What next?
5 Things to think about

Alvin Richards

Technical Director, EMEA
alvin@10gen.com
@jonnyeight

10gen mongoDB

Everything is Done


STOP

Are you really done?

- ✓ A/B done, UX design chosen
- ✓ Code complete
- ✓ QA complete
- ✓ Performance & Stress test complete
- ✓ Software deployed
- ★ Go, Go, GO!!!

You are starting here...

http://community.qlikview.com/cfs-filesystemfile.ashx/__key/CommunityServer.Blogs.Components.WeblogFiles/theqlikviewblog/Cutting-Grass-with-Scissors-_2D00_-2.jpg

And want to get here...

http://www.bitquill.net/blog/wp-content/uploads/2008/07/pack_of_harvesters.jpg

Avoiding this...

http://media.egotvonline.com/wp-content/uploads/2011/07/train-wreck-1.jpg

Five things to think about

1. Schema Design & Shard Key
2. Machine Sizing: Disk and Memory
3. Load Testing and Monitoring
4. Backup and restore
5. Ops Play Book

Schema

- Single biggest performance factor
- More choices than in an RDBMS
- Embedding, index design, shard keys

Activity Stream – Embedded

```
// users - one doc per user with all tweets
{  _id:   "alvin",
   email: "alvin@10gen.com",
   tweets: [
      {
          user:  "bob",
          tweet: "20111209-1231",
          text:  "Best Tweet Ever!"
      }
  ]
}
```

Activity Stream – Linking

```
// users - one doc per user
  {  _id:   "alvin",
     email: "alvin@10gen.com"
  }

// tweets - one doc per user per tweet
  {
      user:  "bob",
      tweet: "20111209-1231",
      text:  "Best Tweet Ever!"
  }
```

Embedding

- Great for read performance
- One seek to load entire object
- One roundtrip to database
- Writes can be slow if adding to objects all the time
- Should you embed tweets?

Activity Stream – Buckets

```
// tweets : one doc per user per day

   {
      _id: "alvin-20111209",
      email: "alvin@10gen.com",
      tweets: [
        { user:  "Bob",
          tweet: "20111209-1231",
          text:  "Best Tweet Ever!" } ,
        { author: "Joe",
          date:   "May 27 2011",
          text:   "Stuck in traffic (again)" }
      ]
    }
```

Adding a Tweet

```
tweet = { user:  "Bob",
          tweet: "20111209-1231",
          text:  "Best Tweet Ever!" }

db.tweets.update( { _id : "alvin-20111209" },
                  { $push : { tweets : tweet } );
```

Getting All Comments

```
cursor = db.tweets.find
           ( { _id : /^alvin/ } ).sort( { _id : -1 } )

while ( cursor.hasNext() ) {
      doc = cursor.next();
      for ( var i=0; i<doc.tweets.length; i++ )
            printjson( doc.tweets[i] )
}
```

Last 5 comments from last bucket

```
db.tweets.find( { _id: "alvin-20111209" },
                { tweets: { $slice : 5 } }
              ).sort( { _id: -1 } ).limit(1)
```

Deleting a Tweet

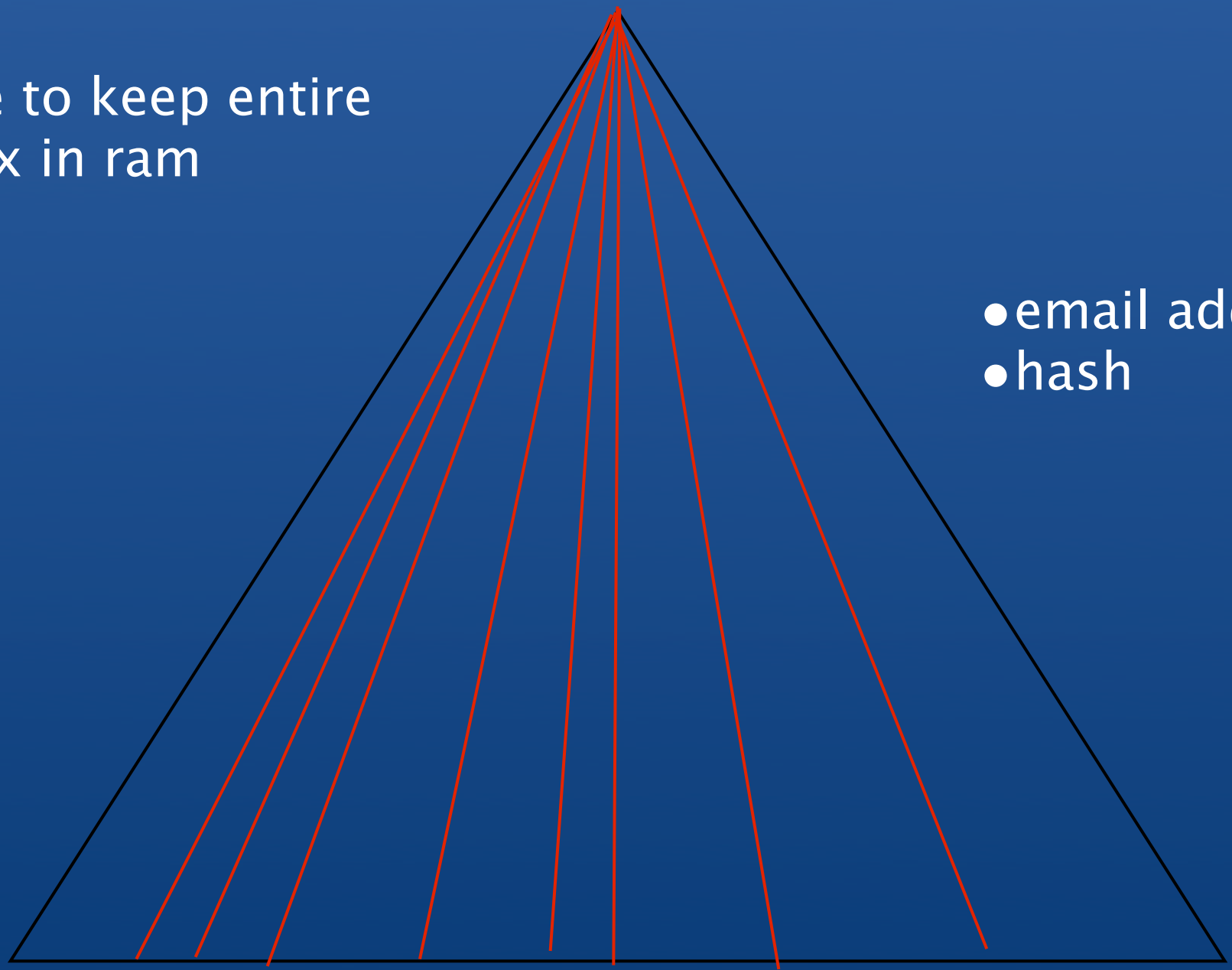
```
db.tweets.update(
   { _id: "alvin-20111209" },
   { $pull: { tweets: { tweet: "20111209-1231"  } } }
)
```

