The Rotating Skip List: A New Non-Blocking Skip List

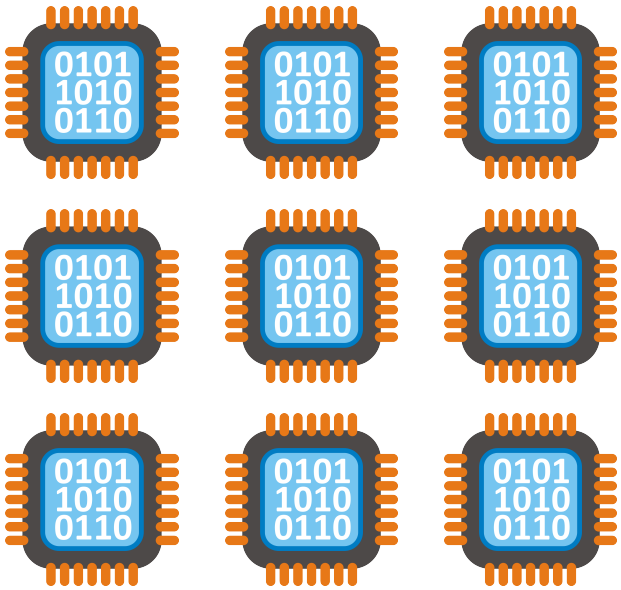
FACULTY OF ENGINEERING & INFORMATION TECHNOLOGIES

Ian Dick | Honours Student
Dr Vincent Gramoli | Supervisor
School of Information Technologies
The University of Sydney, Sydney, Australia

THE UNIVERSITY OF SYDNEY

THE UNIVERSITY OF SYDNEY

Background



intel

AMD

IBM

PYCOMALL.COM

- Uniprocessor speed is plateauing
- Number of cores on processors increasing at an exponential rate [1]
- We need programs that scale well with the number of available cores

THE UNIVERSITY OF SYDNEY

Mutual Exclusion Does Not Scale

- ***Blocking***
- ***Deadlock***
- ***Cache bouncing***

› Traditional method of synchronising threads is using mutual exclusion locks

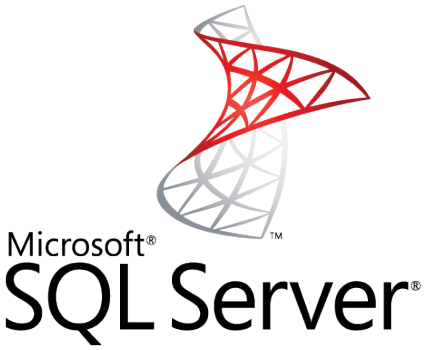
› In highly parallel environments mutual exclusion does not scale

› Focus on ***lock-free*** algorithms and data structures

Non-Blocking Lock-Free Synchronisation

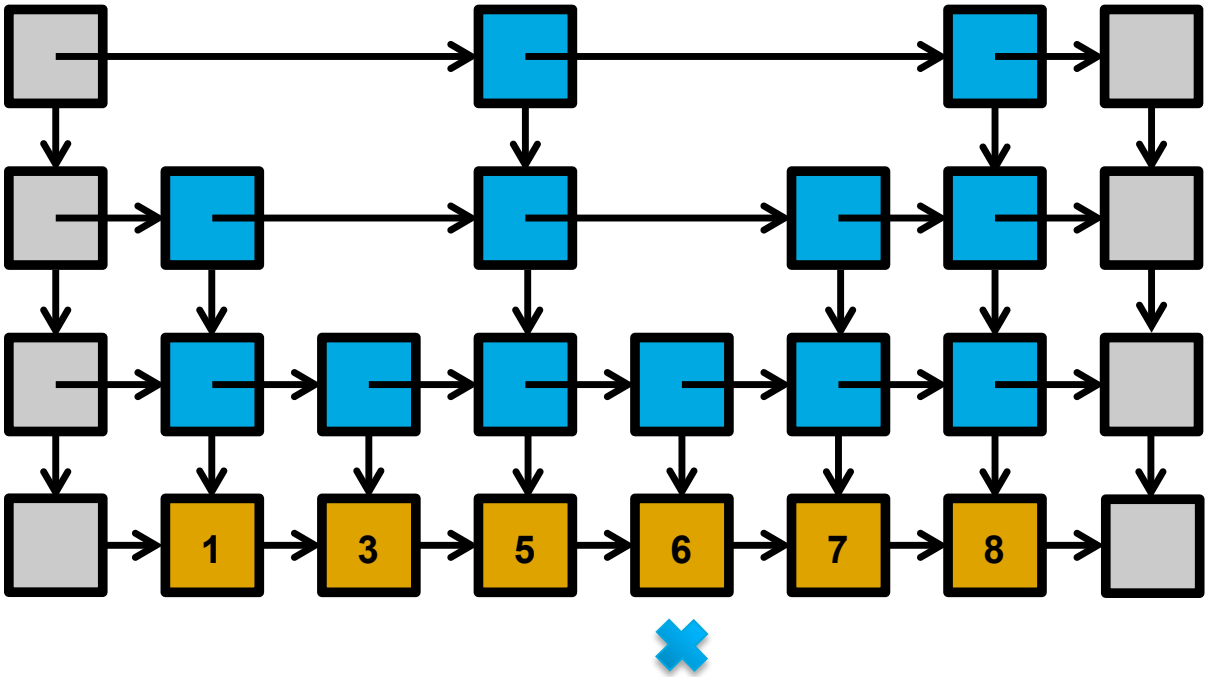
```
CompareAndSwap(x, a, b):
    if (*x = a) then
        x ← b
        return SUCCEED
    else
        return FAIL
```

- ***Forward progress guaranteed***
- ***Threads never block waiting for shared resources***



