Multi-view data types

Scalable concurrency in the multi-core era

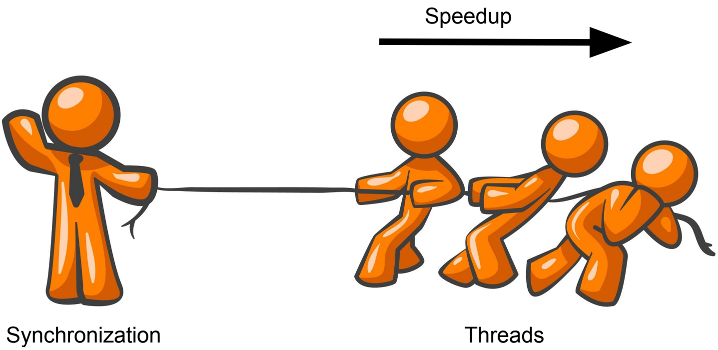
Deepthi Akkoorath[1], José Brandão[2], Annette Bieniusa[1], Carlos Baquero[3]

[1]Technical University of Kaiserslautern
Germany

[2]Universidade do Minho
Braga, Portugal

[3]HASLab, Universidade do Minho & INESC TEC
Braga, Portugal

Concurrent programs in multi-core



Overview

Distributed systems

- Eventual consistency + CRDTs ➔ Synchronisation free
- Fast, Scalable, Available

Goal

- Weak consistency ➔ Less synchronisation
- Speed up!

Overview

Distributed systems

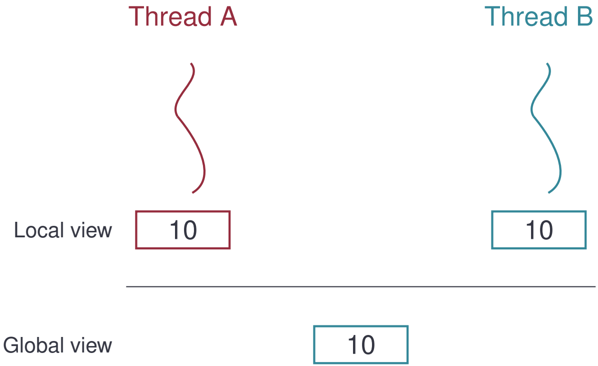
- Eventual consistency + CRDTs ➔ Synchronisation free
- Fast, Scalable, Available

Goal

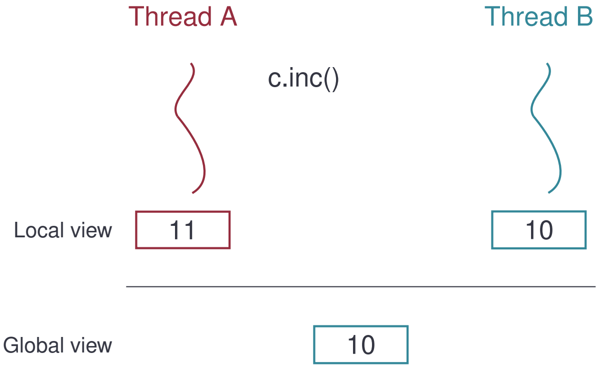
- Weak consistency ➔ Less synchronisation
- Speed up!