Indexes

- Index common queries
- Make sure there aren't duplicates:
  - (A) and (A,B) aren't needed
- Right-balanced indexes keep working set small

Random Index Access

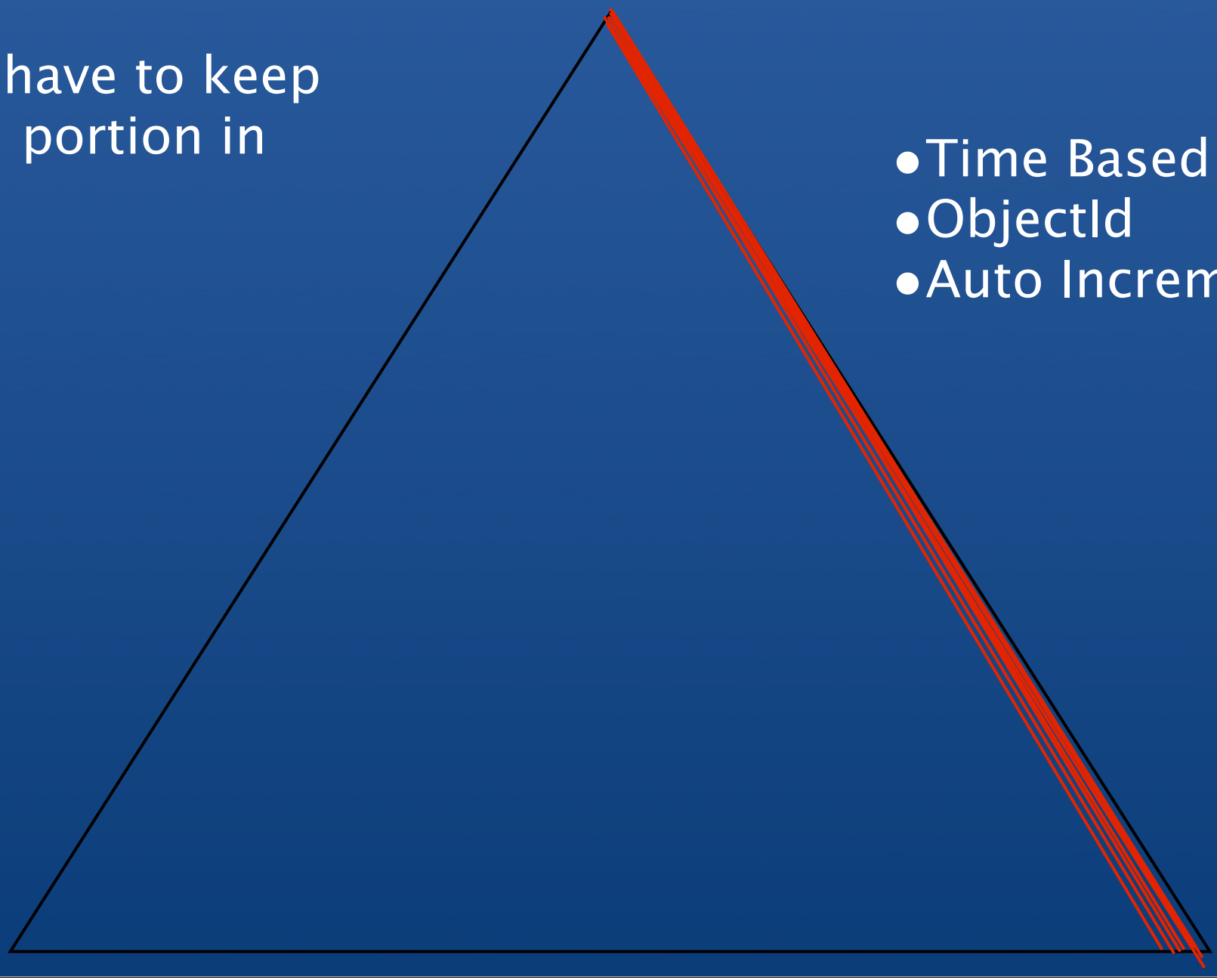
Have to keep entire index in ram



- email address
- hash

Right-Balanced Index Access

Only have to keep small portion in ram



- Time Based
- ObjectId
- Auto Increment

Covered Indexes

- Keep data sequential in index

```
find( { email: /^alvin/ },
      { first: 1, last: 1, state: 1, _id: -1 } )


index:
  { email: 1, first: 1, last: 1, state : 1 }
```

Choosing a Shard Key

- Sharding is used to scale writes and reads
- Shard key determines how data is partitioned
- Critical scalability decision

Range Based

collection	minkey	maxkey	location
users	{ name : 'Miller' }	{ name : 'Nessman' }	$shard_2$
users	{ name : 'Nessman' }	{ name : 'Ogden' }	$shard_4$
...			

collection is broken into chunks by range
chunks default to 64mb or 100,000 objects

Use Case: User Profiles

```
{ email : "alvin@10gen.com" ,
  addresses : [ { state : "CA", country: "USA" },
                { country: "UK" } ]
}
```

- Shard on { email : 1 }
- Lookup by email hits 1 node
- Index on { “addresses.country” : 1 }

Use Case: User Profiles Multiple Identities

```
{ email: "alvin@10gen.com",
  facebook: "alvin.richards",
  twitter:  "jonnyeight",
  addresses : [ { state : "CA", country: "USA" },
                { country: "UK" }
 ]
}
```

- Shard on { email:1 }
- Lookup by email hits 1 node
- Lookup by twitter or facebook is scatter gather

Use Case: User Profiles
Multiple Identities – linking

```
identities
{ type: "email", val: "alvin@10gen.com", info: "1200-42"}
{ type: "fb",    val: "alvin.richards",  info: "1200-42"}
{ type: "tw",    val: "jonnyeight",      info: "1200-42"}

info
{ _id: "1200-42",
  addresses : [ { state : "CA", country: "USA" },
                { country: "UK" }]
}
```

- Shard identities on { type : 1, val : 1 }
- Lookup by type & val hits 1 node
- Shard info on { _id: 1 }
- Lookup on _id hits one node

Use Case: Photos

```
{ photo_id :  ???? , data : <binary> }
```

What's the right key?