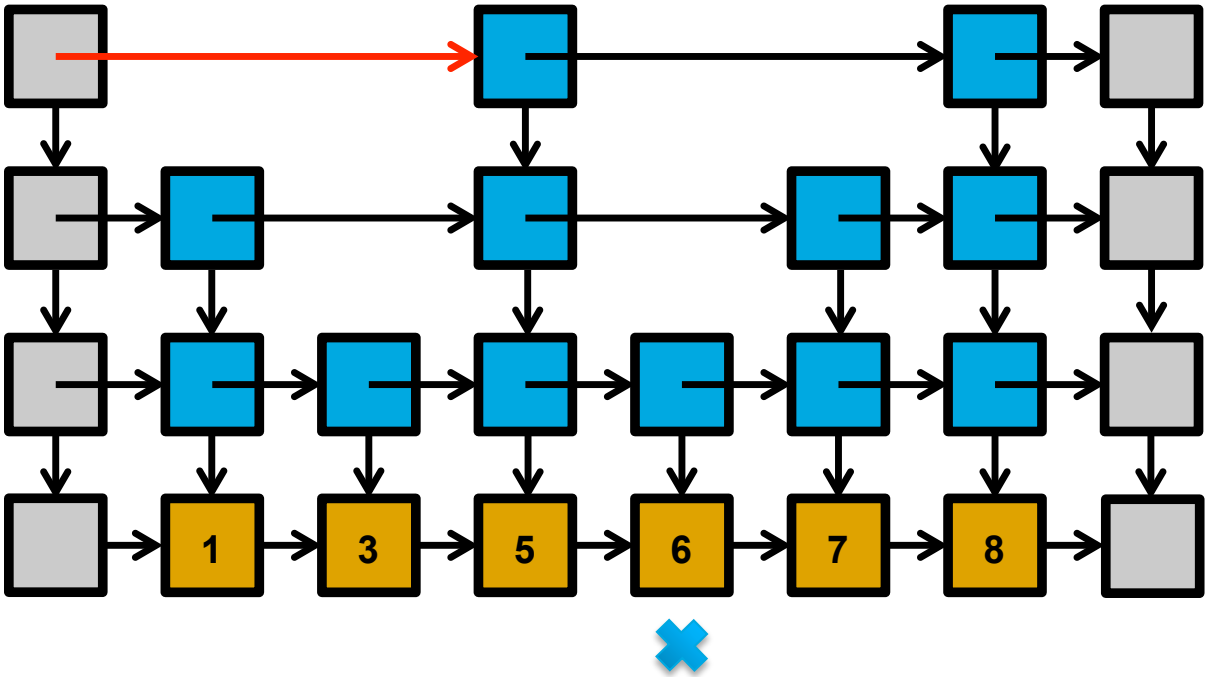
- Atomic Primitives such as compare-and-swap allow synchronisation to be non-blocking and lock-free
- Can avoid the issues associated with locking
- Non-blocking data structures being used in production systems

Skip Lists



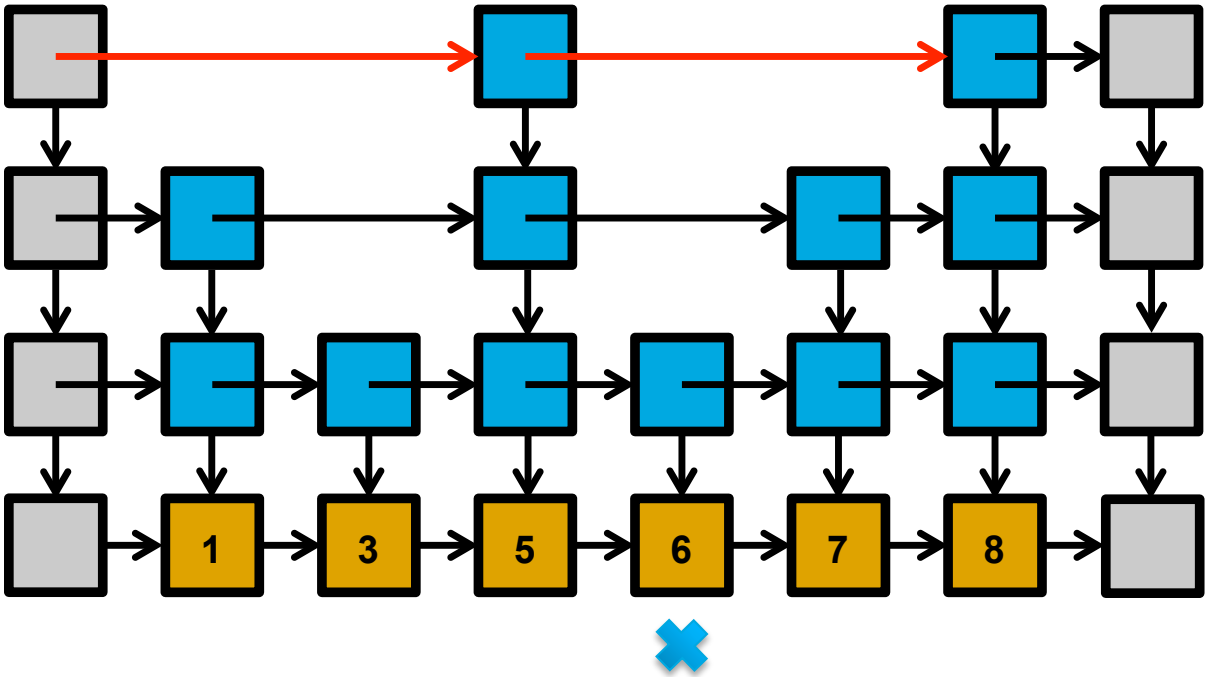
› Skip Lists facilitate logarithmic query times while maintaining a sorted order – alternative to B-Tree as a database indexing structure

Skip Lists



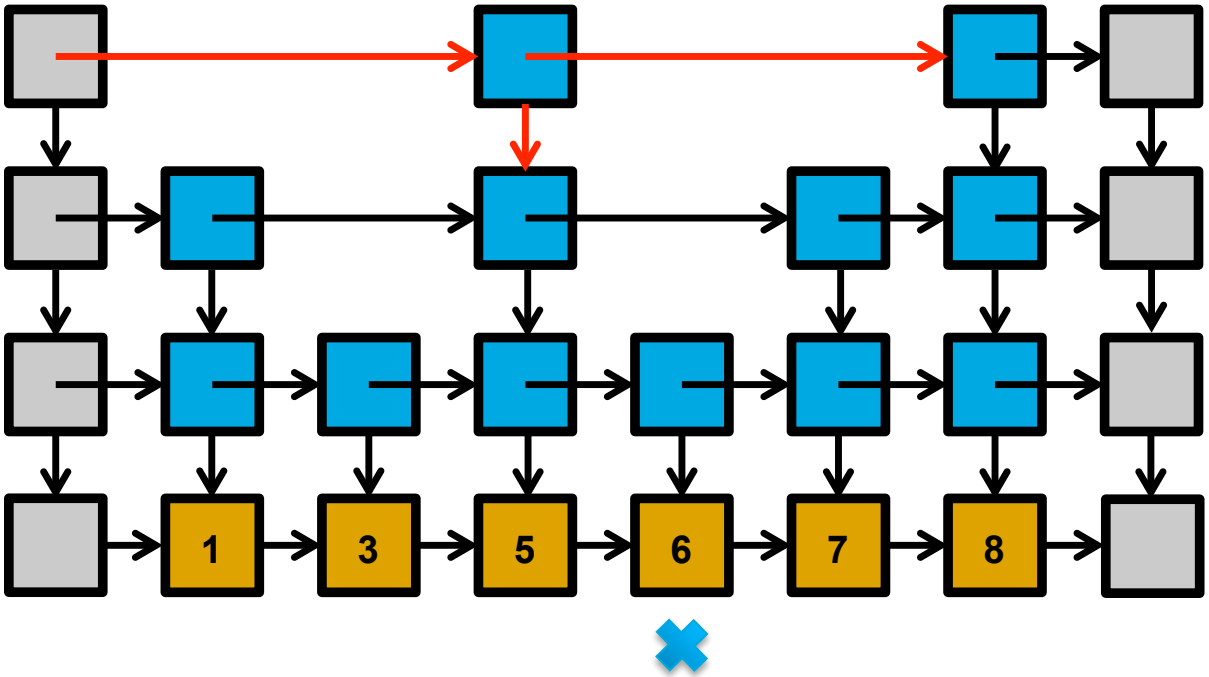
› Skip Lists facilitate logarithmic query times while maintaining a sorted order – alternative to B-Tree as a database indexing structure

