mongoDB

MongoDB v3.0 Deep Dive

{ Name: ‘Bryan Reinero’,
Title: ‘Developer Advocate’,
Twitter: ‘@blimpyacht’,
Email: ‘bryan@mongdb.com’ }

Agenda

- Storage Engine API
- MmapV1
- WiredTiger
- Document Level Concurrency
- Index Improvements
- The Future

Storage Engine API

- Allows to "plug-in" different storage engines
  - Different work sets require different performance characteristics
  - mmapv1 is not ideal for all workloads
  - More flexibility
    - Can mix storage engines on same replica set/sharded cluster
- Opportunity to integrate further (HDFS, native encrypted, hardware optimized …)

Storage Engine API

StorageEngine

Top Level Class for creating a Storage Engine

RecoveryUnit

Durability interface. Ensures data is persisted. On-disk information mutated through this interface

DatabaseCatalogEntry

MongoDB Logical Database

CollectionCatalogEntry

MongoDB Collection

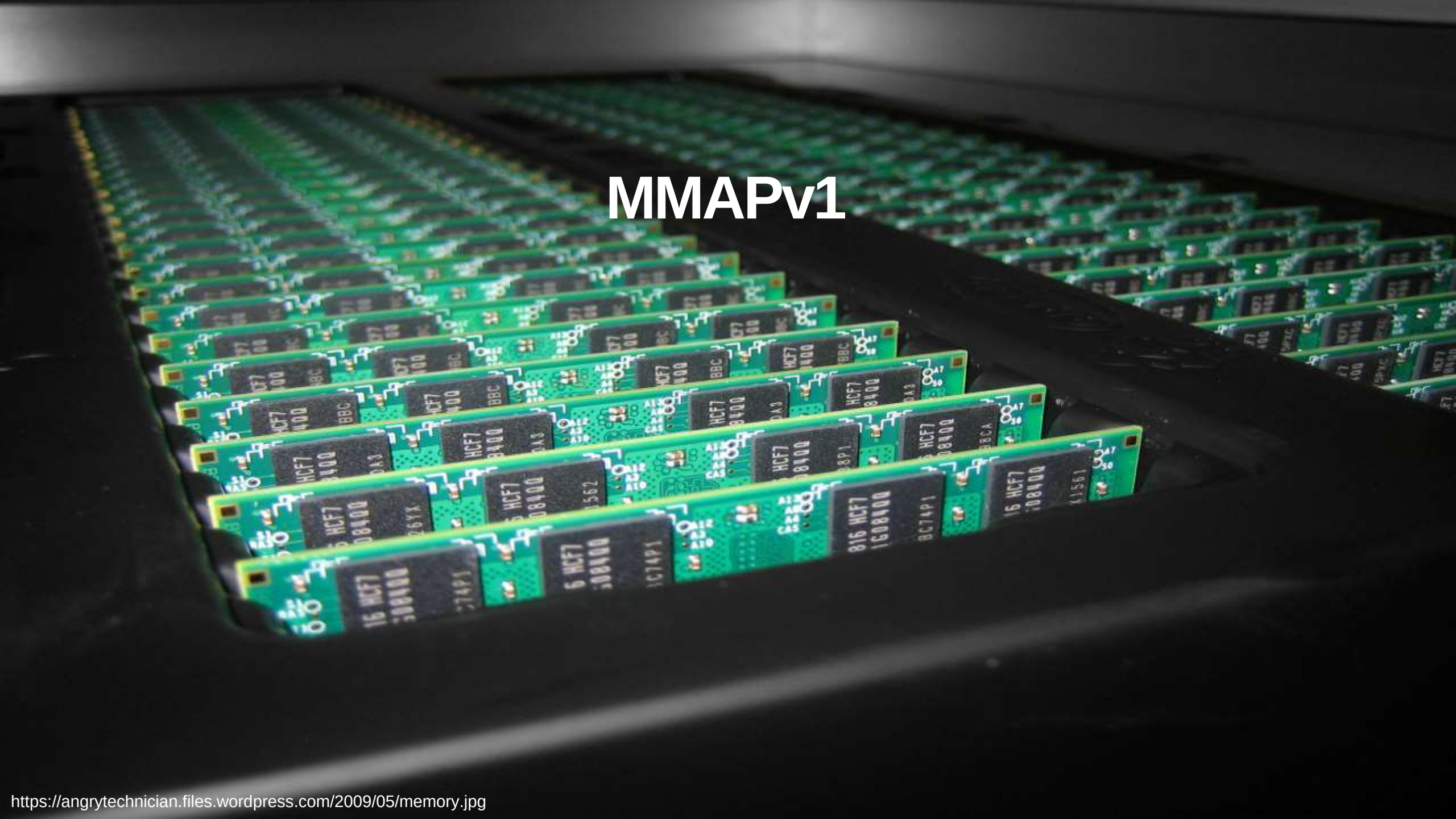
SortedDataInterface

Index implementation. Not all Indexes are B-trees

MongoDB Storage Engines

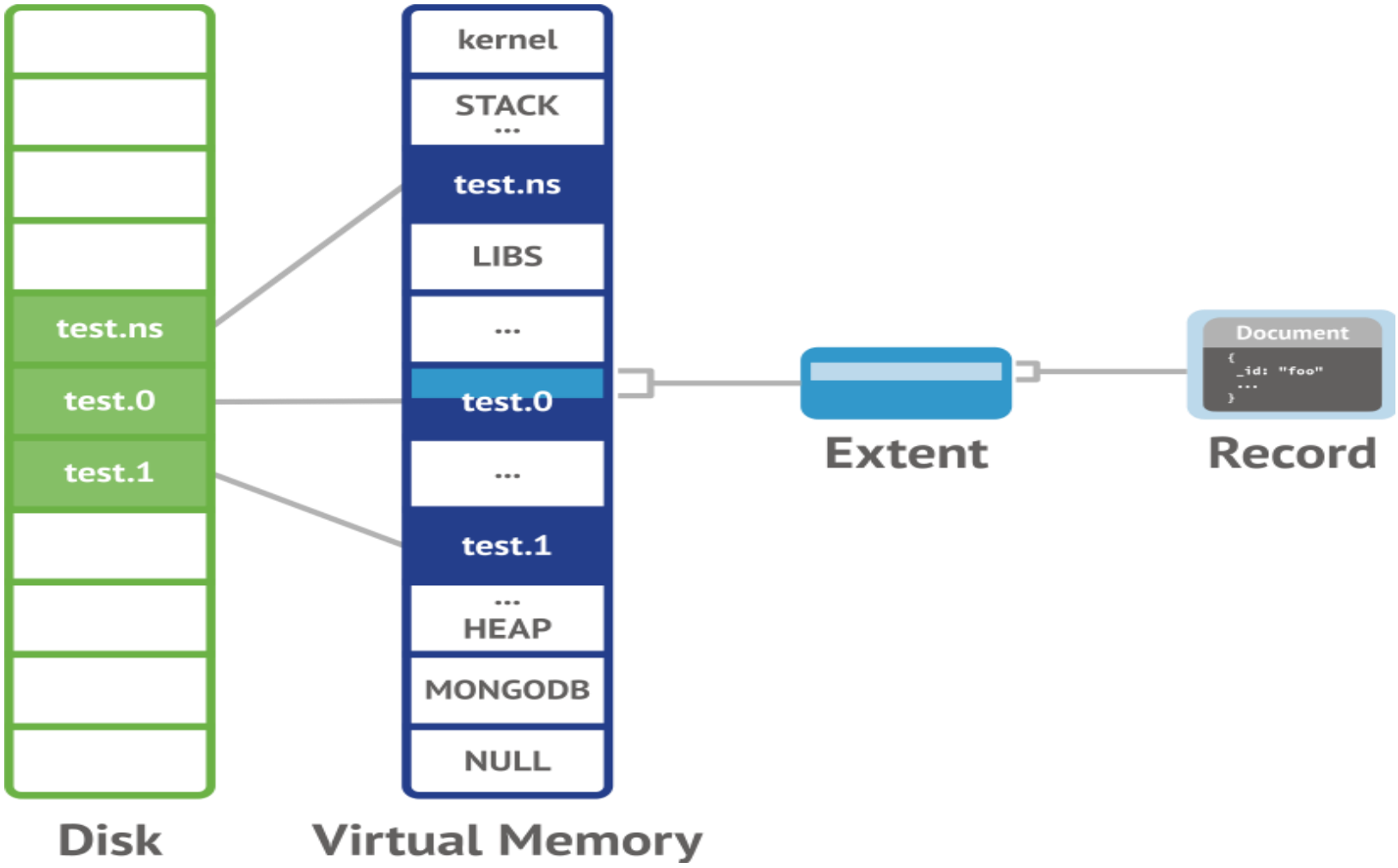
- <= MongoDB 2.6
  - One unique mechanism using Memory Mapped Files
  - "mmapv1" Storage Engine
- MongoDB 3.0 has a few more options
  - mmapv1 – default
  - wiredTiger
  - (in_memory – experimental only)

MMAPv1



https://angrytechnician.files.wordpress.com/2009/05/memory.jpg

MMAPv1


test.ns
test.0
test.1
Disk
kernel
STACK
...
test.ns
LIBS
...
test.0
...
test.1
...
HEAP
MONGODB
NULL
Virtual Memory
Extent
Document
{
_id: "foo"
...
}
Record

WIREDTIGER

Making Big Data Roar

What is WiredTiger?