- auto increment
- MD5(data)
- month() + MD5(data)

Use Case: Logging

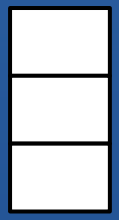
```
{ machine: “app.foo.com”,
  app:  “apache” ,
  when: “2010-12-02:11:33:14”, data : XXX }
```

Possible Shard keys

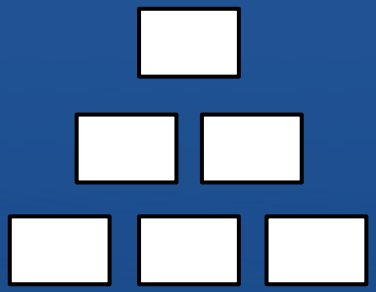
- { machine: 1 }
- { when: 1 }
- { app: 1 }
- { machine: 1, when: -1 }

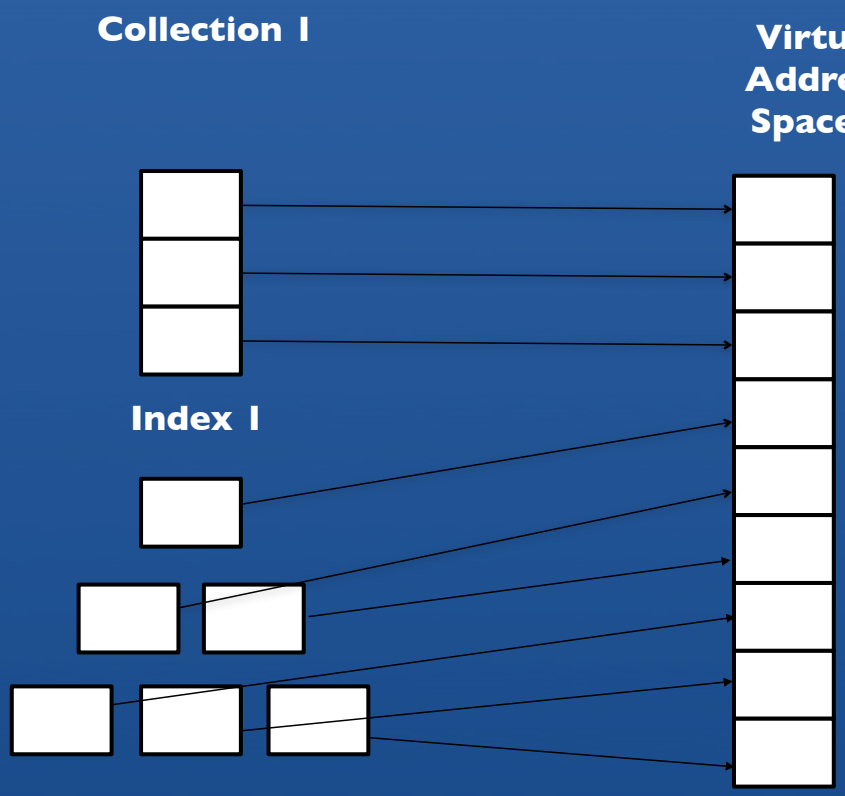
Memory, Virtual Memory & Disk

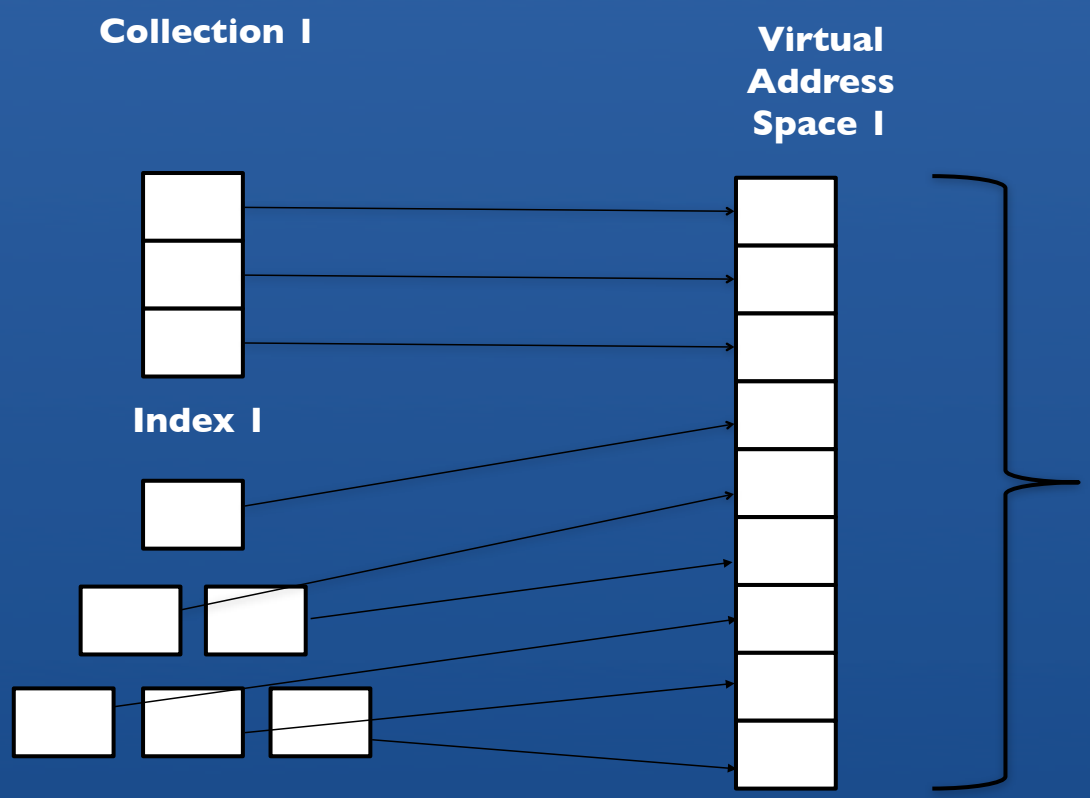
Collection I