Skip Lists



› Skip Lists facilitate logarithmic query times while maintaining a sorted order – alternative to B-Tree as a database indexing structure

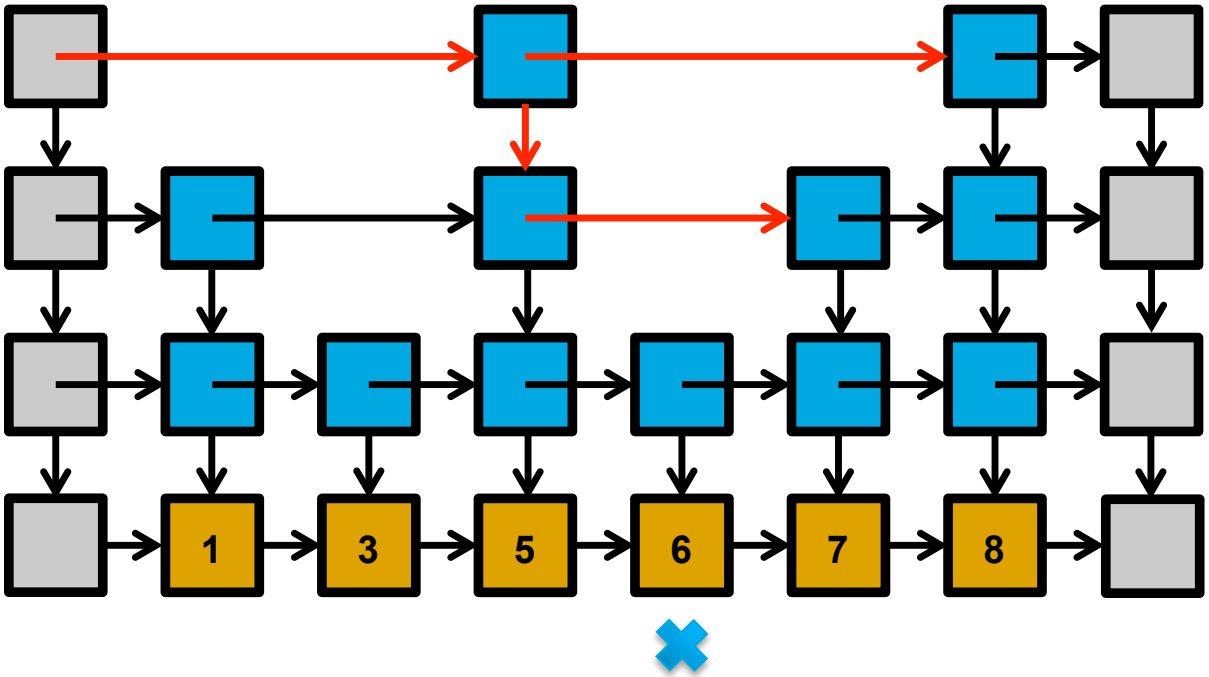
Skip Lists



› Skip Lists facilitate logarithmic query times while maintaining a sorted order – alternative to B-Tree as a database indexing structure

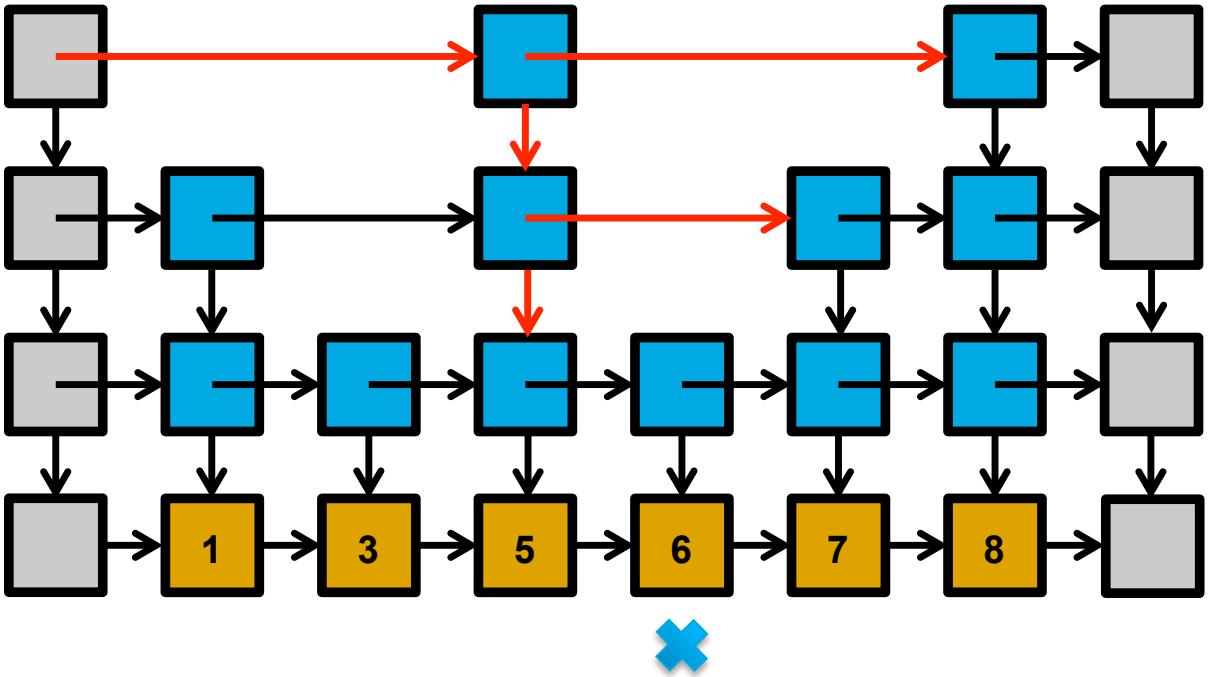
THE UNIVERSITY OF SYDNEY

Skip Lists



› Skip Lists facilitate logarithmic query times while maintaining a sorted order – alternative to B-Tree as a database indexing structure