Global-Local view model
Multi-view data types

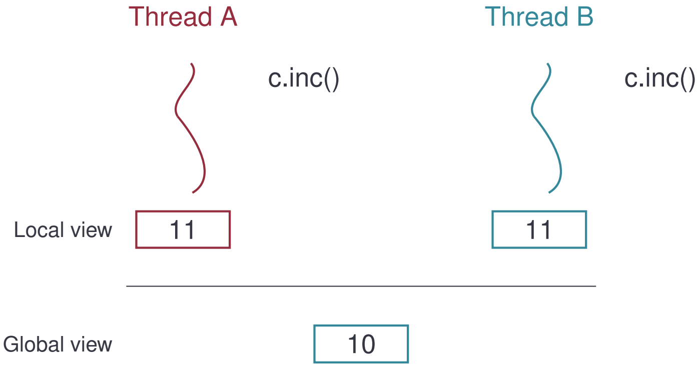
Global-local view model



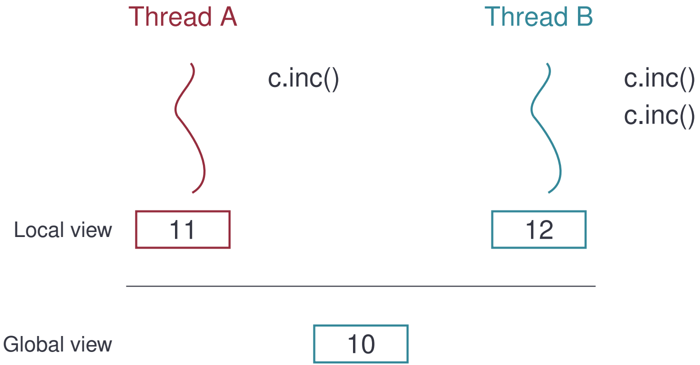
Global-local view model



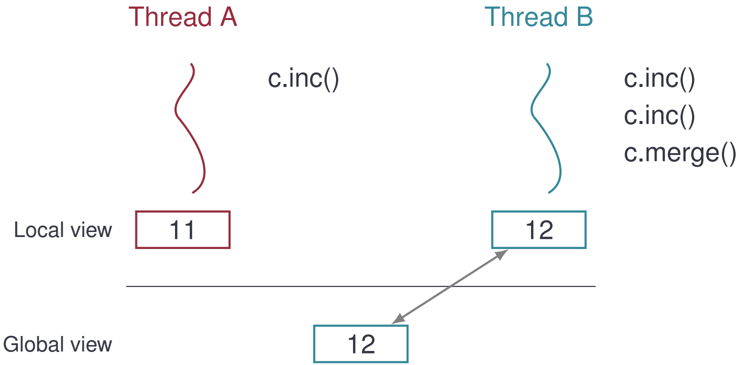
Global-local view model



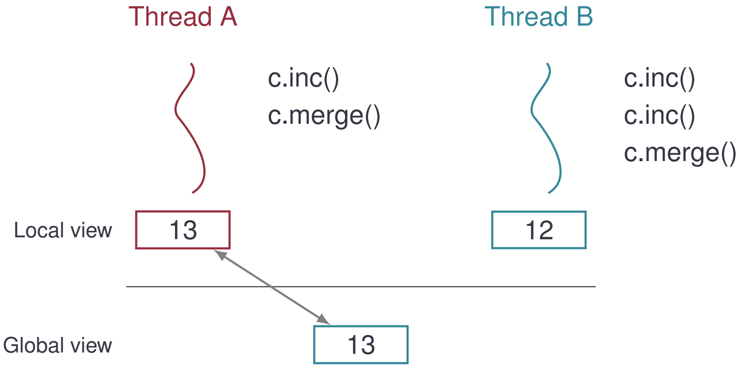
Global-local view model



Global-local view model



Global-local view model



Operations

Local View: (snapshot, local)

- snapshot: Global view that was last seen by this thread
- local: Accumulates local updates

Global View: g

Operations

weakUpdate
weakRead

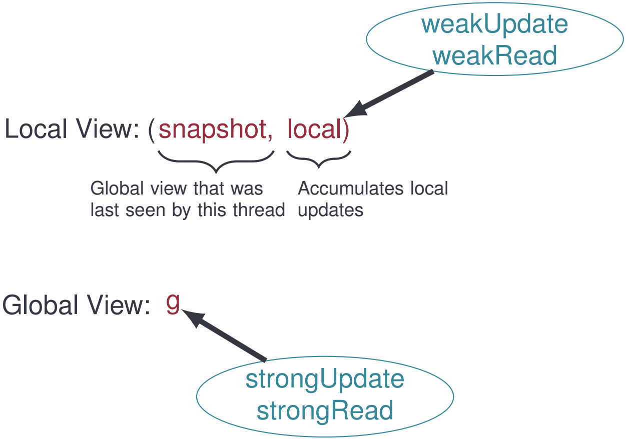
Local View: (snapshot, local)