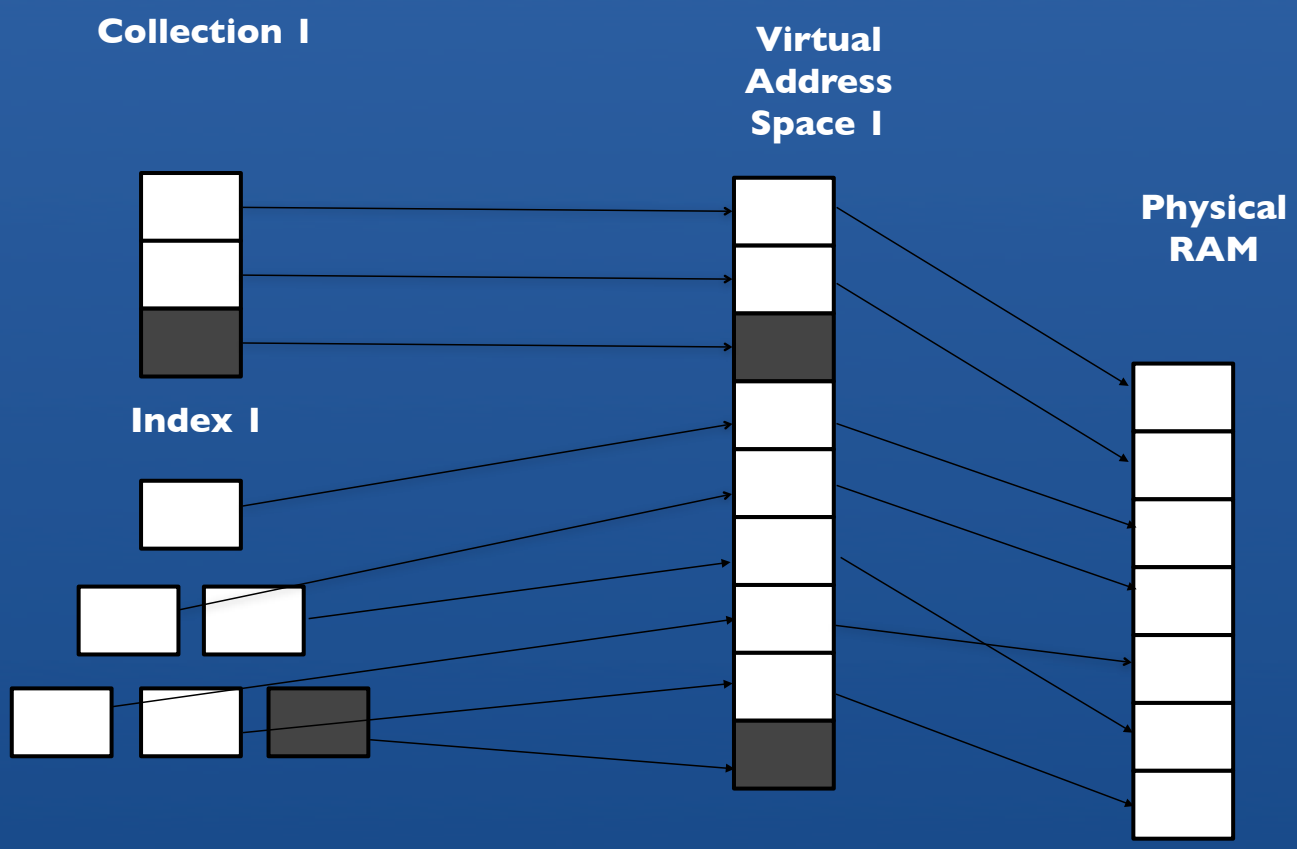
Index I

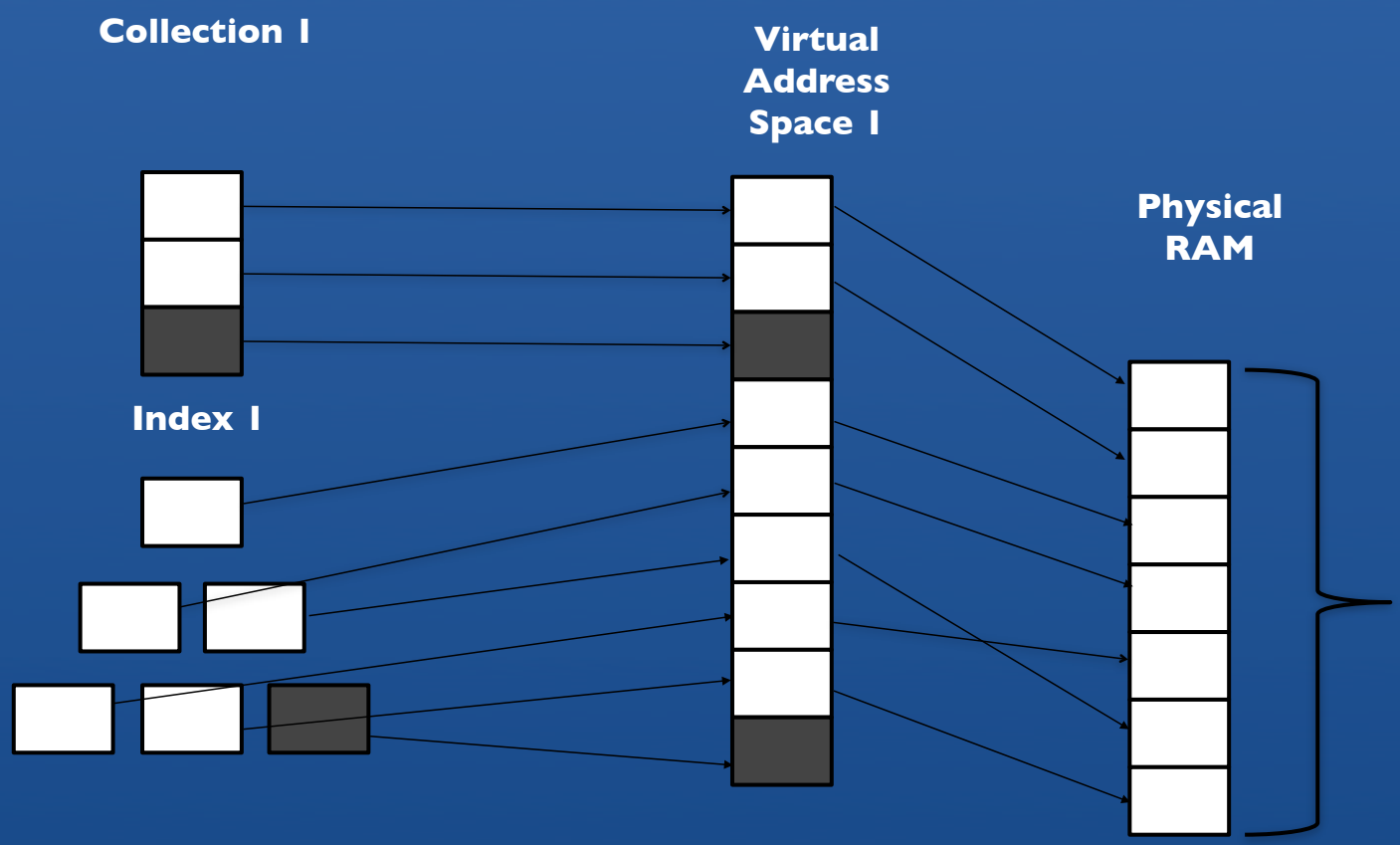


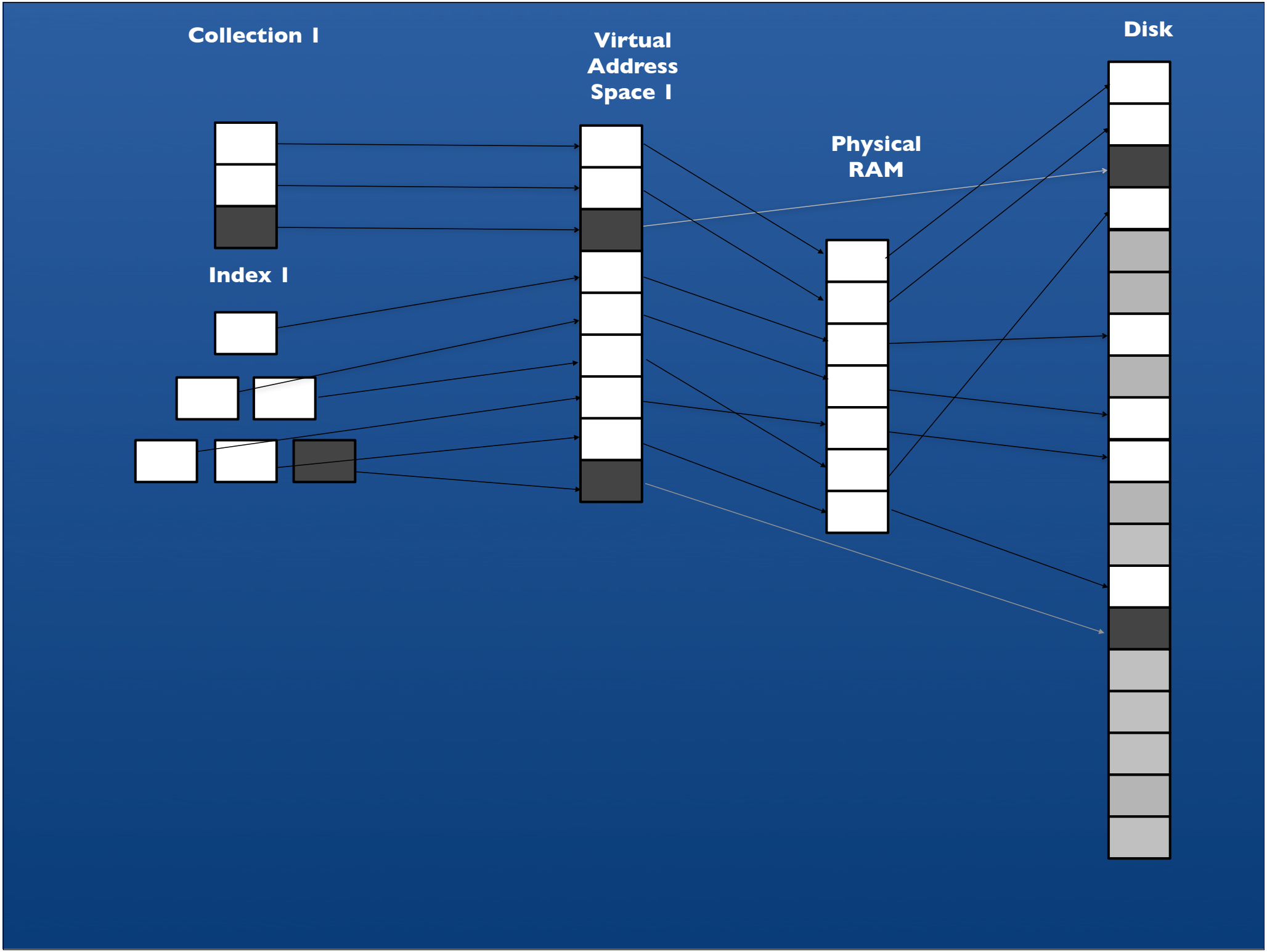

Collection I
Virtual Address Space I
Index I

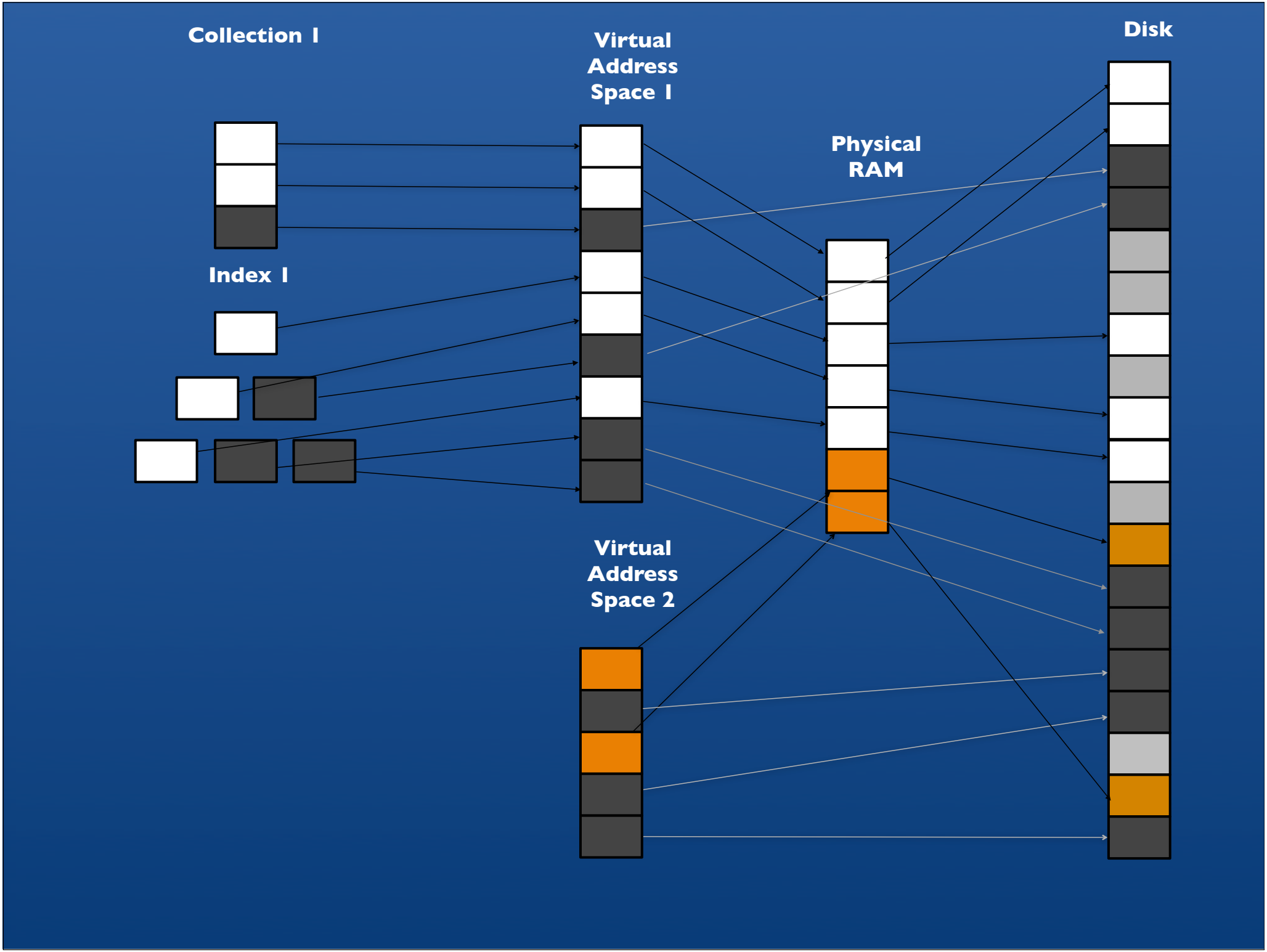


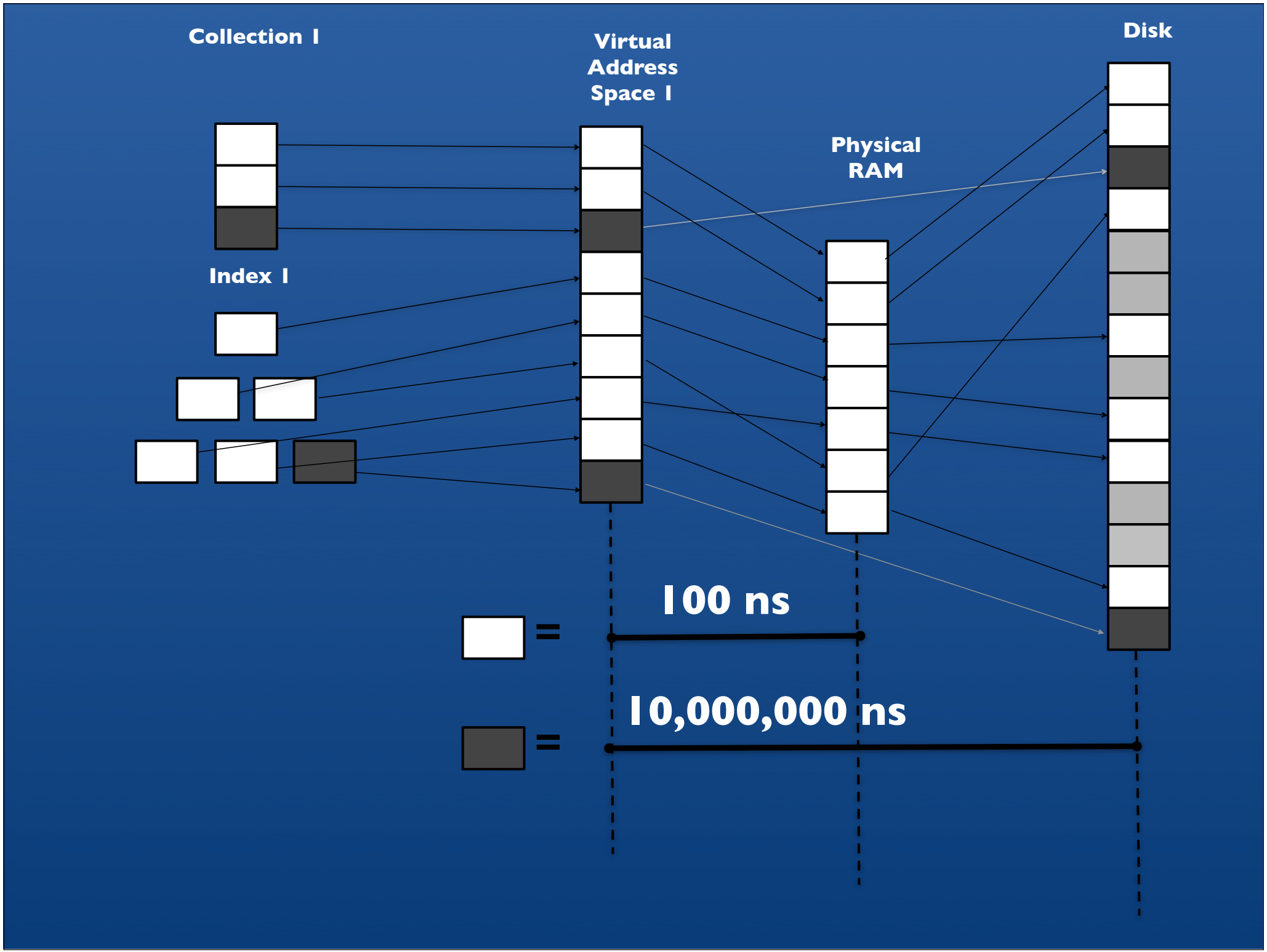
This is your virtual memory size (mapped)


Collection I
Virtual Address Space I
Physical RAM
Index I


Collection 1
Virtual Address Space 1
Physical RAM
Index 1
This is your resident memory size