- Storage engine company founded by BerkeleyDB alums
- Recently acquired by MongoDB
- Available as a storage engine option in MongoDB 3.0

Why is WiredTiger Awesome

- Document-level concurrency
- Disk Compression
- Consistency without journaling
- Better performance on many workloads
  - write heavy

Improving Concurrency

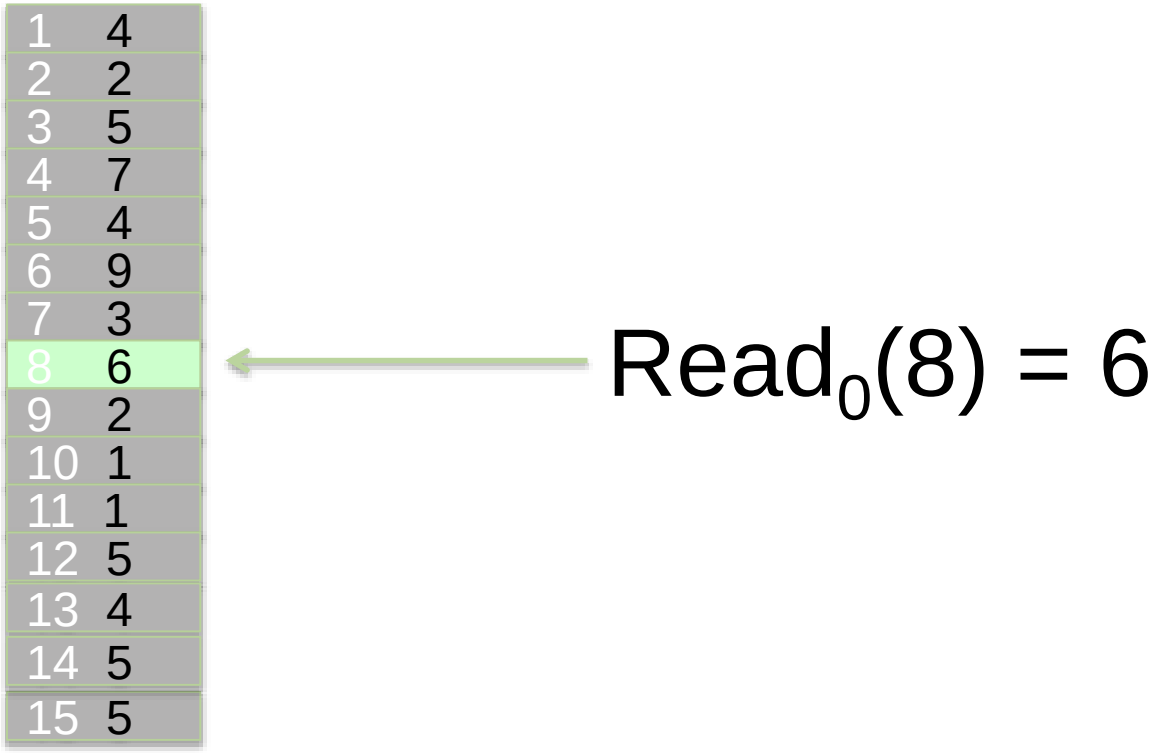
- 2.2 – Global Lock
- 2.4 – Database-level Locking
- 3.0 MMAPv1 – Collection-level Locking
- 3.0 WT – Document-level
  - Writes no longer block all other writes
  - Higher level of concurrency leads to more CPU usage

Lock Free Algorithms

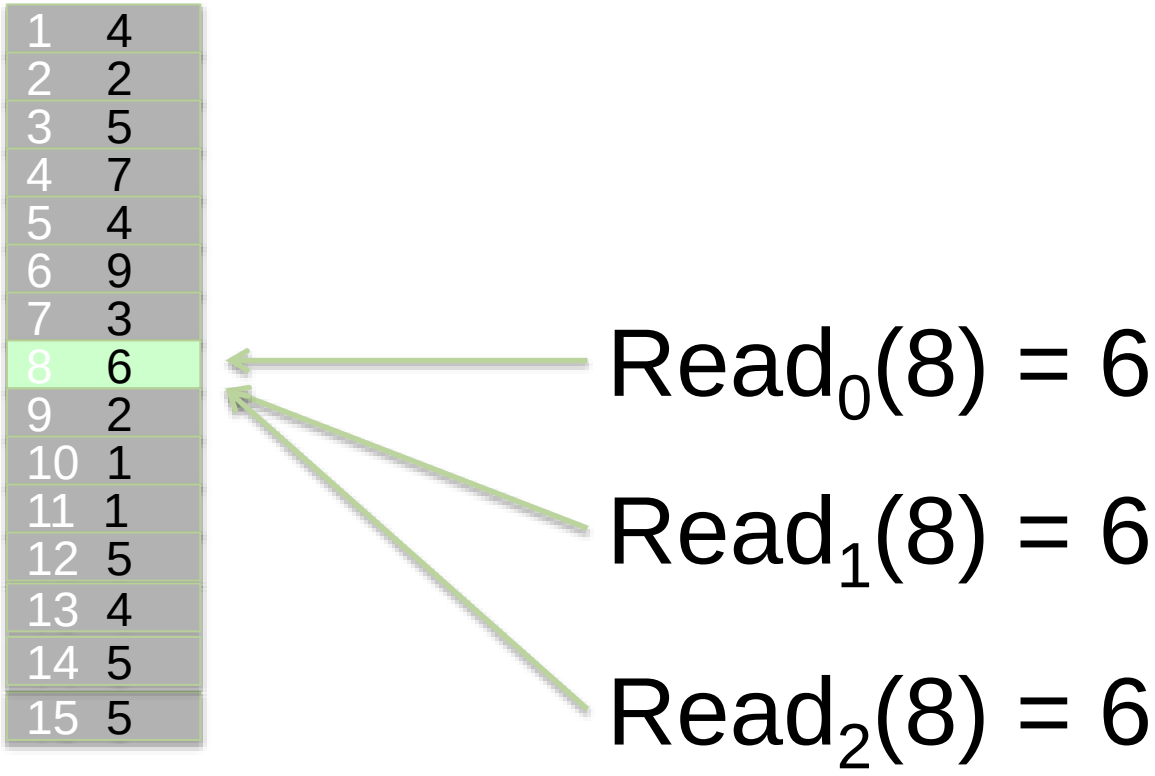
1	4
2	2
3	5
4	7
5	4
6	9
7	3
8	6
9	2
10	1
11	1
12	5
13	4
14	5
15	5

http://ses.library.usyd.edu.au/bitstream/2123/5353/1/michael-cahill-2009-thesis.pdf