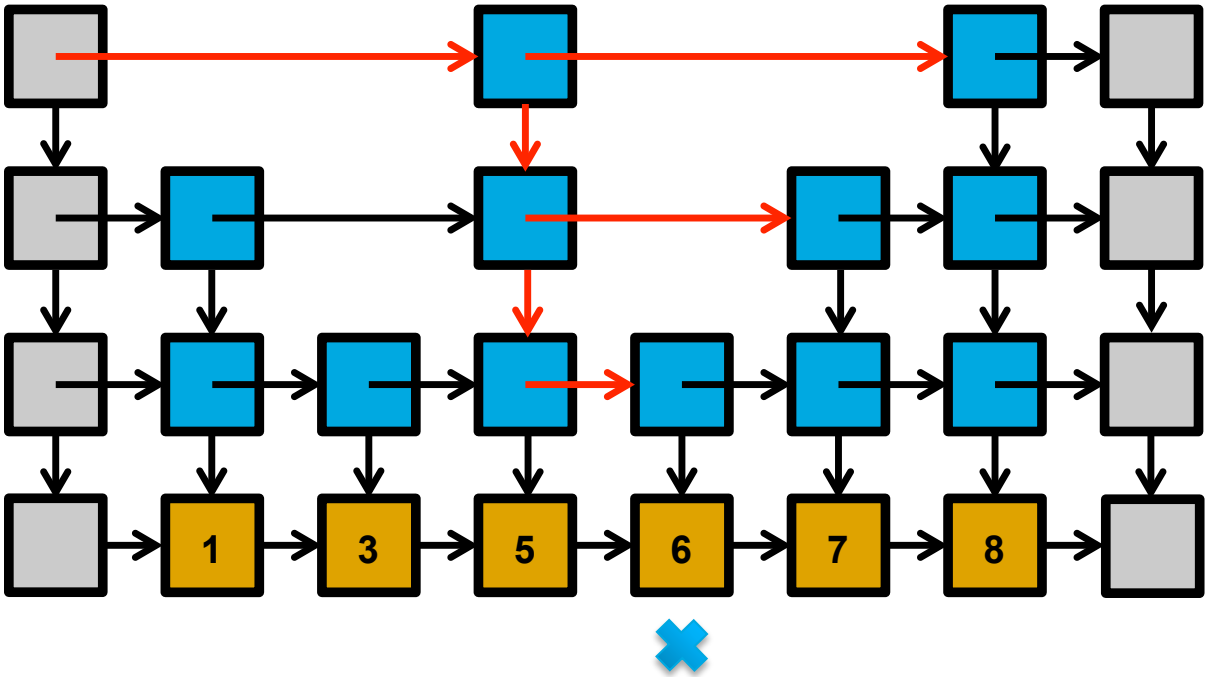
Skip Lists



› Skip Lists facilitate logarithmic query times while maintaining a sorted order – alternative to B-Tree as a database indexing structure

