elastic{ON}15

Life of a document in Elasticsearch

Alexander Reelsen - @spinscale

Boaz Leskes - @bleskes

Agenda

Life of a document in Elasticsearch

Agenda

Life of a document in Elasticsearch

How, when and where is a document stored/processed in Elasticsearch?

About us

Alexander Reelsen

Core & Shield developer

Boaz Leskes

Core developer & Marvel lead

Exhibit A: A JSON document

```
{
  "name" : "Elasticsearch - The definitive guide",
  "pages" : 721, "isbn13" : "978-1449358549",
  "authors" : [ "Clinton Gormley" , "Zachary Tong" ],
  "publish_date" : "2015/01/31",
  "category" : "IT > Search Engines",
  "description" : "Whether you need full-text search
or real-time analytics...",
}
```

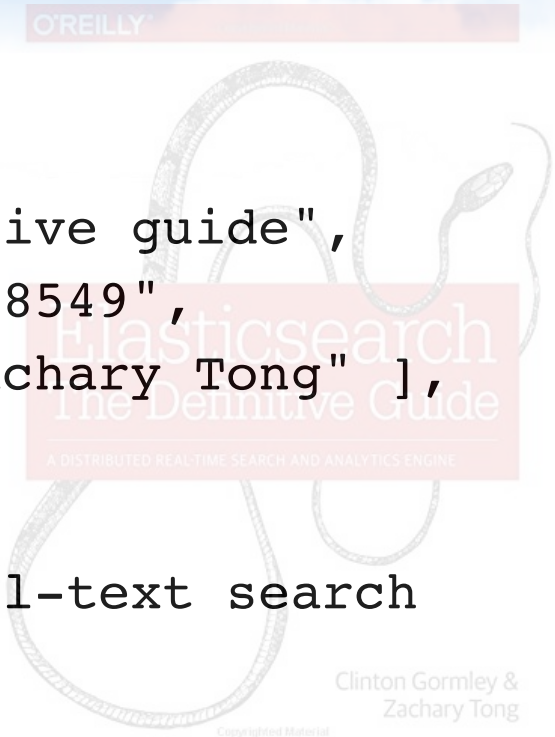
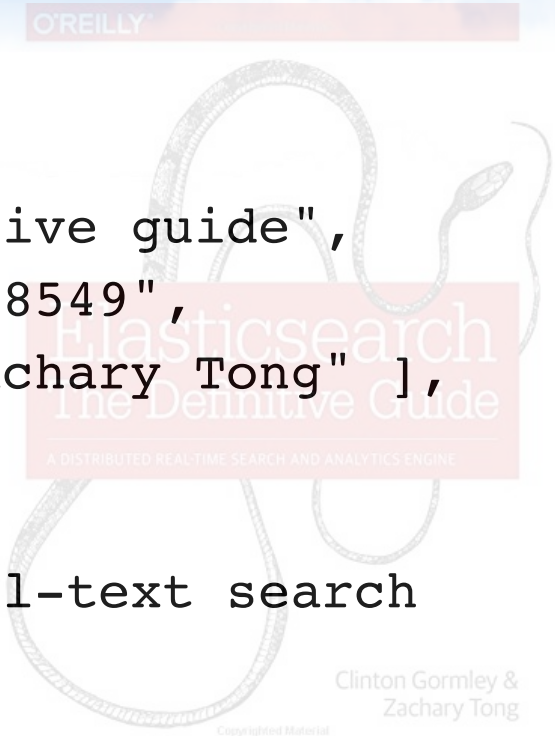
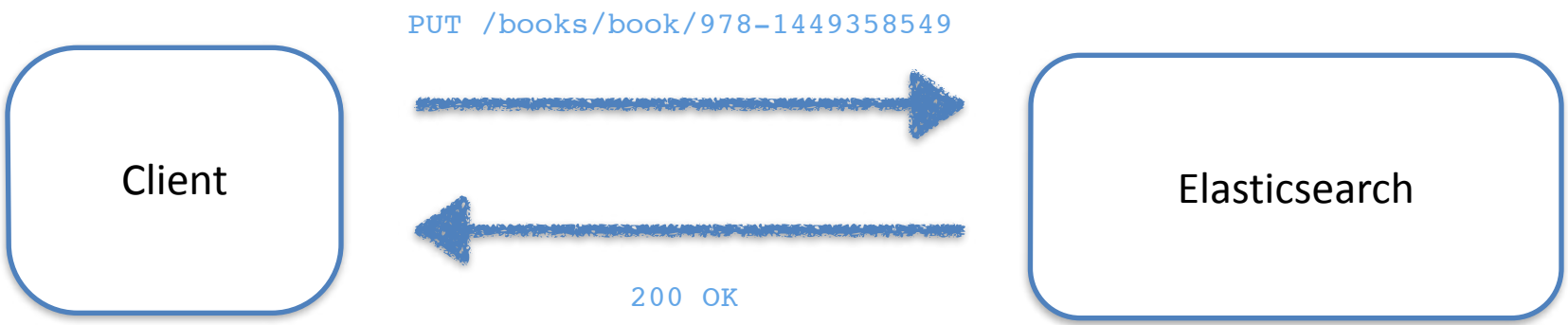