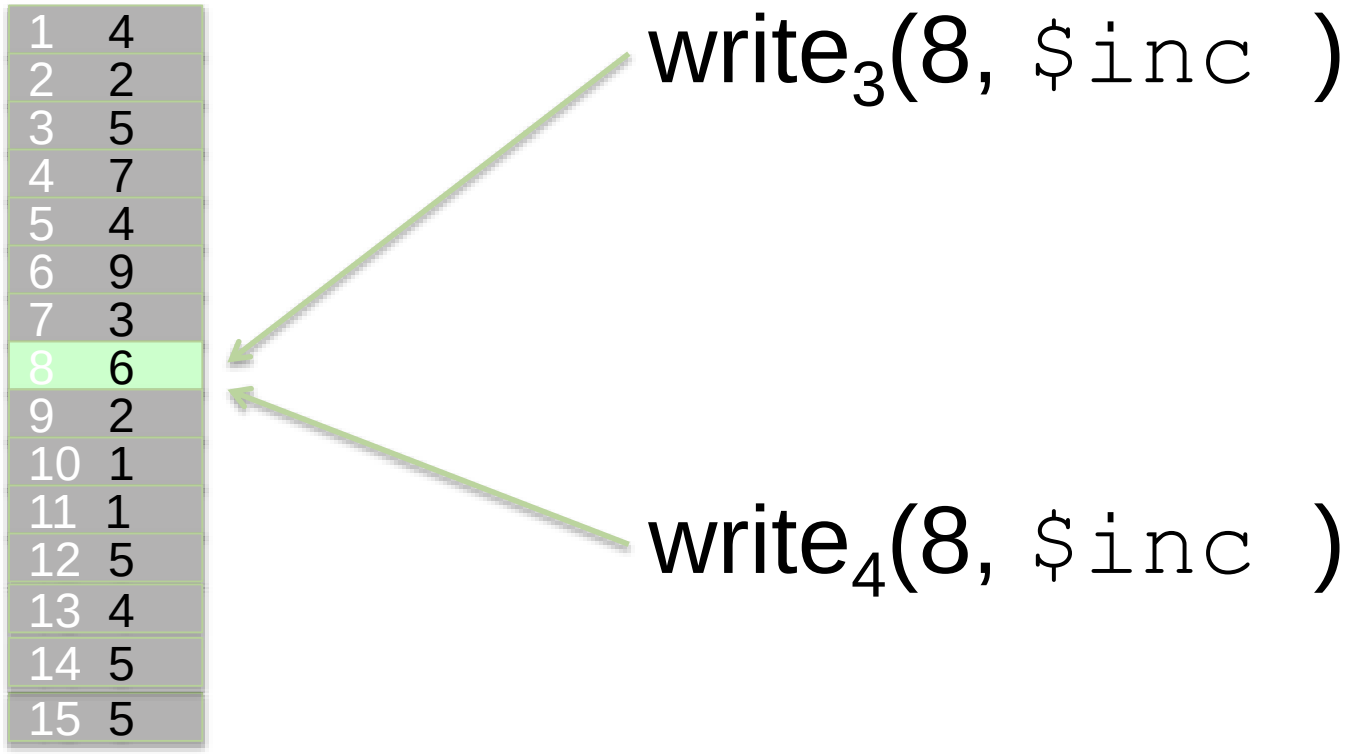
Lock Free Algorithms



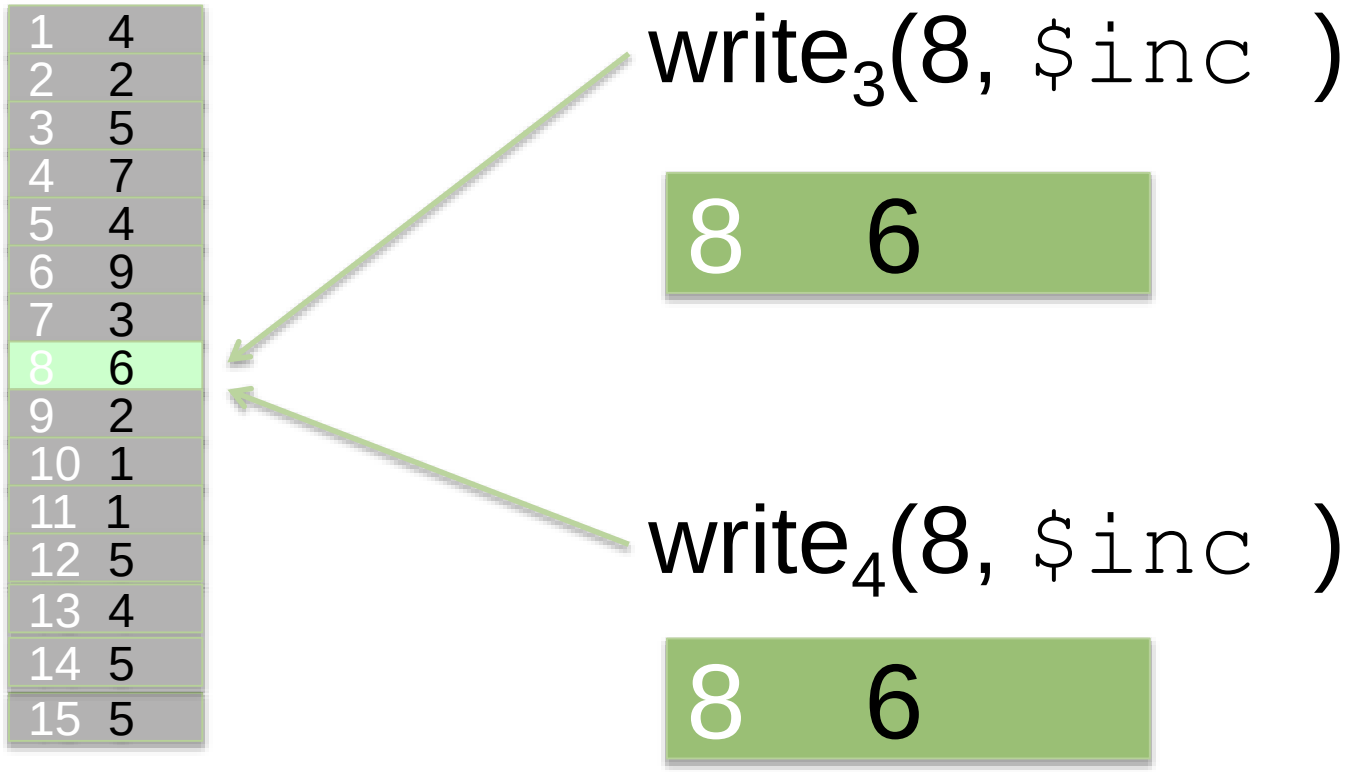
Lock Free Algorithms



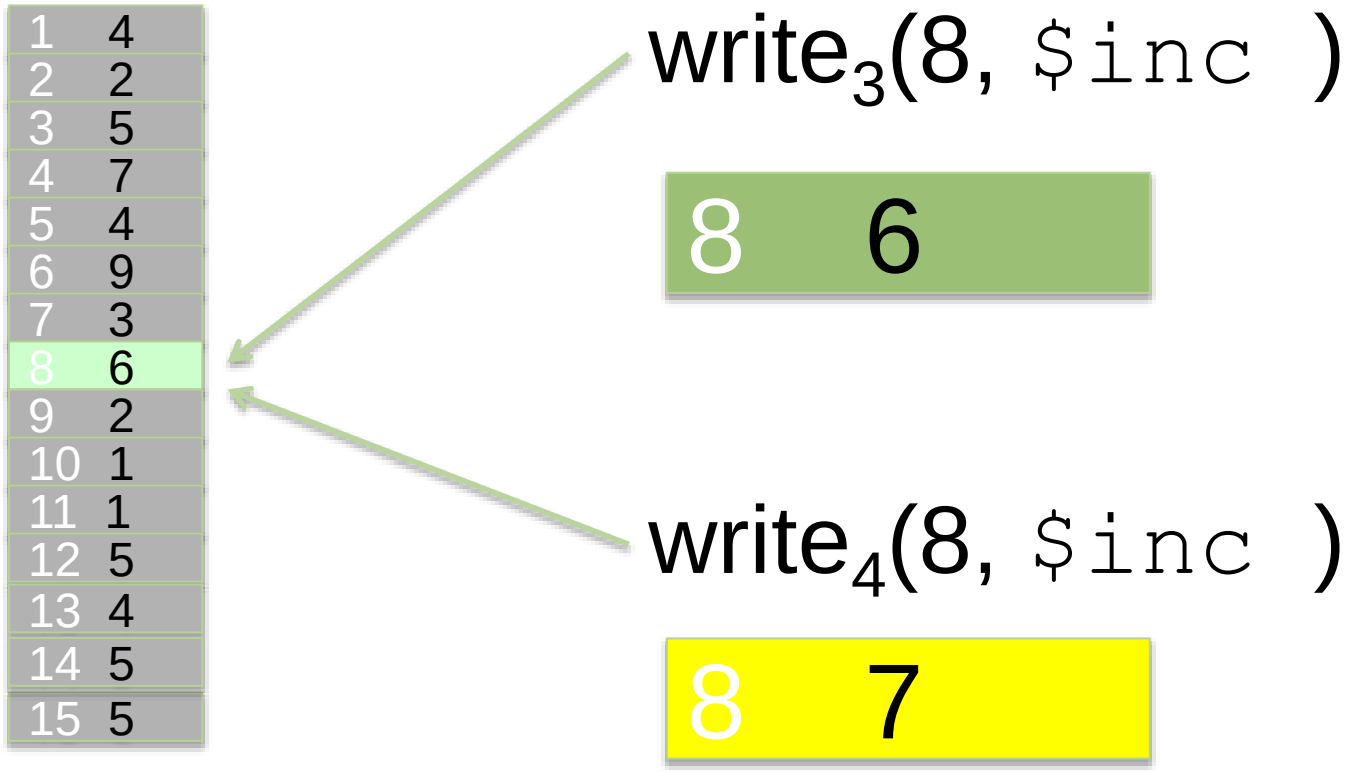
Lock Free Algorithms



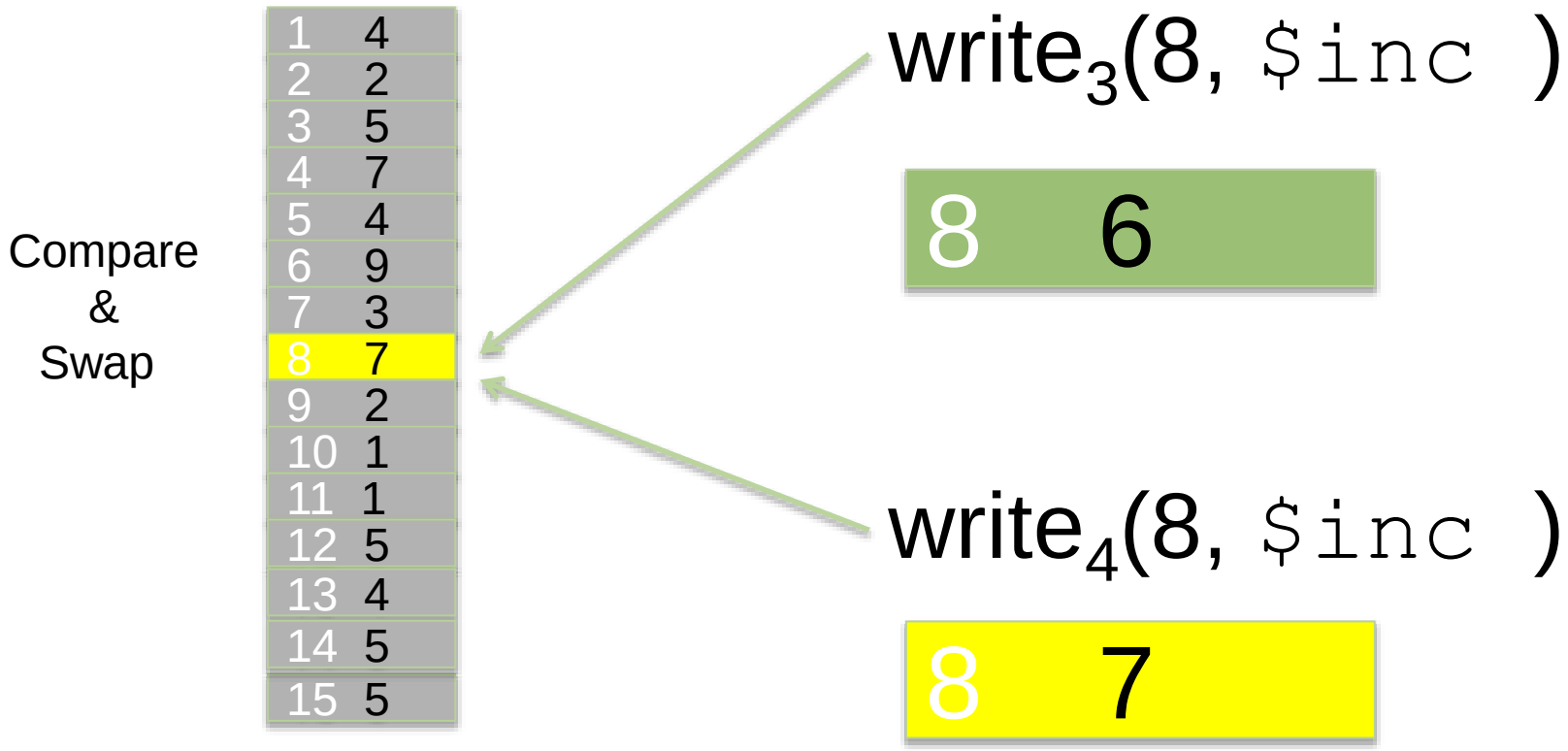
Lock Free Algorithms



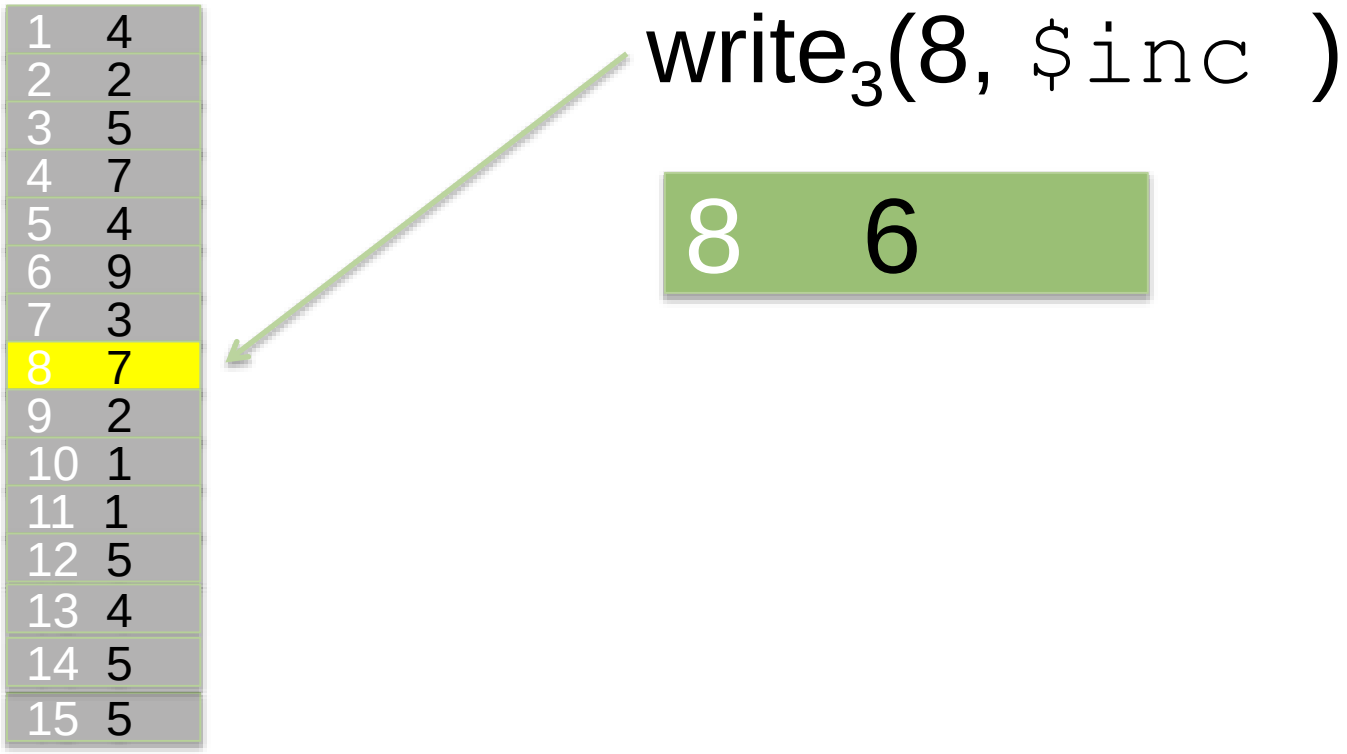
Lock Free Algorithms



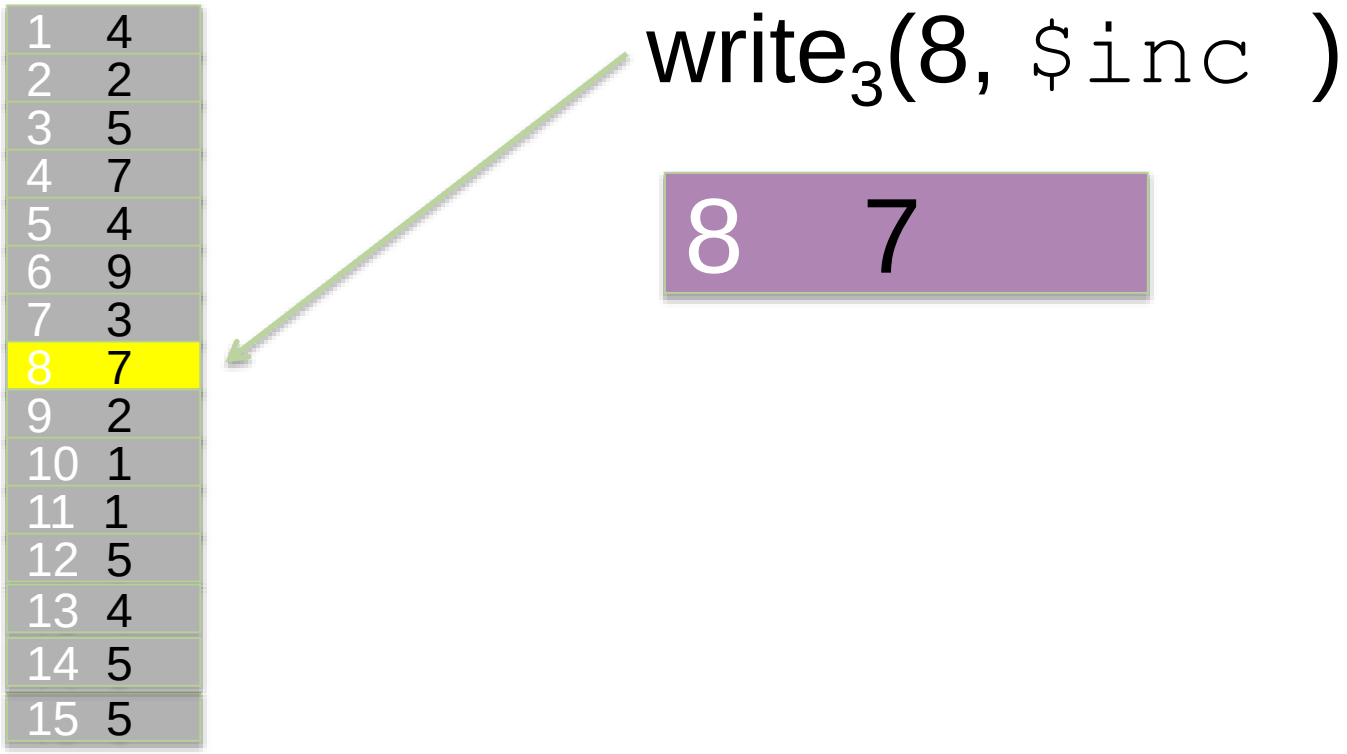
Lock Free Algorithms