Collection 1
Virtual Address Space 1
Disk
Physical RAM
Index 1


Collection 1
Index 1
Virtual Address Space 1
Virtual Address Space 2
Physical RAM
Disk


Collection 1
Virtual Address Space 1
Disk
Physical RAM
Index 1
100 ns
=
10,000,000 ns
=

Sizing RAM and Disk

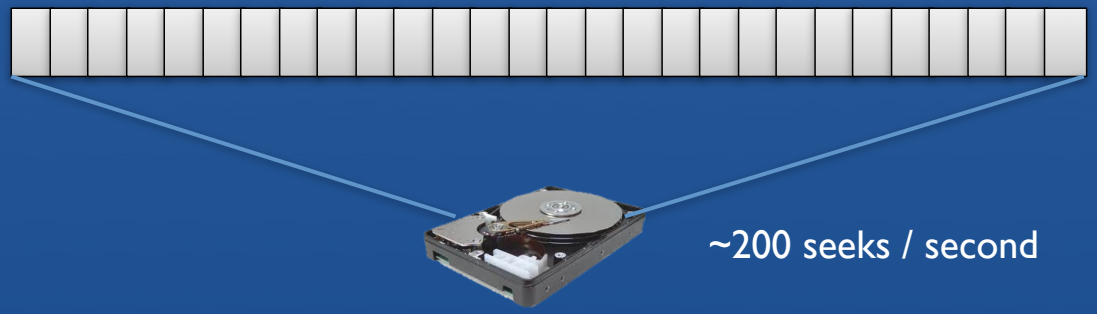
- Working set
- Document Size
- Memory versus disk
- Data lifecycle patterns
    - long tail
    - pure random
    - bulk removes

Figuring out working Set

```
> db.wombats.stats()
{
        "ns" : "test.wombats",
        "count" : 1338330,
        "size" : 46915928,
        "avgObjSize" : 35.05557523181876,
        "storageSize" : 86092032,
        "numExtents" : 12,
        "nindexes" : 2,
        "lastExtentSize" : 20872960,
        "paddingFactor" : 1,
        "flags" : 0,
        "totalIndexSize" : 99860480,
        "indexSizes" : {
                "_id_" : 55877632,
                "name_1" : 43982848
        }
}
```