Exhibit A: A JSON document

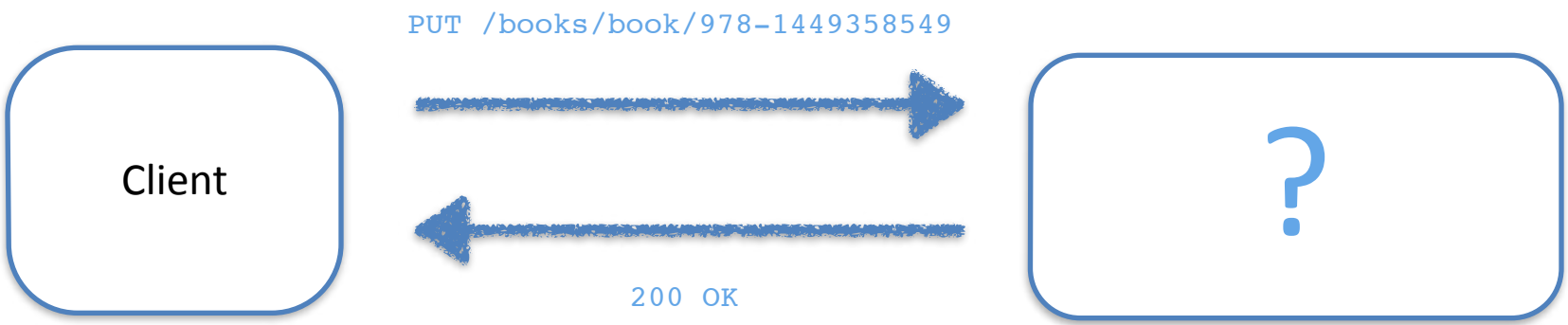
```
PUT /books/book/978-1449358549
{
  "name" : "Elasticsearch - The definitive guide",
  "pages" : 721, "isbn13" : "978-1449358549",
  "authors" : [ "Clinton Gormley" , "Zachary Tong" ],
  "publish_date" : "2015/01/31",
  "category" : "IT > Search Engines",
  "description" : "Whether you need full-text search
or real-time analytics...",
}
```