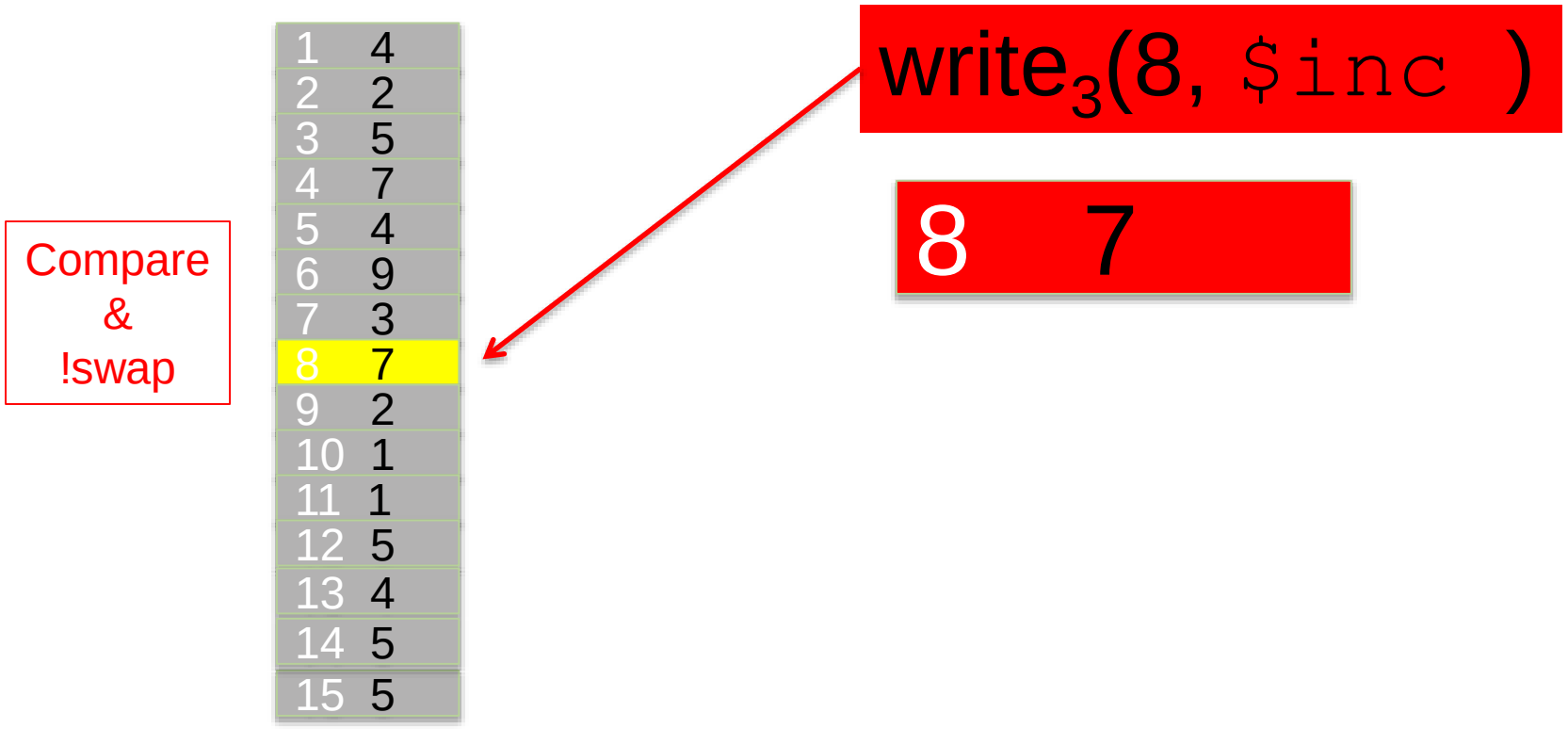
Lock Free Algorithms



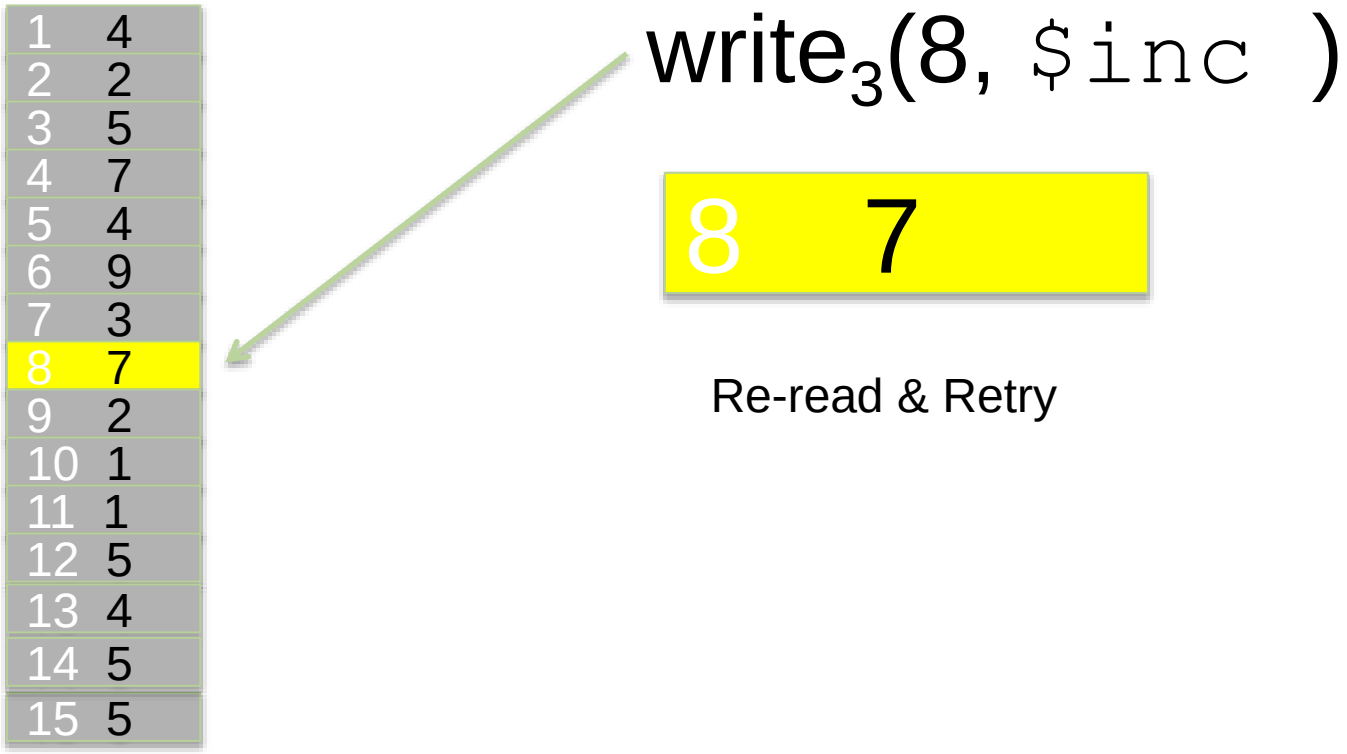
Lock Free Algorithms



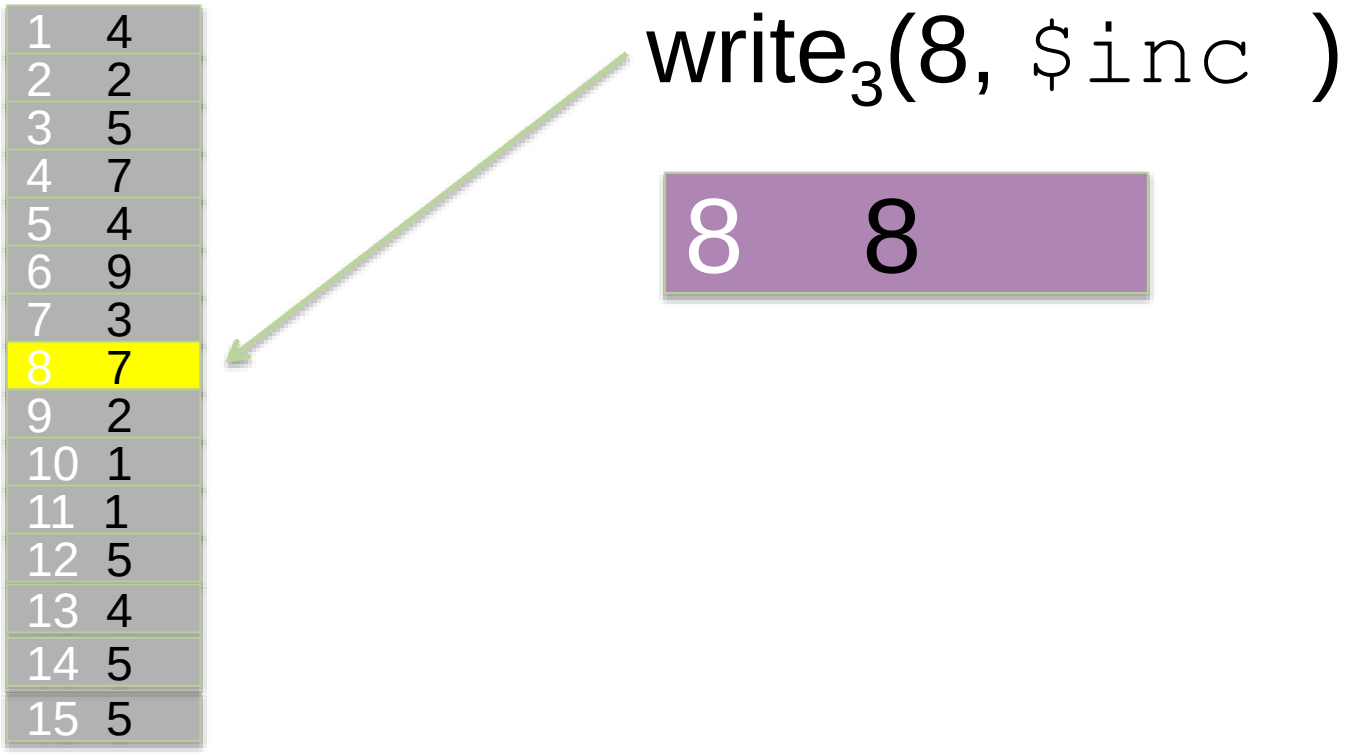
Lock Free Algorithms



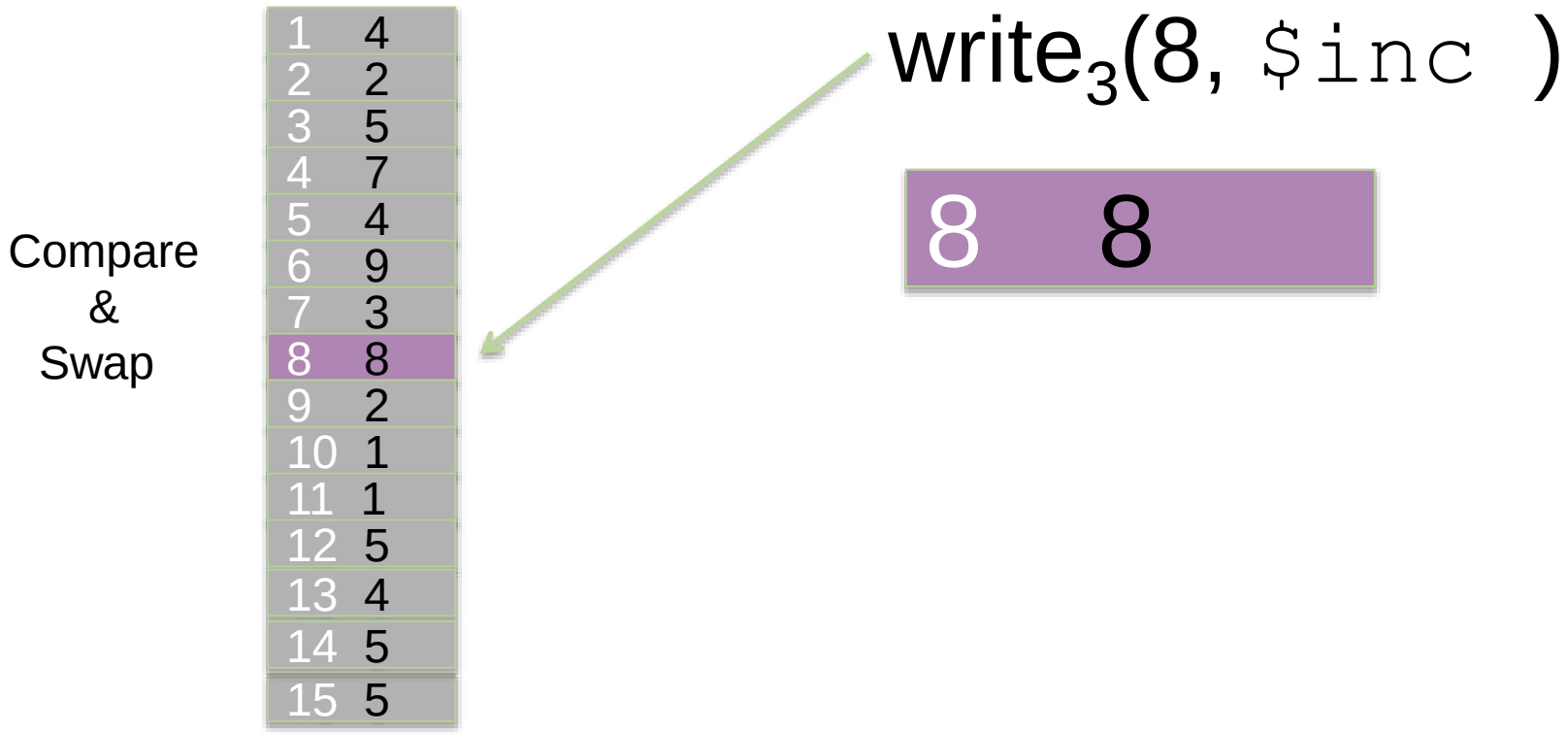
Lock Free Algorithms



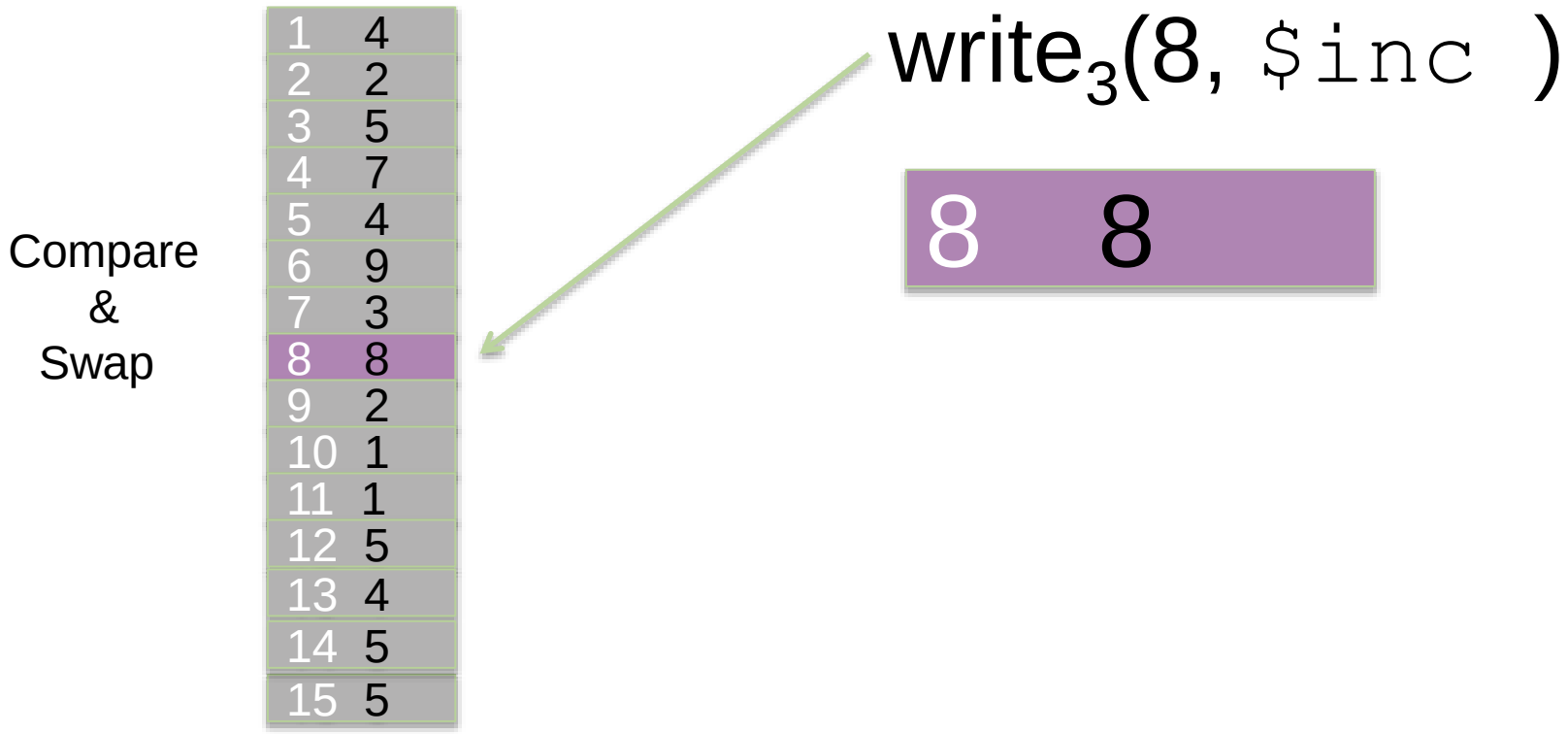
Lock Free Algorithms



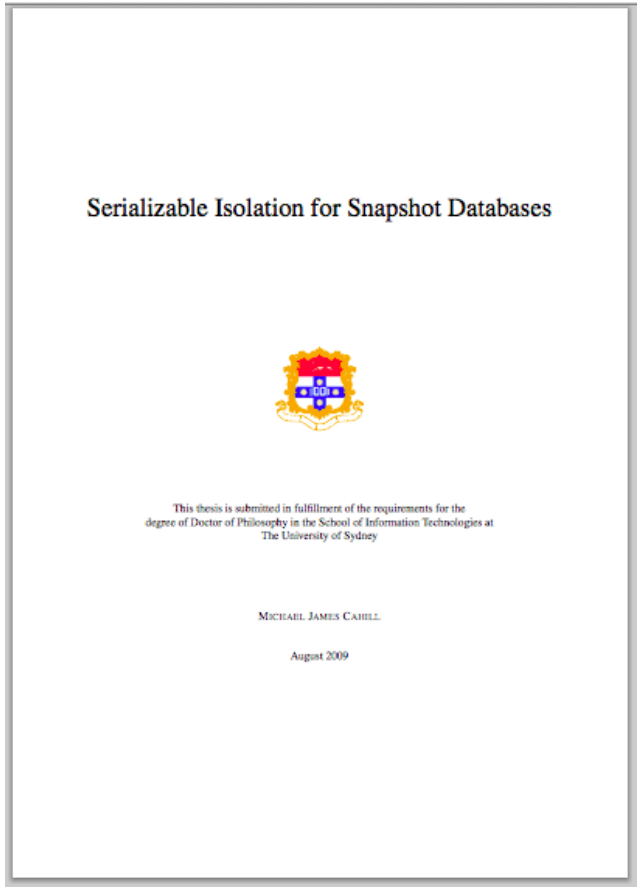
Document Level Concurrency Control