Disk configurations

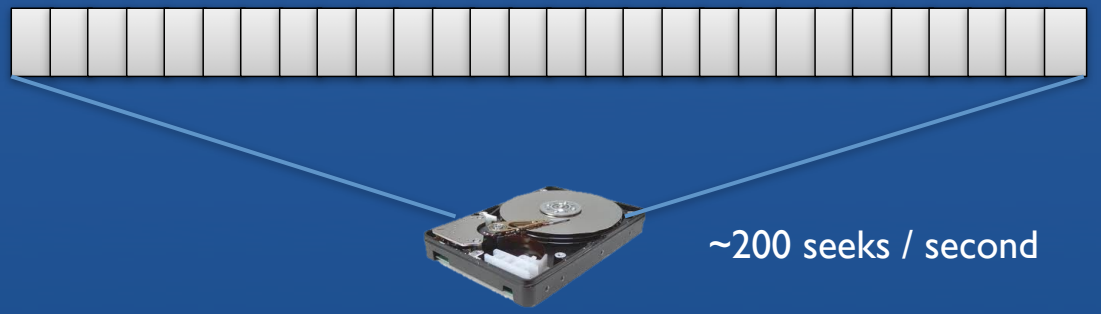
Single Disk ⟶



~200 seeks / second

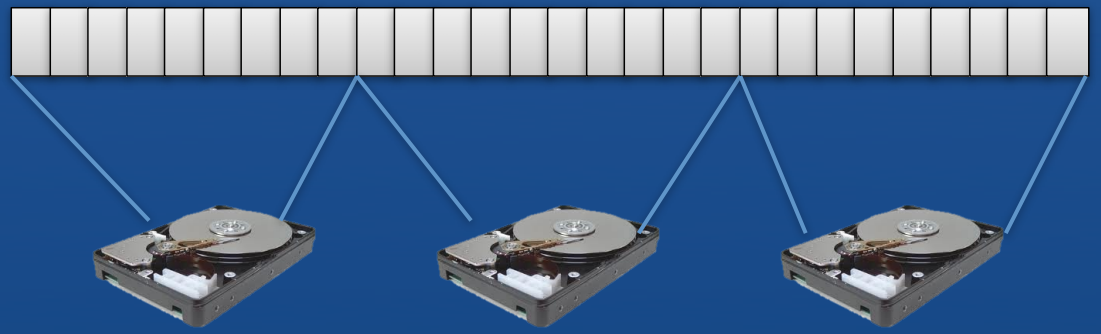
Disk configurations

Single Disk →



~200 seeks / second

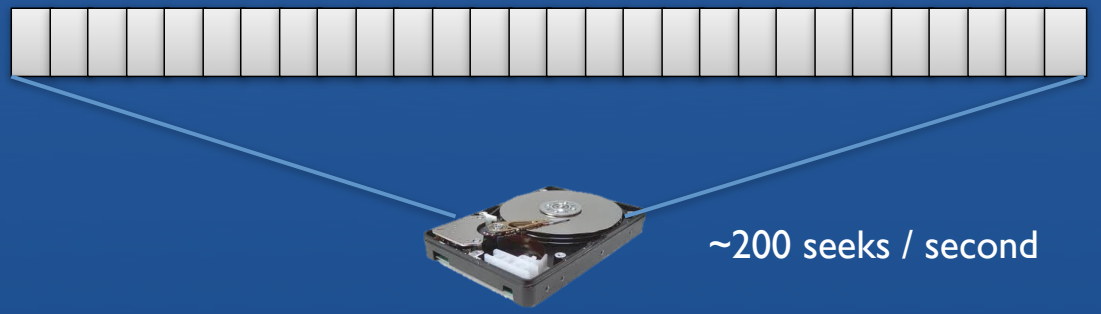
RAID 0 →



~200 seeks / second ~200 seeks / second ~200 seeks / second

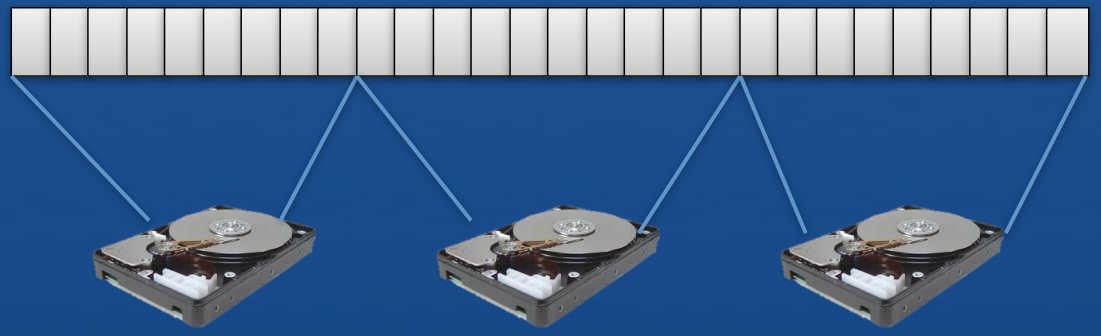
Disk configurations

Single Disk →



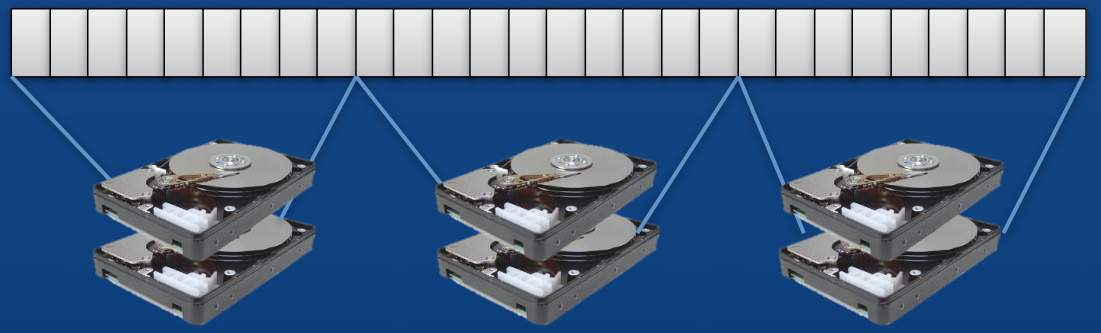
~200 seeks / second

RAID 0 →



~200 seeks / second ~200 seeks / second ~200 seeks / second

RAID 10 →



~400 seeks / second ~400 seeks / second ~400 seeks / second

SSD?