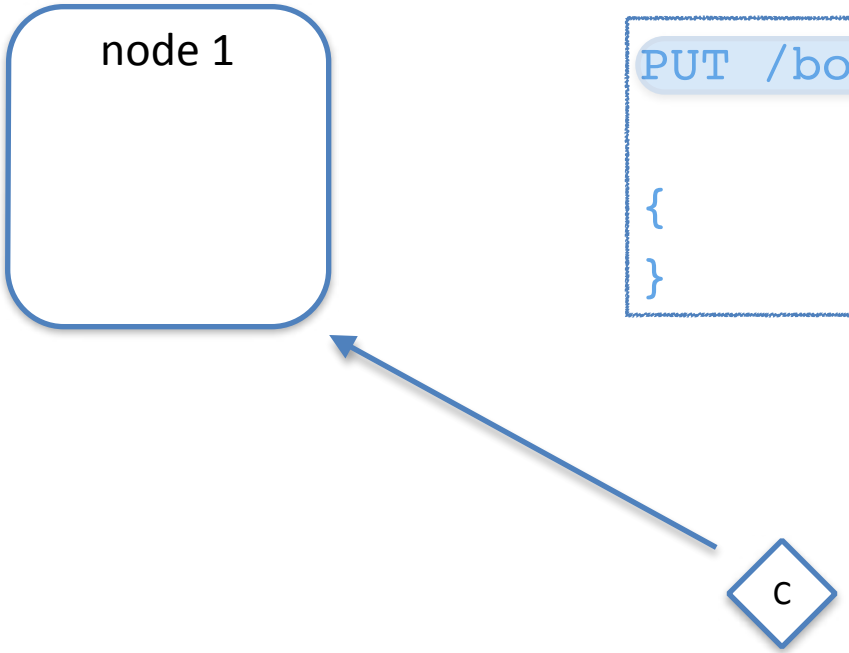
Indexing a document



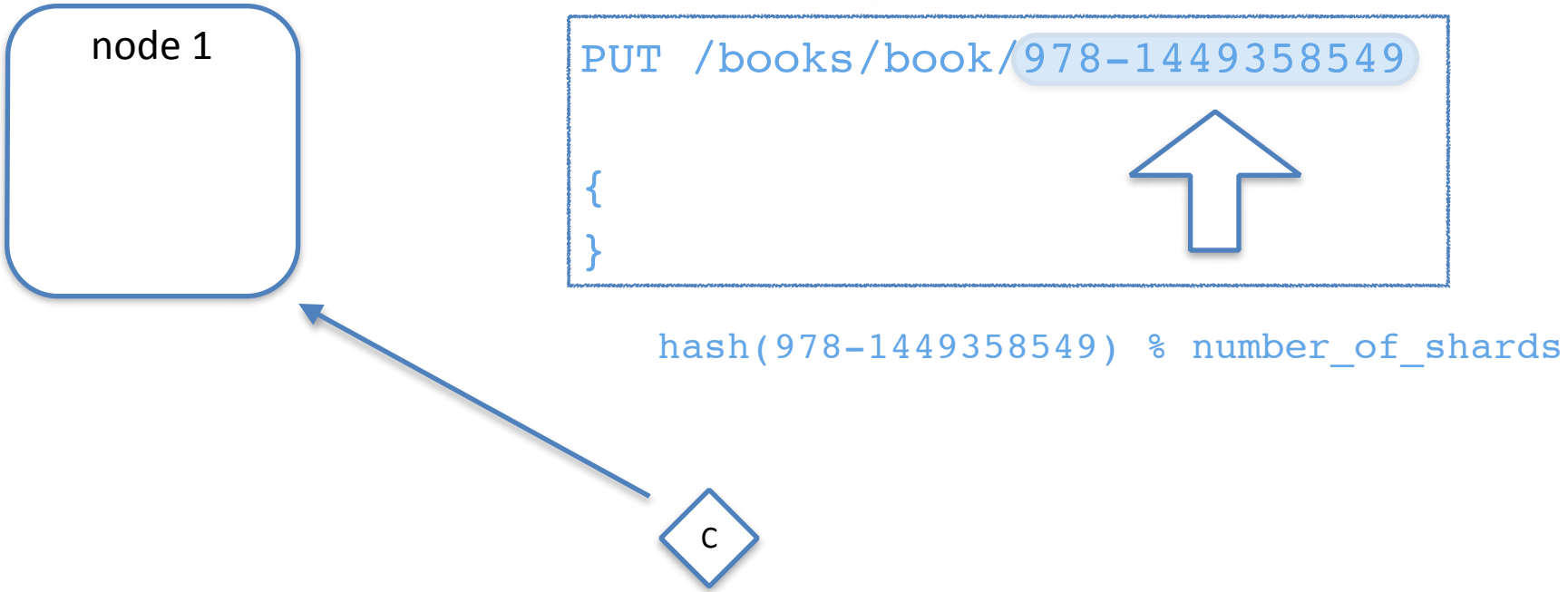
Indexing a document



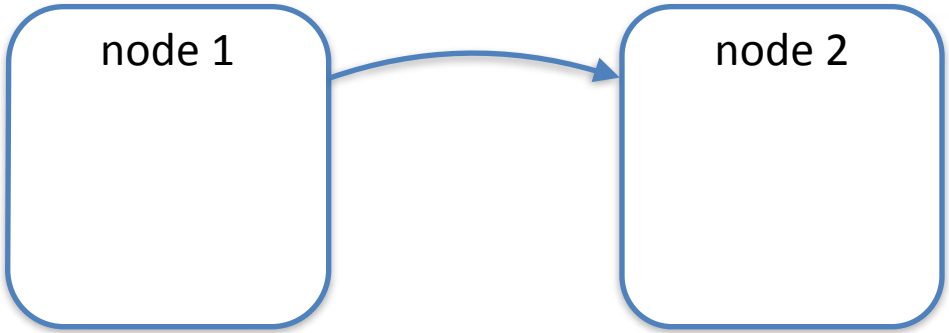
Receiving: Where to store it?



Receiving: Where to put it?



Receiving: Where to put it?




```
PUT /books/book/978-1449358549


{
}
```

C

Next step: Analysis

Each field is put into the inverted index

```
{
  "name" : "Elasticsearch - The definitive guide",
  "pages" : 721, "isbn13" : "978-1449358549",
  "authors" : [ "Clinton Gormley" , "Zachary Tong" ],