Document Level Concurrency Control



Read More



Serializable Isolation for Snapshot Databases

This thesis is submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in the School of Information Technologies at The University of Sydney

Michael James Cahill

August 2009

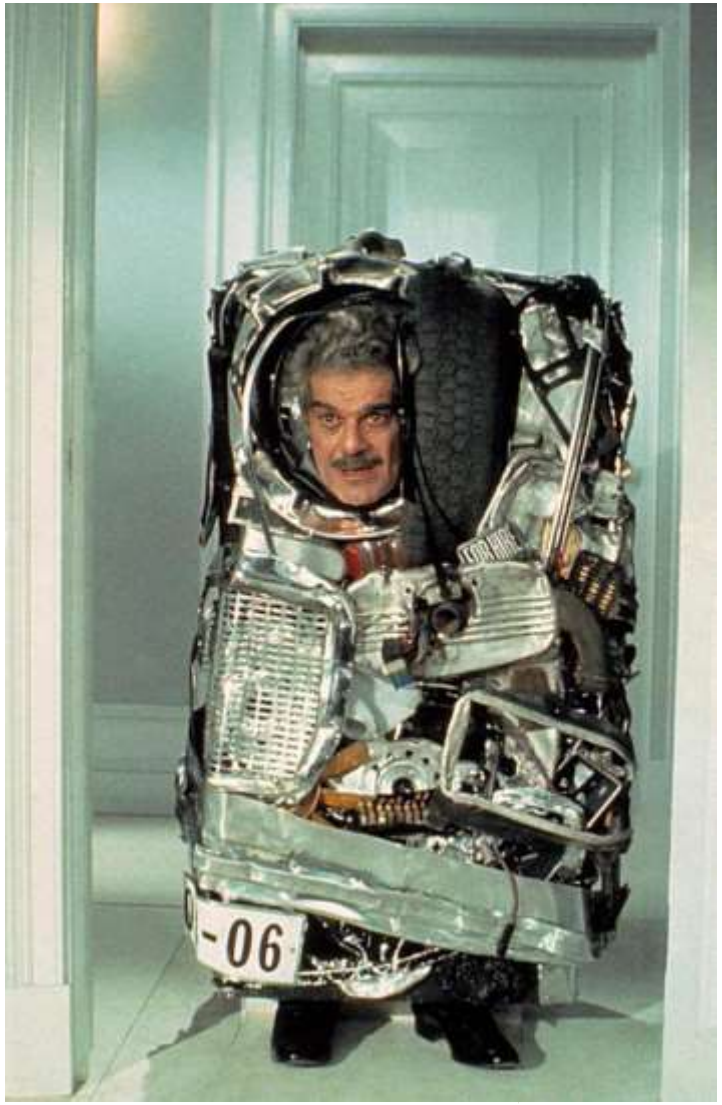
http://ses.library.usyd.edu.au/bitstream/2123/5353/1/michael-cahill-2009-thesis.pdf