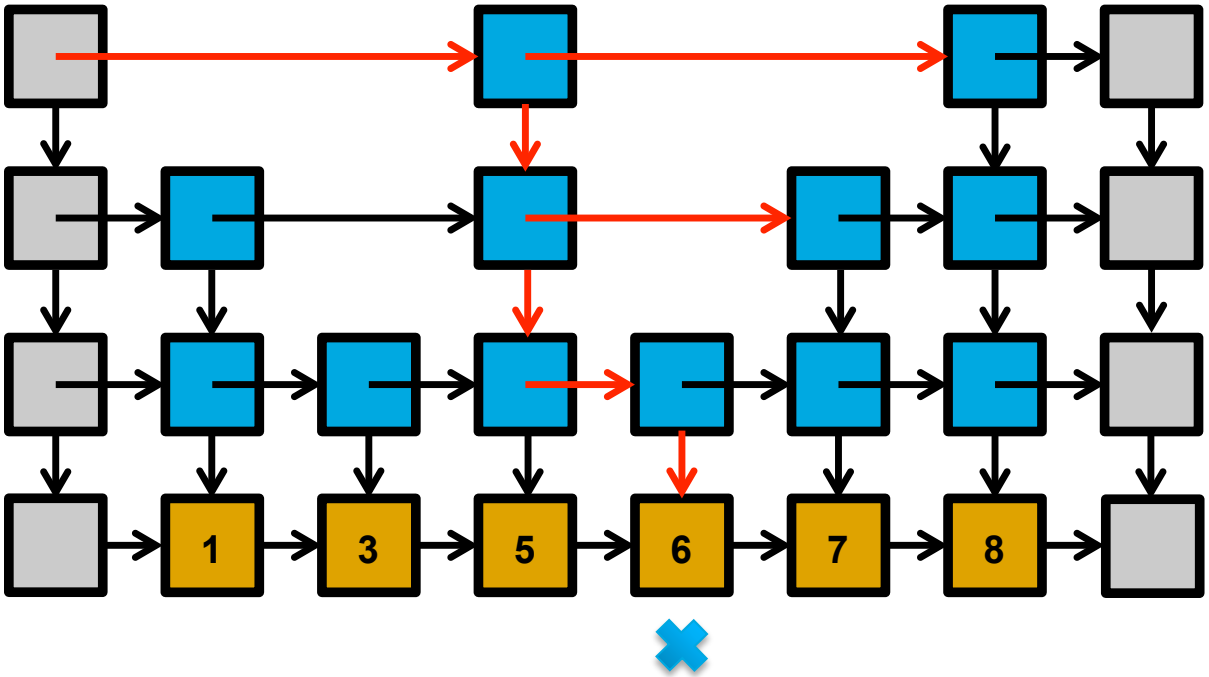
THE UNIVERSITY OF SYDNEY

Skip Lists



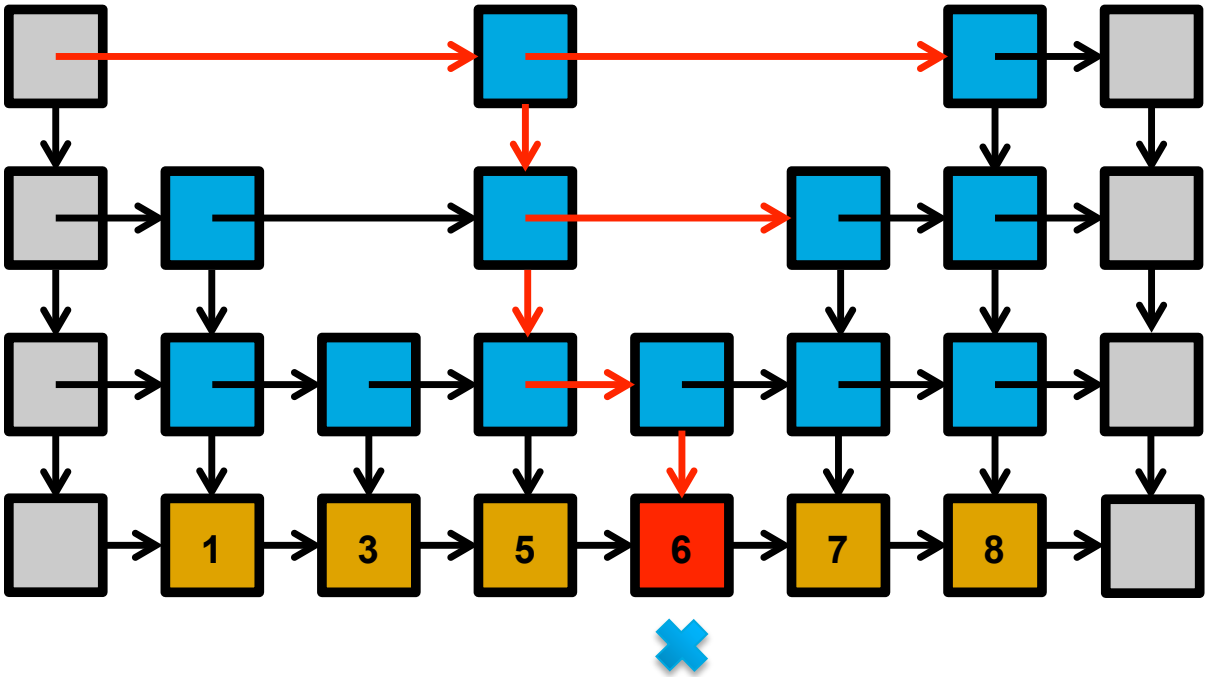
› Skip Lists facilitate logarithmic query times while maintaining a sorted order – alternative to B-Tree as a database indexing structure

Skip Lists



› Skip Lists facilitate logarithmic query times while maintaining a sorted order – alternative to B-Tree as a database indexing structure

Skip Lists



memsql

ArangoDB

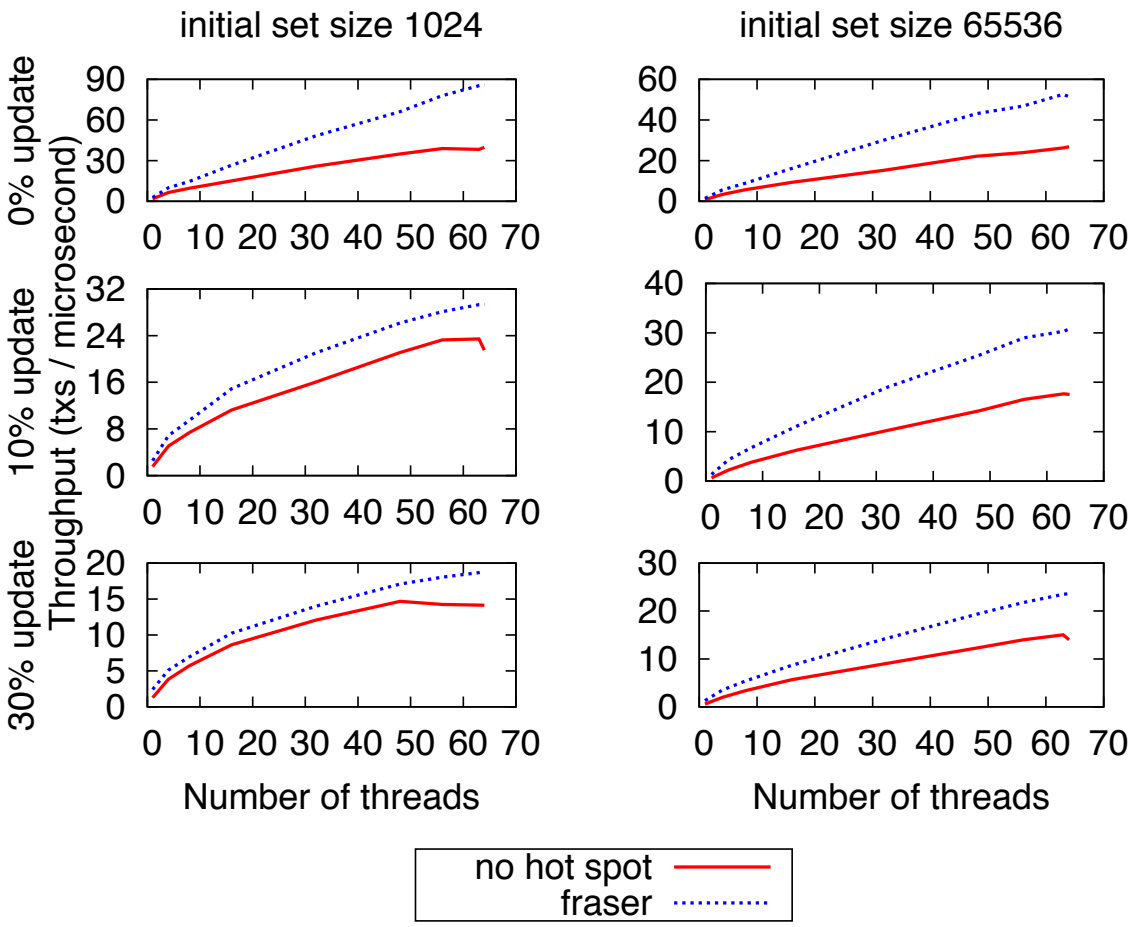
- Skip Lists facilitate logarithmic query times while maintaining a sorted order – alternative to B-Tree as a database indexing structure
- Well-suited to non-blocking synchronisation
- Production databases are already using skip lists

The No Hot Spot Non-Blocking Skip List

- Proposed by Crain, Gramoli and Raynal [2]
  - Appeared in ICDCS 2013
- Public skip list interface decoupled from maintenance
  - Background thread conducts skip list maintenance
  - Index level modifications are deterministic and single-threaded
- Outperforms Doug Lea's *ConcurrentSkipListMap* from the JDK

