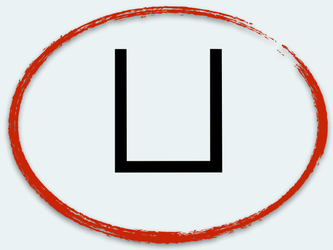
Created by Amy Schwartz
from the Noun Project



Created by Dara Ullrich
from Noun Project

Created by Amy Schwartz
from the Noun Project

Created by Amy Schwartz
from the Noun Project



Created by Amy Schwartz
from the Noun Project

What about removes?

Created by Amy Schwartz
from the Noun Project

```rust
/// basic add wins set
#[derive(Clone, Debug)]
struct AddWinsSet {
    adds: BTreeMap<Bytes, BTreeSet<u64>>,
    removes: BTreeMap<Bytes, BTreeSet<u64>>,
}
```

```rust
fn add(&mut self, elem: Bytes, tag: u64) {
    if let Some(tags) = self.adds.get(&elem) {
        self.removes.entry(elem.clone()).or_default().extend(tags);
    }

    self.adds.entry(elem).or_default().insert(tag);
}
```

```rust
fn remove(&mut self, elem: Bytes) {
    if let Some(tags) = self.adds.get(&elem) {
        self.removes.entry(elem).or_default().extend(tags);
    }
}
```

```rust
fn members(&self) -> BTreeSet<Bytes> {
    let mut members = BTreeSet::new();
    for (elem, tags) in self.adds.iter() {
        if self.removes.get(elem).map_or(true, |r| !tags.is_subset(r)) {
            members.insert(elem.clone());
        }
    }
    members
}
```

```rust
fn merge(&mut self, other: Self) {
    for (elem, tags) in other.removes {
        self.removes.entry(elem.clone()).or_default().extend(tags);
    }
    for (elem, tags) in other.adds {
        self.adds.entry(elem.clone()).or_default().extend(tags);
    }
}
```

Created by Amy Schwartz
from the Noun Project

Merge

Created by Amy Schwartz
from the Noun Project

Created by Amy Schwartz

Optimisations

Add Wins Sets Without Tombstones

Delta Replication (various flavours)

Papers: One CRDT in memory

Paper vs Production

One CRDT in Memory vs Many CRDTs on Disk

Deltas are great on the network BUT quadratic on disk

(Read-Deserialise-Mutate-Serialise-Write)

(Send-Delta)

(Read-Deserialise-Apply Delta-Serialise-Write)

Bigsets

Decompose CRDT elements on Disk

Read minimum
Generate Delta
Write minimum

Send Delta
Read minimum
Apply Delta

Write minimum

(Trade-Off? Full Set Read is Slow (but enables querying over elements))

Bigset -> DSON

- Bigsets “paper” https://dl.acm.org/doi/10.1145/2911151.2911156
- Δ-CRDTs: making δ-CRDTs delta-based https://dl.acm.org/doi/10.1145/2911151.2911163
- Efficient Synchronisation of State-based CRDTs https://arxiv.org/abs/1803.02750
- DSON: JSON CRDT using delta-mutations for document stores https://dl.acm.org/doi/10.14778/3510397.3510403

Bigsets-rs

https://github.com/russelldb/bigsets-rs