Wired Tiger Concurrency

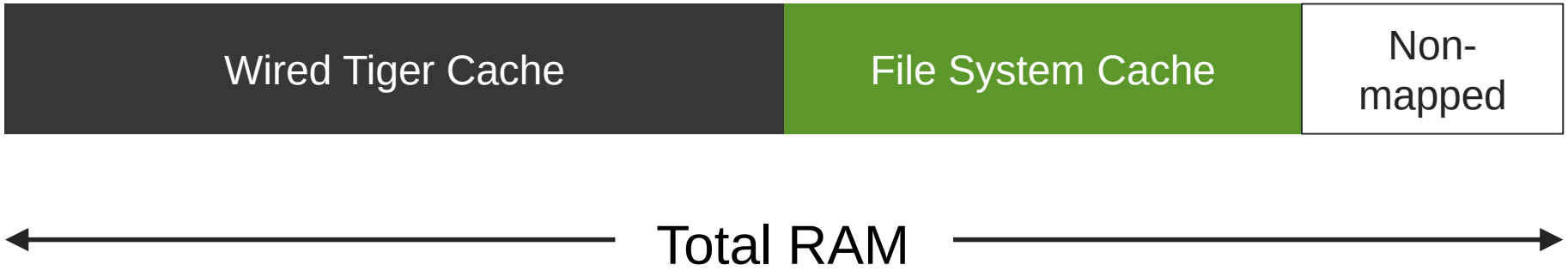
- Fine grained
- Lock free
- Wait free
- Stone cold
- Superfly

Compression

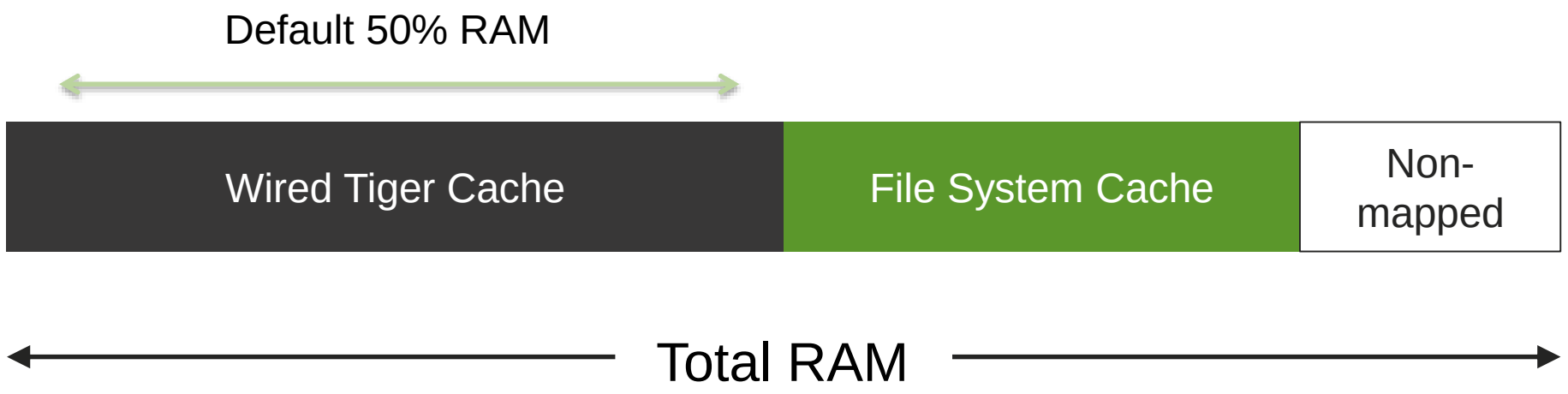
- Data is compressed on disk
- 2 supported algorithms
  - snappy: default. Good compression, relatively low overhead
  - zlib: Better
- Indexes are compressed using prefix compression
  - Allows compression in memory