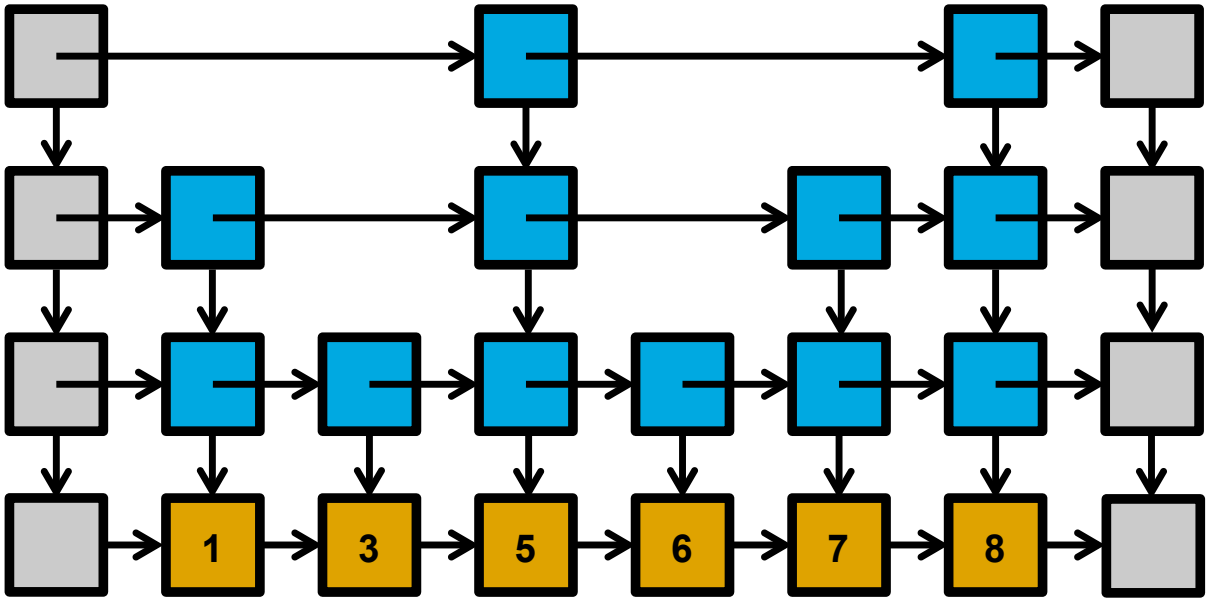
The No Hot Spot Non-Blocking Skip List

- I implemented the No Hot Spot Non-Blocking Skip List in C using the Java design specifications
- Benchmarked against Fraser's non-blocking skip list [3]
- Poor performance



THE UNIVERSITY OF SYDNEY

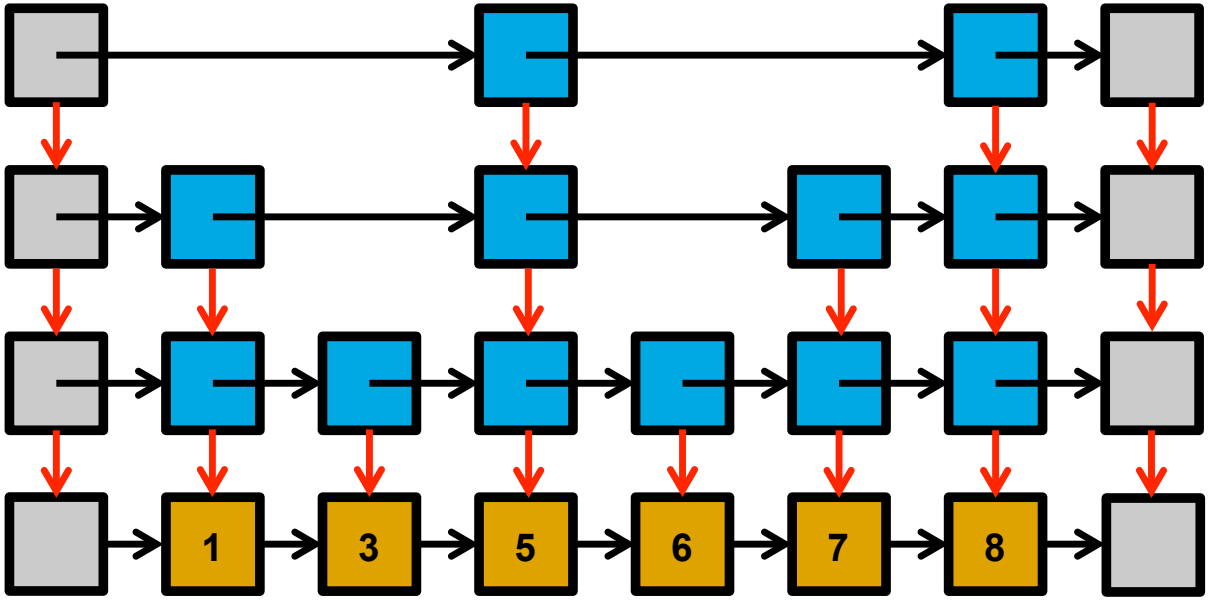
The Rotating Skip List



› Skip list using the no hot spot trick and optimised for the C language

THE UNIVERSITY OF SYDNEY

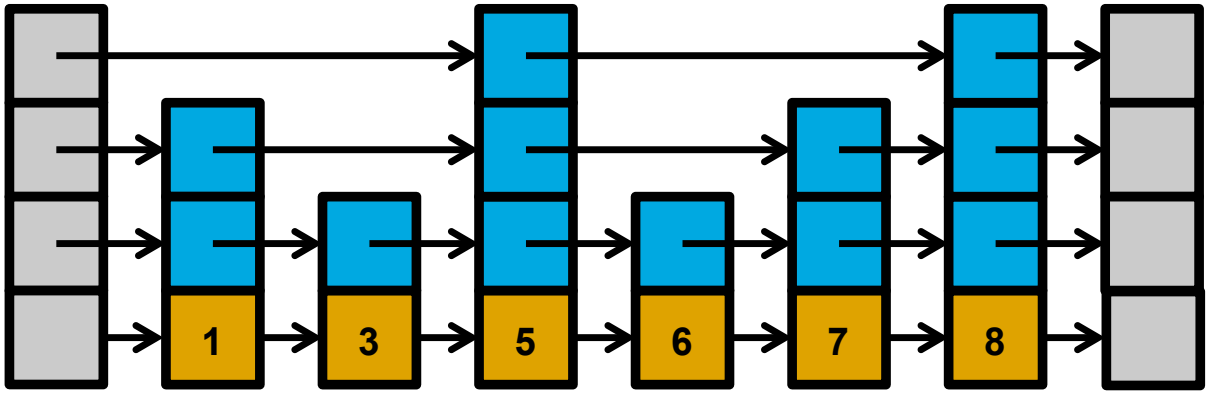
The Rotating Skip List



- Skip list using the no hot spot trick and optimised for the C language
- Index towers consolidated into rotating arrays