Global view that was last seen by this thread

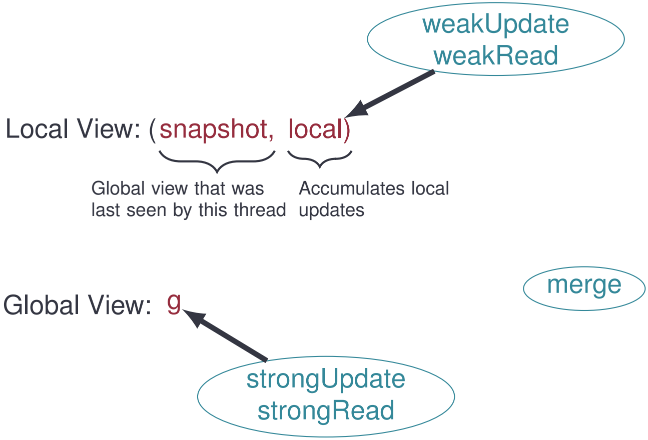
Accumulates local updates

Global View: g

Operations


weakUpdate
weakRead
Local View: (snapshot, local)
Global view that was last seen by this thread
Accumulates local updates
Global View: g
strongUpdate
strongRead

Operations



weakUpdate
weakRead
Local View: (snapshot, local)
Global view that was last seen by this thread
Accumulates local updates
merge
Global View: g
strongUpdate
strongRead

Multi-view data types

Mergeable types

- Implements weak operations and merge

Hybrid types

- Implements weak, strong and merge operations
- Hybrid counter
  synchronous increment when close to a target
- Hybrid queue
  weak enqueue and synchronous dequeue

CRDTs?

CRDTs?

- G-Set
  - merge = union of sets
- Counter
  - Map: id $\rightarrow$ int
  - merge = max of each elem

CRDTs?

- G-Set
  - merge = union of sets
- Counter
  - Map: id $\rightarrow$ int
  - merge = max of each elem

CRDT merge is expensive

CRDTs?

- G-Set
  - merge = union of sets
- Counter
  - Map: id $\rightarrow$ int
  - merge = max of each elem

CRDT merge is expensive

Multi-view data types

- Multiple versions (view)
- Isolated access to each view
- Fast merge

Counter

- Global view: **int** g
- Local view :
  (**int** s, **int** l)

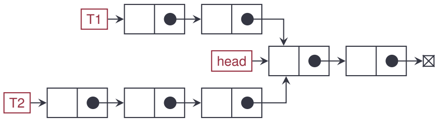
  Thread-local copies

  Exclusive access ⇒ no synchronization

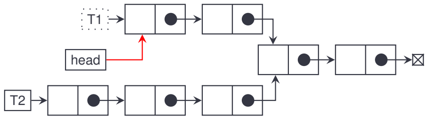
  Synchronous merge

```
weakInc() {
  l++;
}
weakValue(){
  return s+l;
}
merge(){
  atomic {g += l;
          s = g; l = 0;}
}
strongInc() {
        atomic {g++;}
}
```

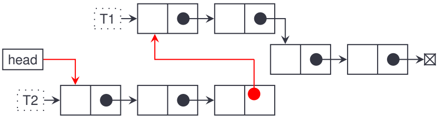
Multi-view list



After $T1$ commits:



After $T2$ commits:

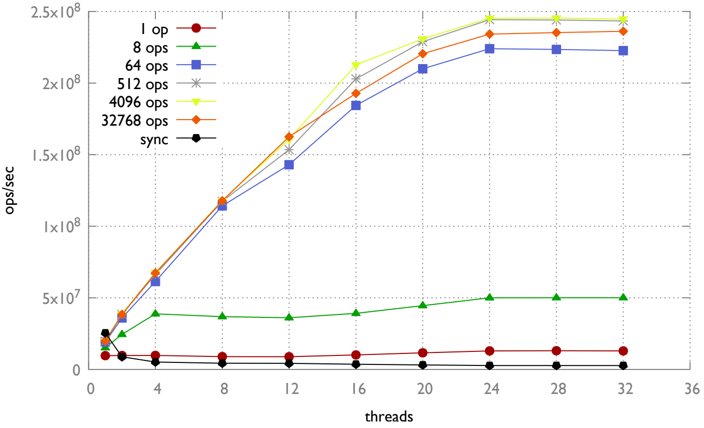


Evaluation: Hybrid Counter

Goal: increment until a target

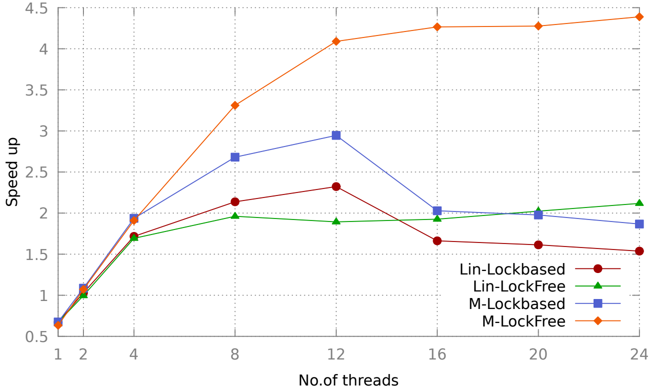
Periodic merge ⇒ Divergence from target

Switches to strong update after a threshold



Evaluation: Breadth first traversal

Using hybrid queue : weak enqueue and strong dequeue



Related work

Mergeable types

Doppel [Narula et al., 2014]

in-memory transactions

Concurrent revisions [Burckhardt et al., 2010]

fork join model

“mergeable” types

Weak consistency

Quasi linearizability [Afek et al., 2010]

Weak/medium future linearizability [Kogan and Herlihy, 2014]

K-linearizability [Aiyer et al., 2005]

Quiescent consistency [Aspnes et al., 1994]

Summary

Global-local view model

- fast local state, distant global state

Impact on underlying data structure design

- Multiple versions, Merge

Combination of weak and strong updates

- A spectrum of consistency

Thank you!

akkoorath@cs.uni-kl.de

References I

Afek, Y., Korland, G., and Yanovsky, E. (2010).
Quasi-linearizability: Relaxed consistency for improved concurrency.
In *Proceedings of the 14th International Conference on Principles of Distributed Systems*, OPODIS'10, pages 395–410, Berlin, Heidelberg. Springer-Verlag.

Aiyer, A., Alvisi, L., and Bazzi, R. A. (2005).
On the availability of non-strict quorum systems.
In *Proceedings of the 19th International Conference on Distributed Computing*, DISC'05, pages 48–62, Berlin, Heidelberg. Springer-Verlag.

Aspnes, J., Herlihy, M., and Shavit, N. (1994).
Counting networks.
J. ACM, 41(5):1020–1048.

Burckhardt, S., Baldassin, A., and Leijen, D. (2010).
Concurrent programming with revisions and isolation types.
In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '10, pages 691–707, New York, NY, USA. ACM.

References II

Kogan, A. and Herlihy, M. (2014).
The future(s) of shared data structures.
In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, PODC '14, pages 30–39, New York, NY, USA. ACM.

Narula, N., Cutler, C., Kohler, E., and Morris, R. (2014).
Phase reconciliation for contended in-memory transactions.
In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 511–524, Berkeley, CA, USA. USENIX Association.