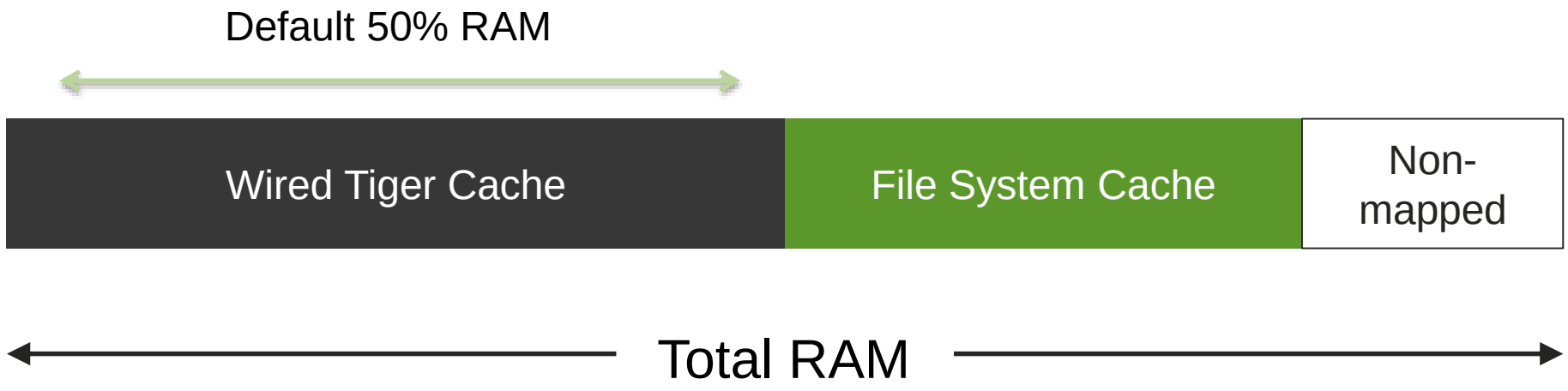
Tuning Wired Tiger



Tuning Wired Tiger



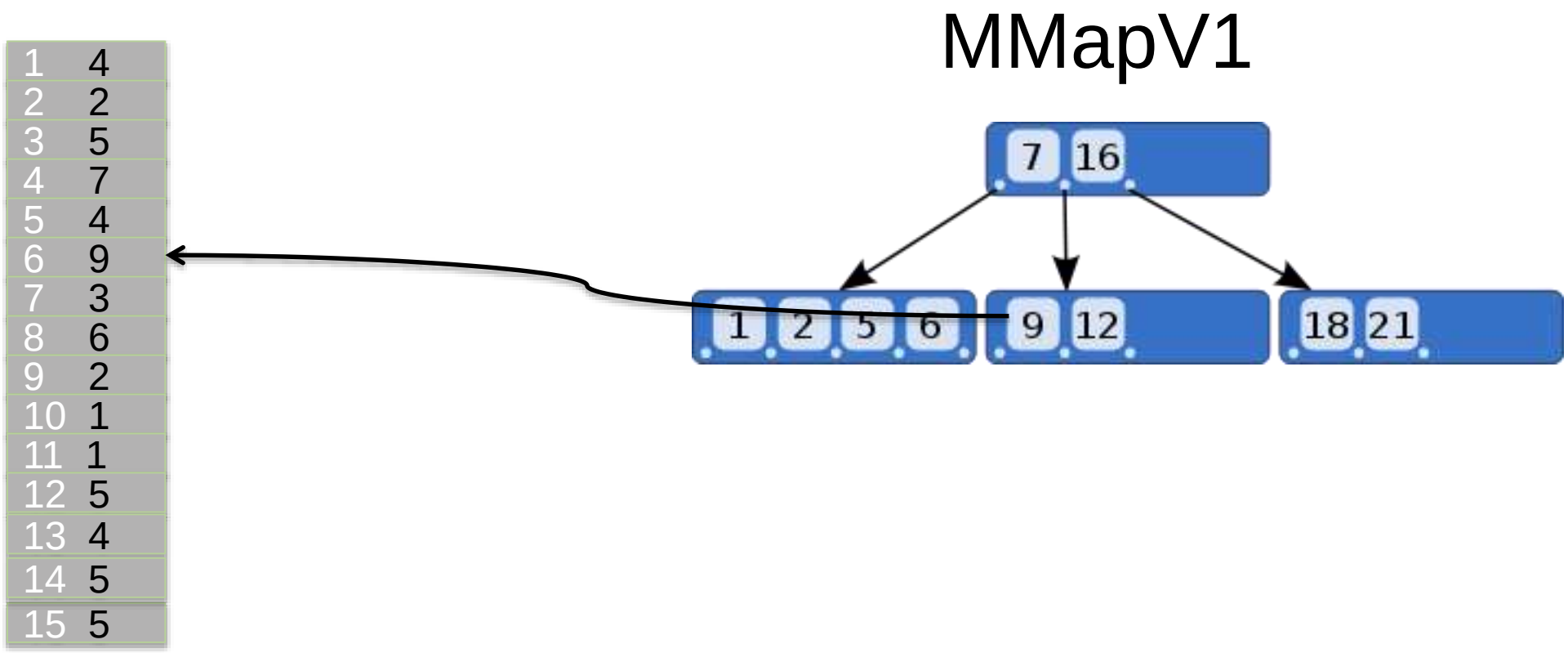
Tuning Wired Tiger



Knobs

- Wired Tiger Cache Size
- Compression
  - Snappy
  - Zlib
  - off

Indexing Improvements



Indexing Improvements

1	4
2	2
3	5
4	7
5	4
6	9
7	3
8	6
9	2
10	1
11	1
12	5
13	4
14	5
15	5

```
{
  _id: 6,
  categories: [
    “database”,
    “distributed”,
    “document store”
  ]
}
```

Indexing Improvements

1	4
2	2
3	5
4	7
5	4
6	9
7	3
8	6
9	2
10	1
11	1
12	5
13	4
14	5
15	5

```
{
  _id: 6,
  categories: [
    "database",
    "distributed",
    "document store",
    "sharded"
  ]
}
```

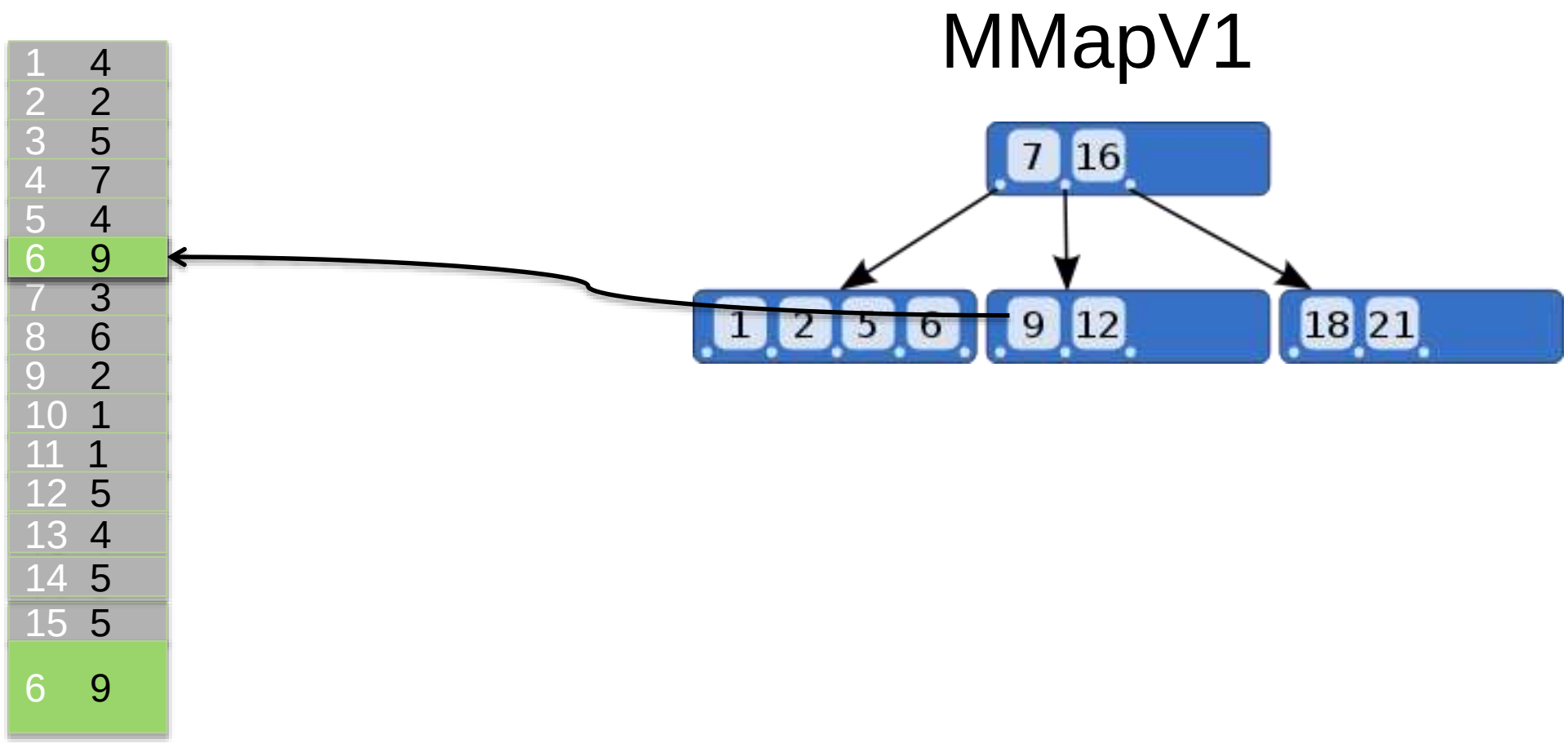
```
{$push:
  {
    categories:
    "sharded"
  }
}
```

Indexing Improvements

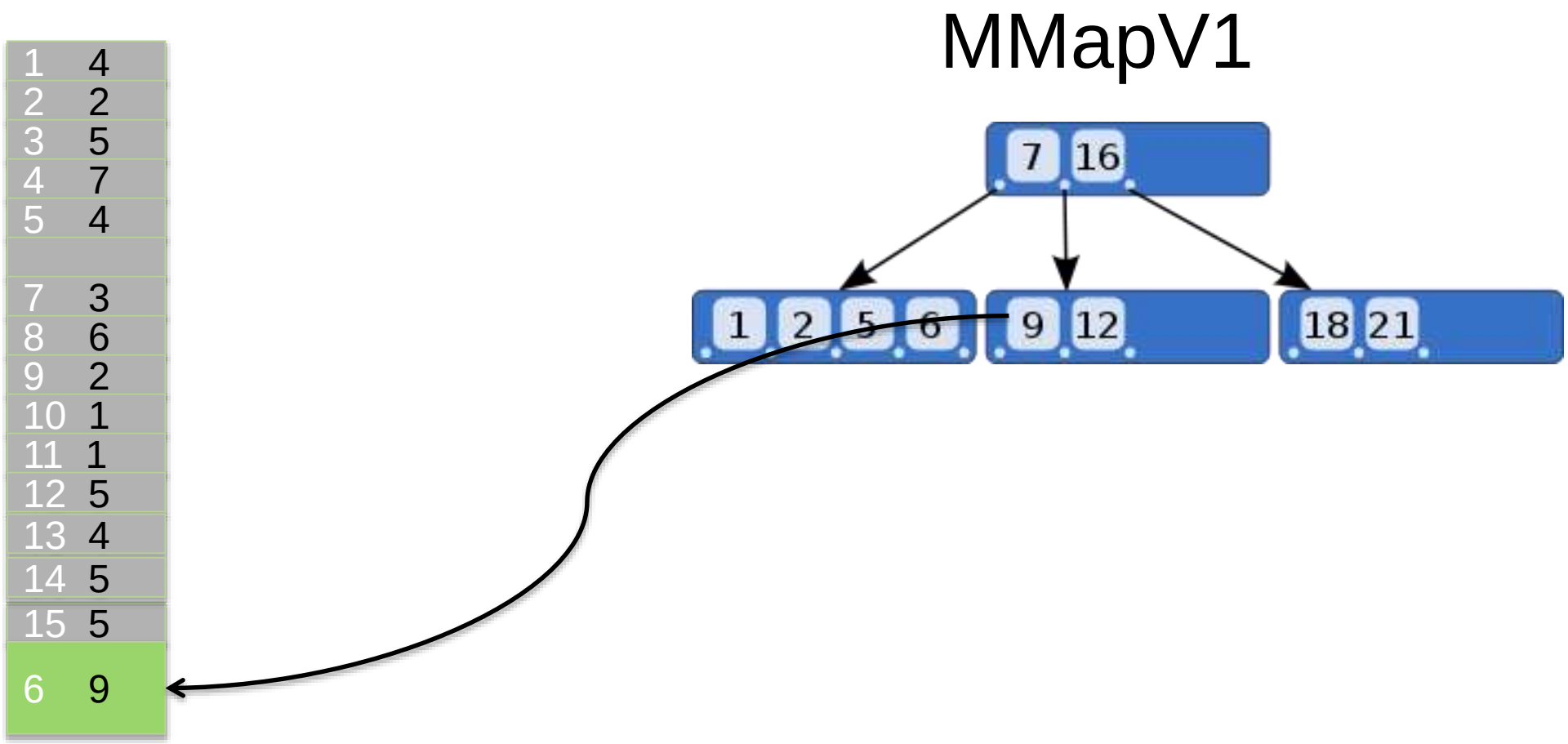
1	4
2	2
3	5
4	7
5	4
6	9
7	3
8	6
9	2
10	1
11	1
12	5
13	4
14	5
15	5

```
{
  _id: 6,
  categories: [
    “database”,
    “distributed”,
    “document store”,
    “sharded”
  ]
}
```