THE UNIVERSITY OF SYDNEY

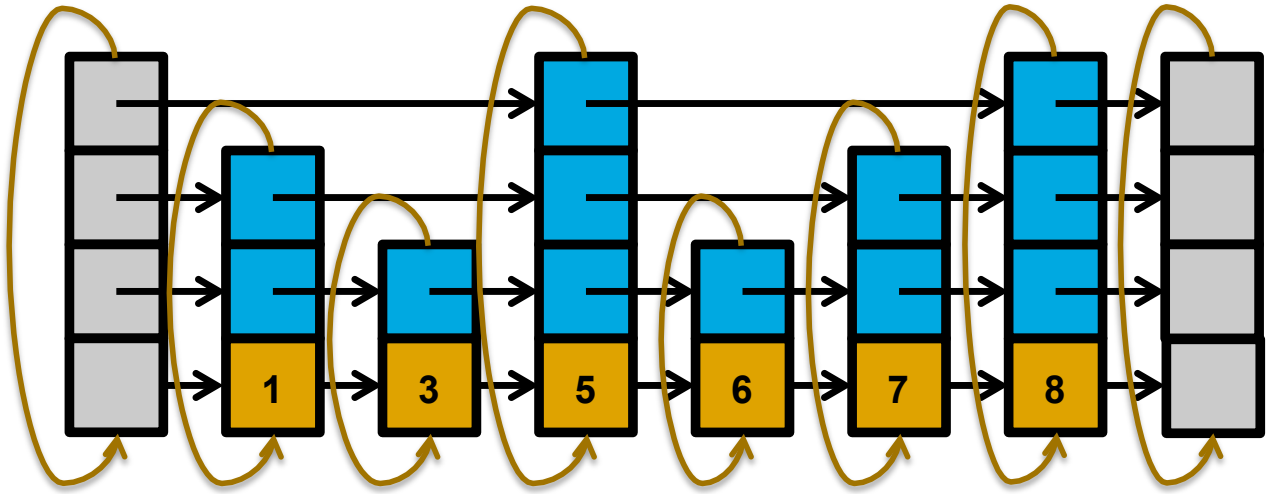
The Rotating Skip List



- Skip list using the no hot spot trick and optimised for the C language
- Index towers consolidated into rotating arrays

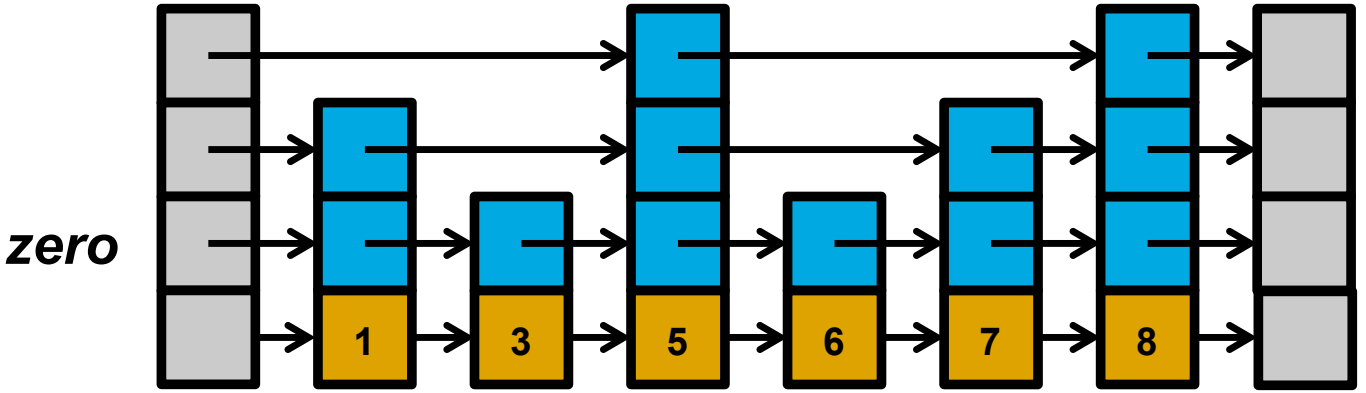
THE UNIVERSITY OF SYDNEY

The Rotating Skip List



- Skip list using the no hot spot trick and optimised for the C language
- Index towers consolidated into rotating arrays
- Background thread sleeps between maintenance iterations
- Worker threads do *simple* deletes

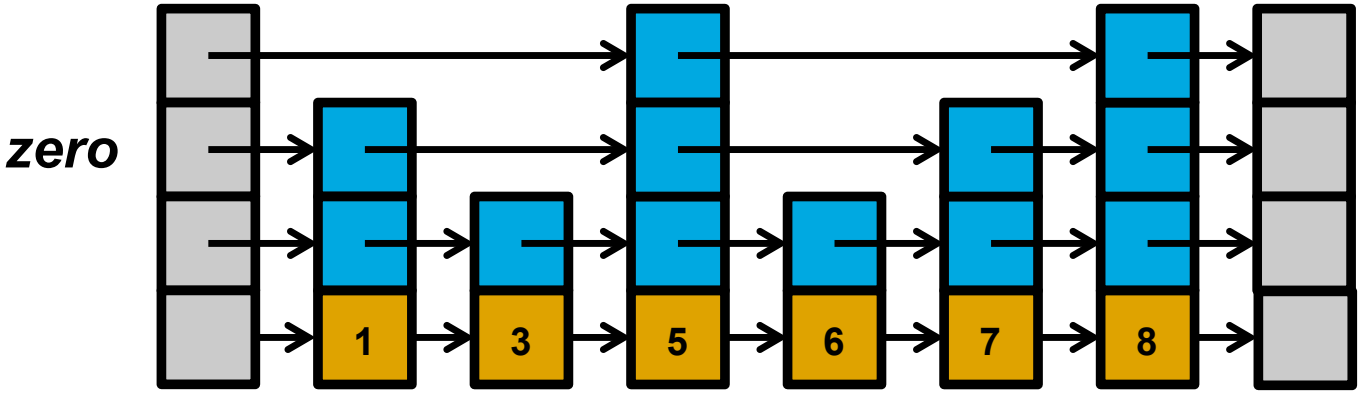
The Rotating Skip List




```
next = array[(zero + i) % N];
```

› Lowering of the Rotating Skip List is done by incrementing a global zero index

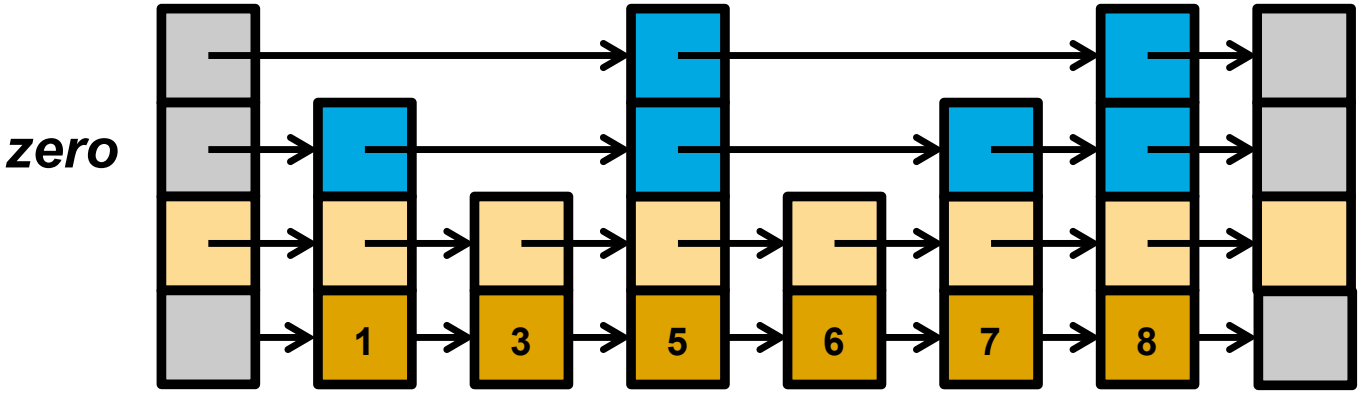
The Rotating Skip List




```
next = array[(zero + i) % N];
```