  "publish_date" : "2015/01/31",
  "category" : "IT > Search Engines",
  "description" : "Whether you need full-text search
or real-time analytics...",
}
```

Analysis

Each field is put into the inverted index

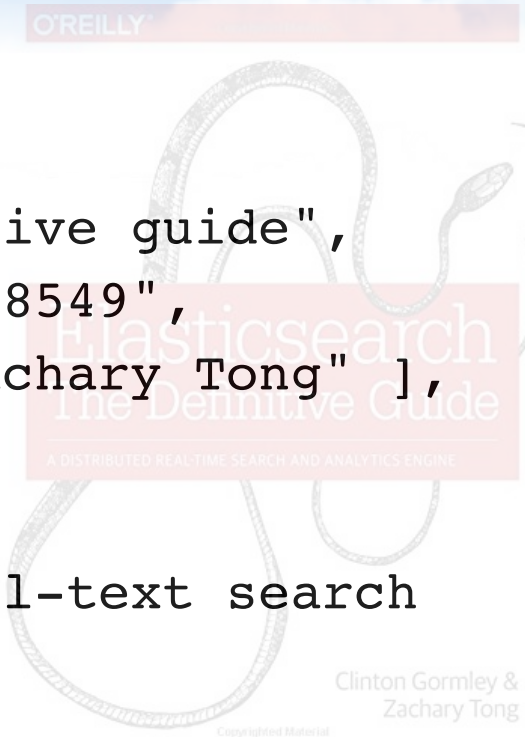
Input	*Tokenization*	*Analysis*
Clinton Gormley	*[Clinton, Gormley]*	*[clinton, gormley]*
Zachary Tong	*[Zachary, Tong]*	*[zachary, tong]*

Analysis

Resulting in a postings list

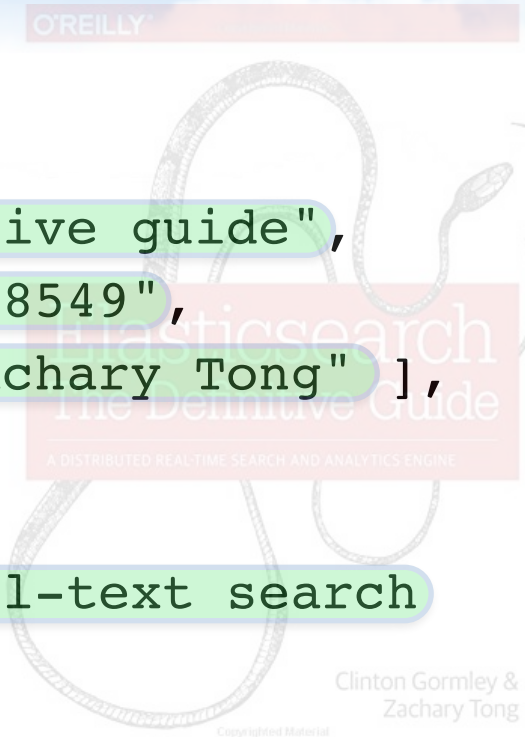
Value	*Document ID*
clinton	***1***
gormley	***1***
tong	***1***
zachary	***1***