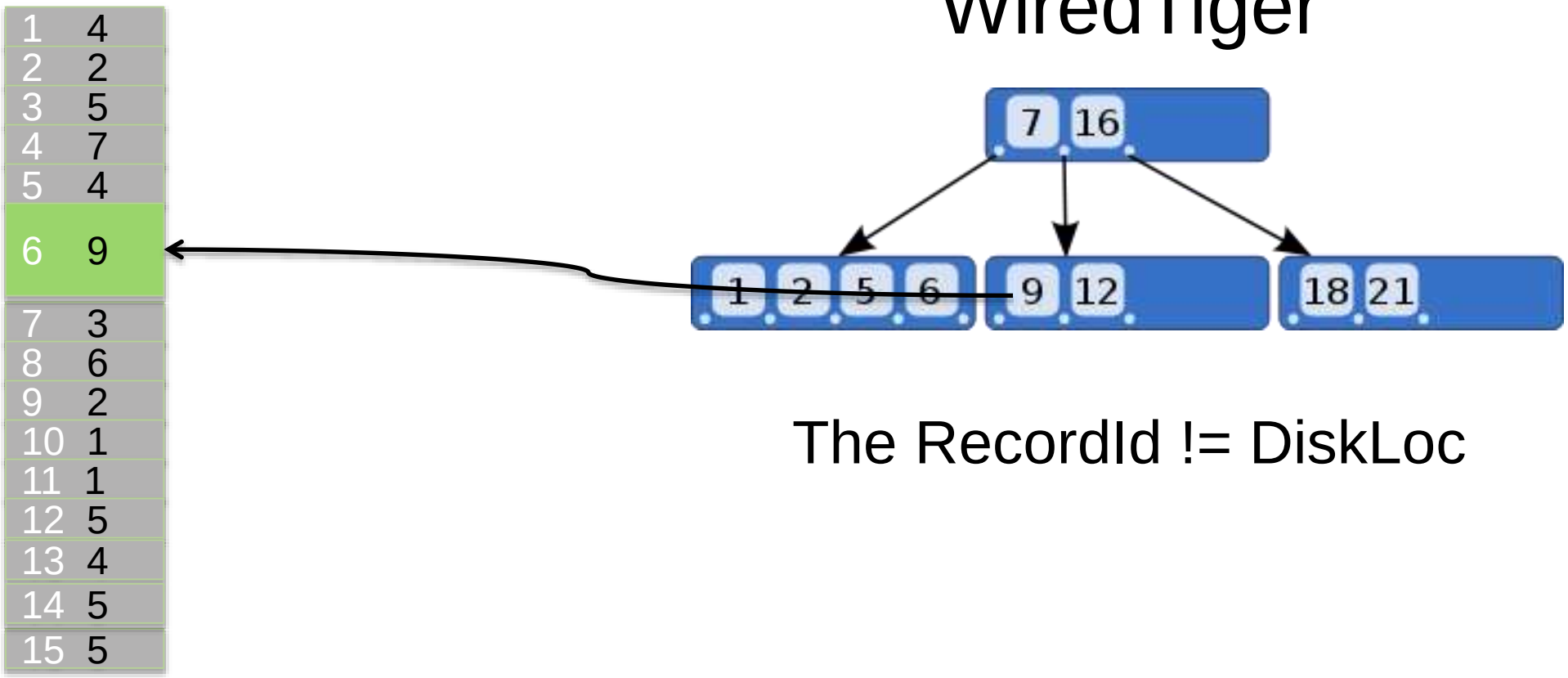
Indexing Improvements



Indexing Improvements



Indexing Improvements



The RecordId != DiskLoc

Consistency without Journaling

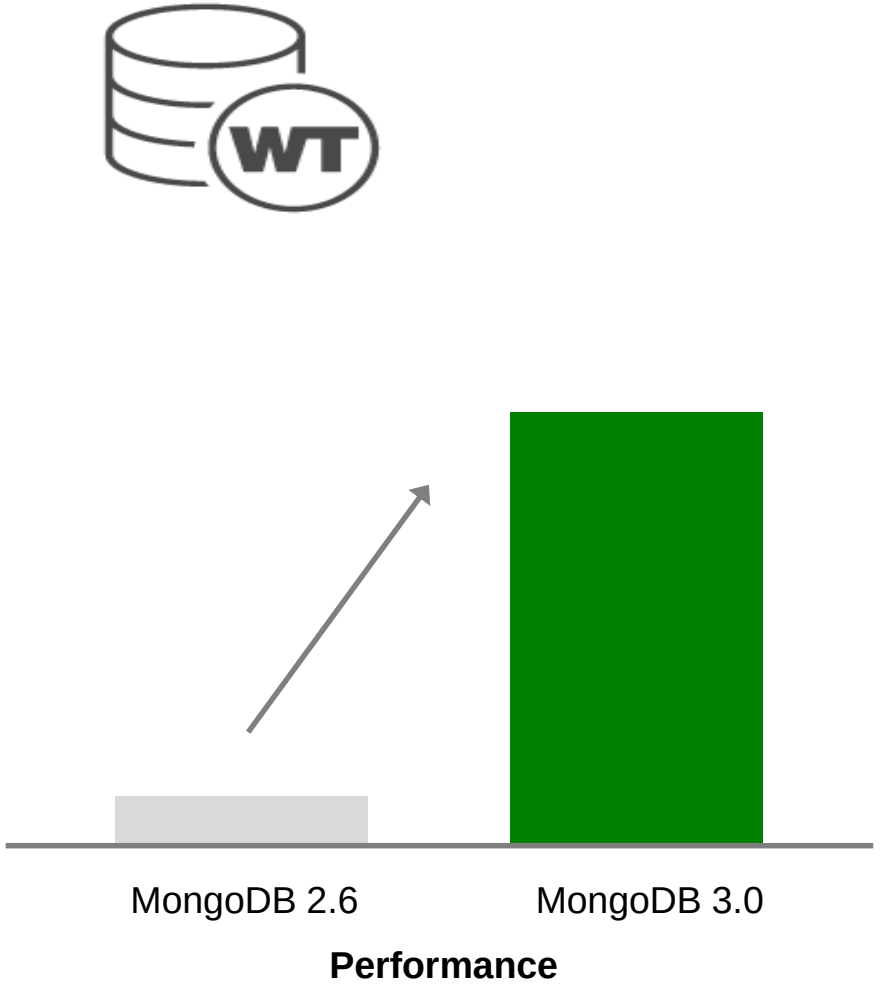
- MMAPv1 uses write-ahead log (journal) to guarantee consistency
- WT doesn't have this need: no in-place updates
  - Write-ahead log committed at checkpoints
    - 2GB or 60sec by default – configurable!
  - No journal commit interval: writes are written to journal as they come in
  - Better for insert-heavy workloads
- Replication guarantees the durability

7x-10x Performance, 50%-80% Less Storage

How: WiredTiger Storage Engine

- Same data model, same query language, same ops
- Write performance gains driven by document-level concurrency control
- Storage savings driven by native compression
- 100% backwards compatible
- Non-disruptive upgrade

https://www.mongodb.com/blog/post/high-performance-benchmarking-mongodb-and-nosql-systems



Playing nice together

- Can not
  - Can't copy database files
  - Can't just restart w/ same dbpath
- Yes we can!
  - Initial sync from replica set works perfectly!
  - mongodump/restore
- Rolling upgrade of replica set to WT:
  - Shutdown secondary
  - Delete dbpath
  - Relaunch w/ --storageEngine=wiredTiger
  - Rollover


VERSION 3.2
AND BEYOND THE INFINITE

Storage Engine

Storage Engines

- WiredTiger (now default)
- In-Memory
- Encryption at Rest

Tools

- Schema visualizer

Features

- $lookup (Enterprise)
- Read Committed
- Schema Validation Rules
- Partial Indexes

Storage Engine

Storage Engines

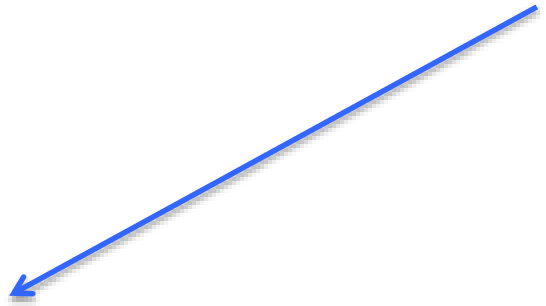
- WiredTiger (now default)
- In-Memory
- Encryption at Rest

Tools

- Schema visualizer

Features

- $lookup (Enterprise)
- Read Committed
- Schema Validation Rules
- Partial Indexes



Features now available in v3.1.6 Community release

Thanks!

{ name: ‘Bryan Reinero’,
title: ‘Developer Advocate’,
twitter: ‘@blimpyacht’,
code: ‘github.com/breinero’
email: ‘bryan@mongdb.com’ }