› Lowering of the Rotating Skip List is done by incrementing a global zero index

The Rotating Skip List

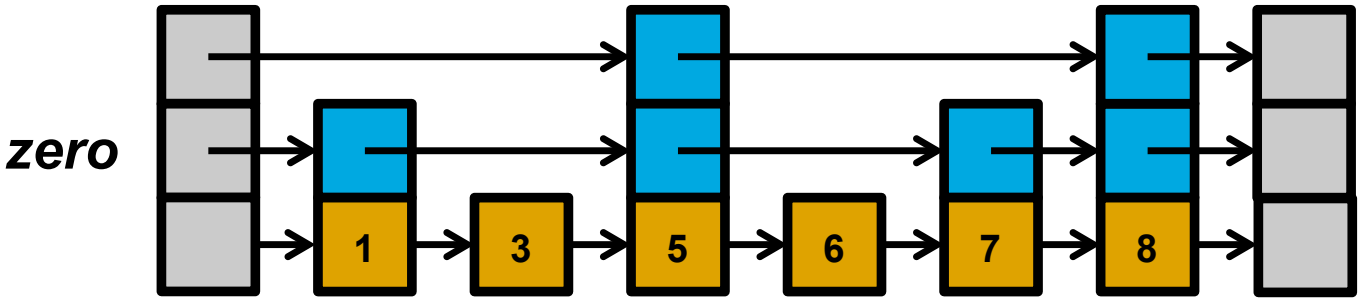



```
next = array[(zero + i) % N];
```

› Lowering of the Rotating Skip List is done by incrementing a global zero index

THE UNIVERSITY OF SYDNEY

The Rotating Skip List

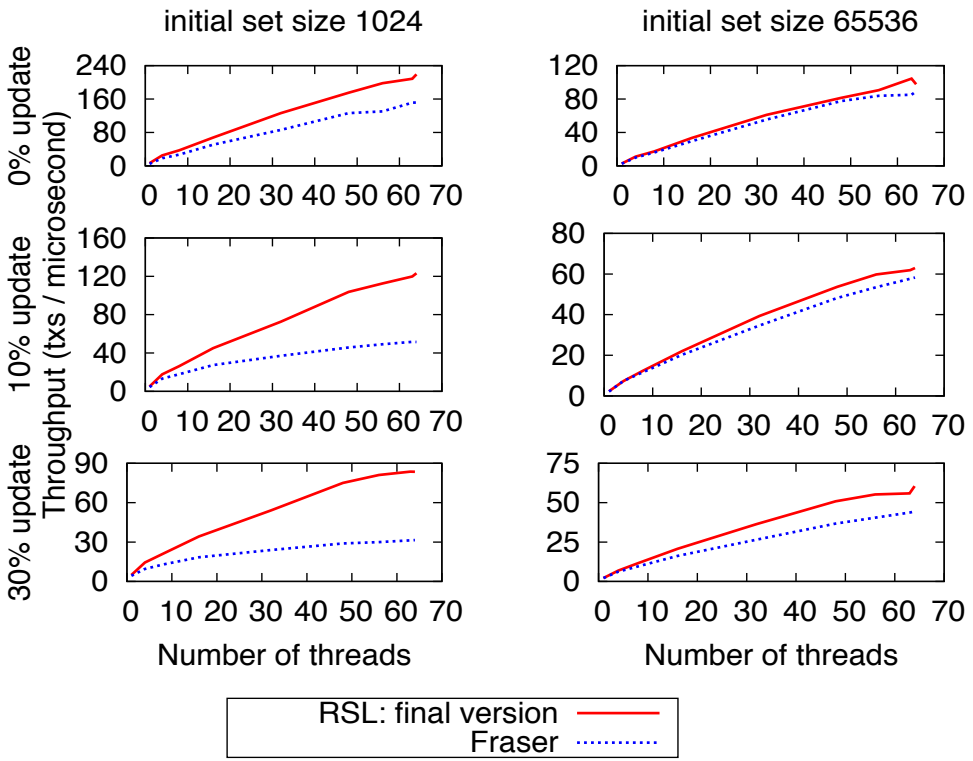



```
next = array[(zero + i) % N];
```

- Lowering of the Rotating Skip List is done by incrementing a global zero index
- Arrays must be rotating so that we don't fall off the end after several lowering

THE UNIVERSITY OF SYDNEY

The Rotating Skip List



- The Rotating Skip List outperforms the state-of-the-art alternative for workloads tested during research
- The Rotating Skip List scales better with the percentage of updates and with the number of threads

THE UNIVERSITY OF SYDNEY

Discussion

- The decoupling technique of the No Hot Spot Non-Blocking Skip List can provide a performance benefit in the C context, but:
- Other design considerations are important:
  - Array representation of index towers more effective
  - Maintenance thread does not need to be continuously running
  - Worker threads do not need to process full deletions

Suggestions for Future Work

› The Rotating Skip List could be extended in the following ways:

- A dynamic background thread could respond to changes in workload to reduce rebalancing latency
- Range queries could be added to the interface to make the Rotating Skip List more useful as a data structure for a production system such as an in-memory database

THE UNIVERSITY OF SYDNEY

THANK YOU

Questions?

References

1. Ryan Johnson, Ippokratis Pandis, Nikos Hardavellas, Anastasia Ailamaki, and Babak Falsafi. Shore-mt: a scalable storage manager for the multicore era. In *Proceedings of the 12th Interna- tional Conference on Extending Database Technology: Advances in Database Technology,* EDBT '09, pages 24–35, New York, NY, USA, 2009. ACM.
2. Tyler Crain, Vincent Gramoli, and Michael Raynal. No hot spot non-blocking skip list. *33rd International Conference on Distributed Computing Systems (ICDCS), 2013.*
3. Keir Fraser. *Practical lock-freedom.* PhD thesis, Cambridge University Computer Laboratory, 2003. Also available as Technical Report UCAM-CL-TR-579, 2004.