Lucene at work: Field by field




```
{
  "name" : "Elasticsearch - The definitive guide",
  "pages" : 721, "isbn13" : "978-1449358549",
  "authors" : [ "Clinton Gormley" , "Zachary Tong" ],
  "publish_date" : "2015/01/31",
  "category" : "IT > Search Engines",
  "description" : "Whether you need full-text search
or real-time analytics...",
}
```

Special case: _all field




```
{
   "name" : "Elasticsearch - The definitive guide",
   "pages" : 721, "isbn13" : "978-1449358549",
   "authors" : [ "Clinton Gormley" , "Zachary Tong" ],
   "publish_date" : "2015/01/31",
   "category" : "IT > Search Engines",
   "description" : "Whether you need full-text search
or real-time analytics...",
}
```

Special case: _source field

```
{
  "name" : "Elasticsearch - The definitive guide",
  "pages" : 721, "isbn13" : "978-1449358549",
  "authors" : [ "Clinton Gormley" , "Zachary Tong" ],
  "publish_date" : "2015/01/31",
  "category" : "IT > Search Engines",
  "description" : "Whether you need full-text search
or real-time analytics...",
}
```

Multifields

Sorting & Aggregations

```
PUT /books/book/_mapping
{
  "book" : {
    "properties" : {
      "authors": {
        "type": "string", "analyzer": "standard",
        "fields": { "raw": { "type": "string", "index": “not_analyzed" } }
      }
    }
  }
}
```

Multifields

Resulting in multiple postings lists

Field: authors

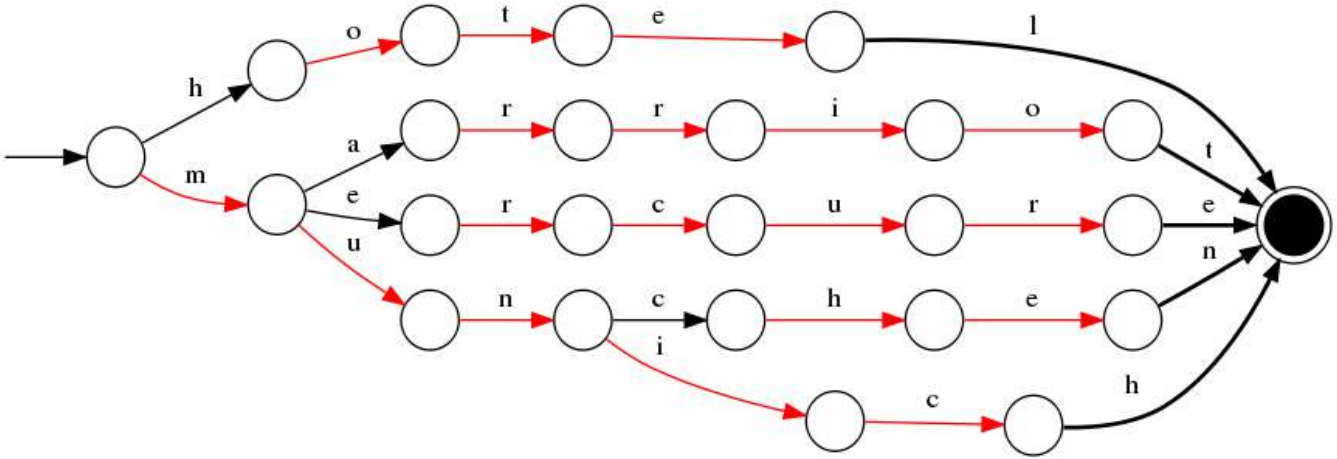
Value	Document ID
clinton	1
gormley	1
tong	1
zachary	1