Seek time of 0.1ms vs 5ms

(200 seeks / sec => 10000 seeks / sec)

But expensive

Takeaway

Know how important page faults are

- If you want low latency, avoid page faults

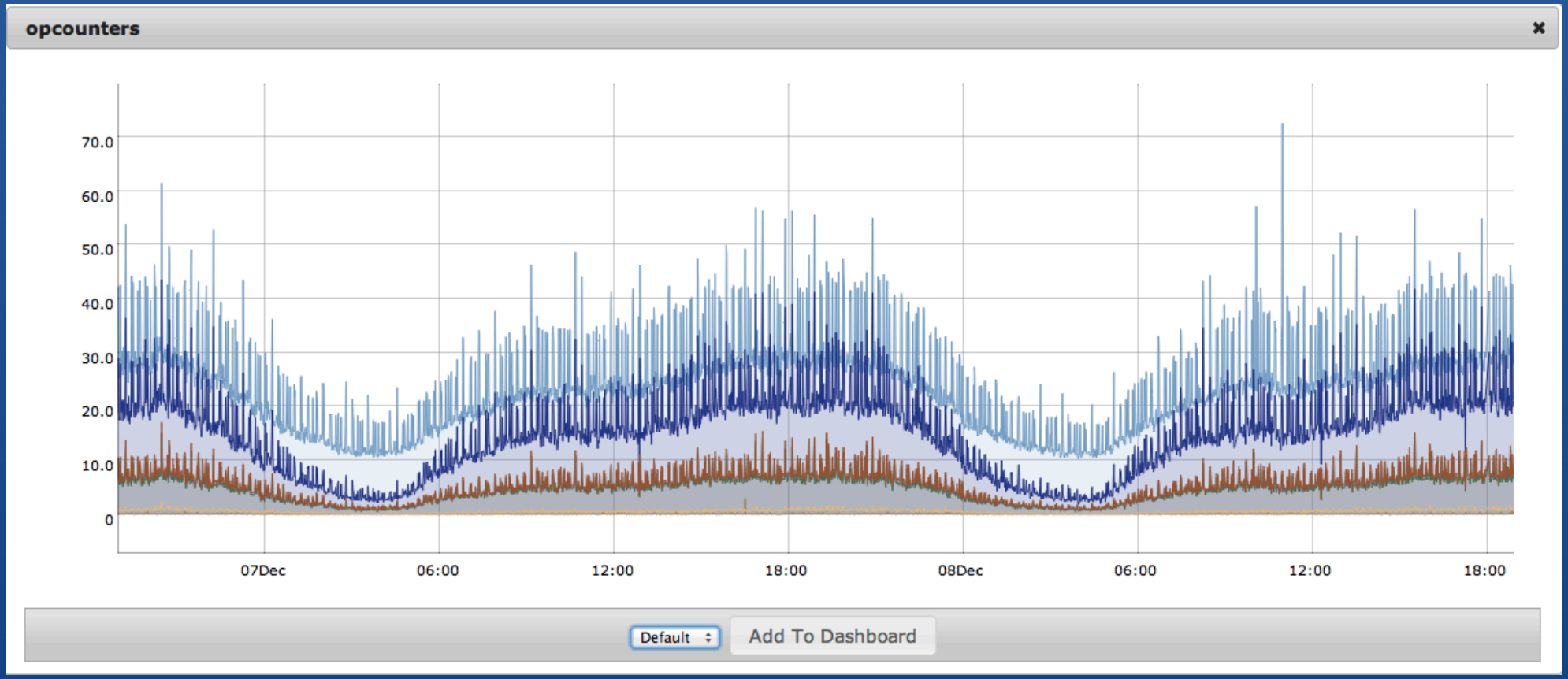
Size memory appropriately

- Avoid page faults, fit working set in RAM
- Collection Data + Index Data

Provision disk appropriately

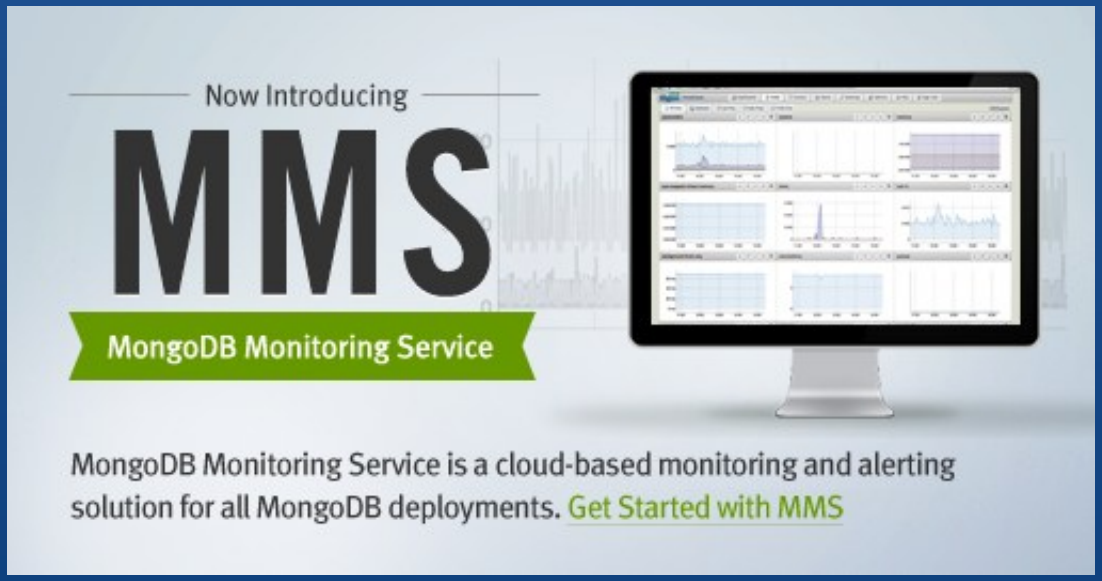
- RAID10 is recommended
- SSD's are fast, if you can afford them

Monitoring is your friend!



MongoDB Monitoring Service

- SaaS solution providing instrumentation and visibility into MongoDB systems
- Included in the 10gen commercial subscriptions
- Free community version available
- 3,500+ customers signed up and using service



Load Testing!

Understand what you think the system should do

- Load and test your hypothesis
  - Use the profiler e.g. db.setProfilingLevel(2)
- Use a trending monitoring tool to analyze
  - MMS, munin, etc.

Backups!

- mongodump versus storage snapshots
  - mongodump reads all data (page faults), write to file
  - snapshots avoid page faults
  - journaling avoids need to fsync+lock
- Restore
  - member, whole rep set, whole cluster
- Don't forget your config dbs in a sharded cluster

Plan for the worst

Not everything will go to plan

- Have a run/play book
- Practice basic procedures
  - backups, restore
  - rolling upgrade
  - failing over primary

Who is 10gen ?

- Began the MongoDB project
  - Core maintainers
  - Own the copyright, distribute under GPL 3.0
- Provide MongoDB services
  - Support, Subscribers Build, Monitoring
  - Consulting, Training
- 100+ employees
- Offices in New York, Palo Alto, London and Dublin
- Investors: Sequoia, Flybridge, Union Square

download at mongodb.org

We're Hiring !

http://www.10gen.com/jobs

conferences, appearances, and meetups

http://www.10gen.com/events

Facebook

http://bit.ly/mongofb

Twitter

@mongodb

LinkedIn

http://linkd.in/joinmongo

10gen mongoDB