Field: authors.raw

Value	Document ID
Clinton Gormley	1
Zachary Tong	1

Completion suggester

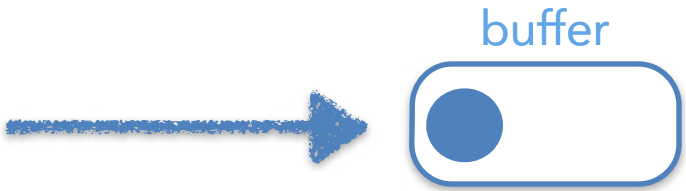
- Auto-complete, search-as-you-type
- Data structure is stored on disk on indexing, then loaded into memory



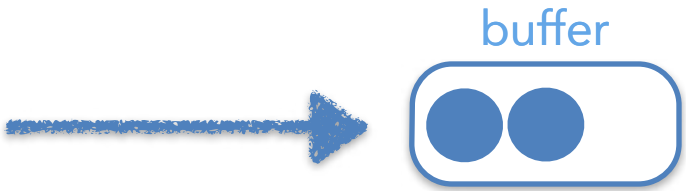
Term vectors

- Handy when hit highlighting huge documents
- An index of a single document
- Allows to easily find sorted unique terms as well as original positions and offsets

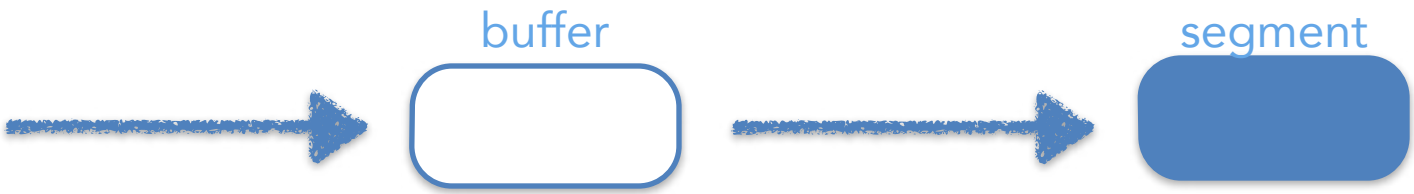
Buffering documents



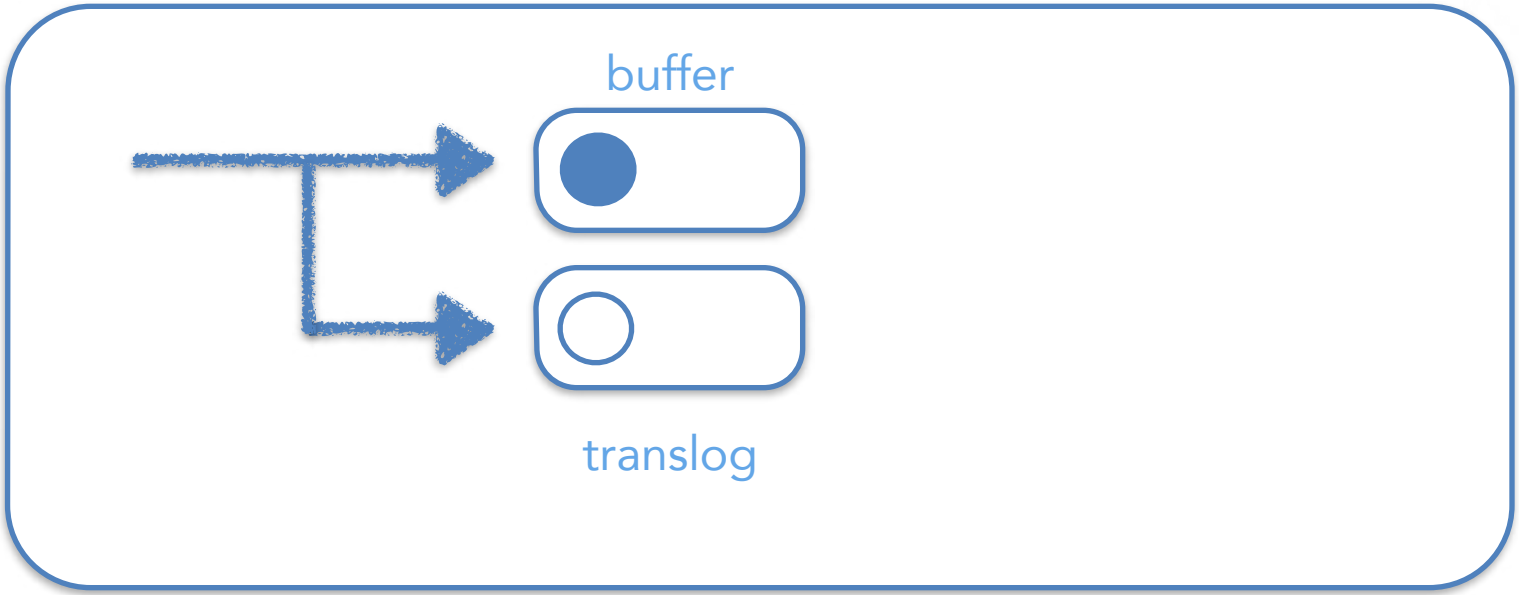
Buffering documents



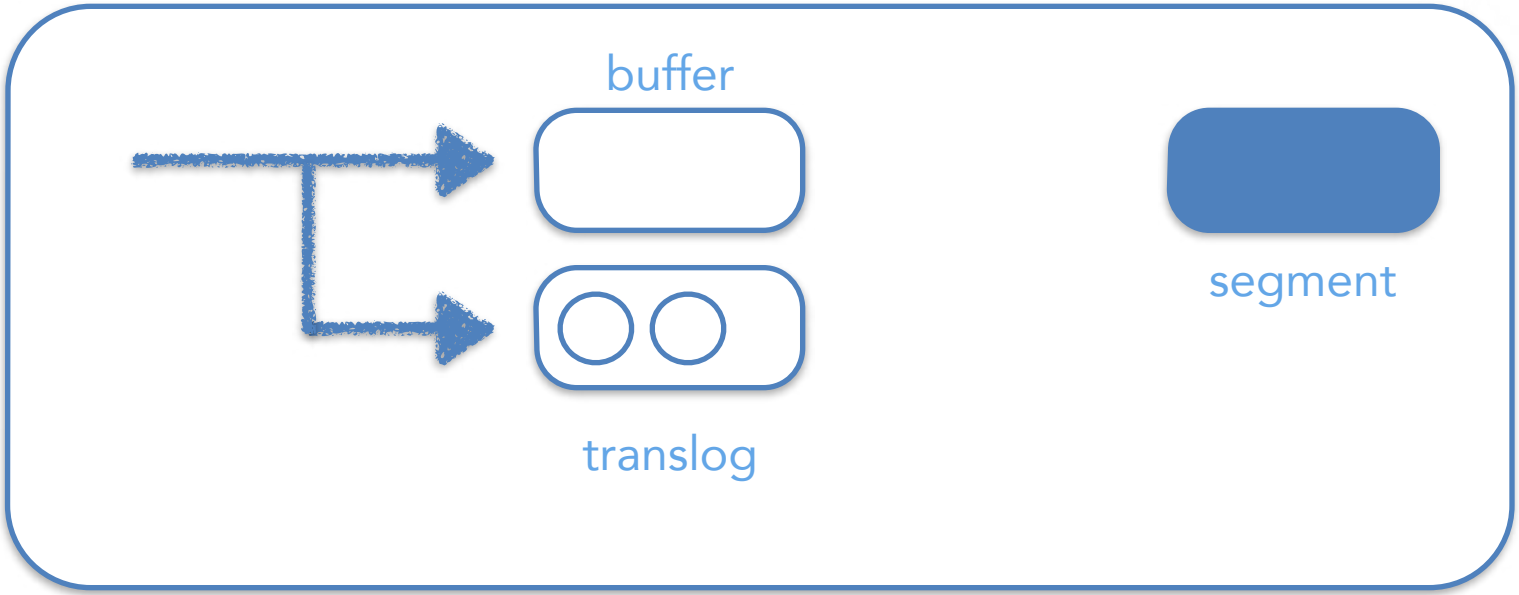
Buffering documents



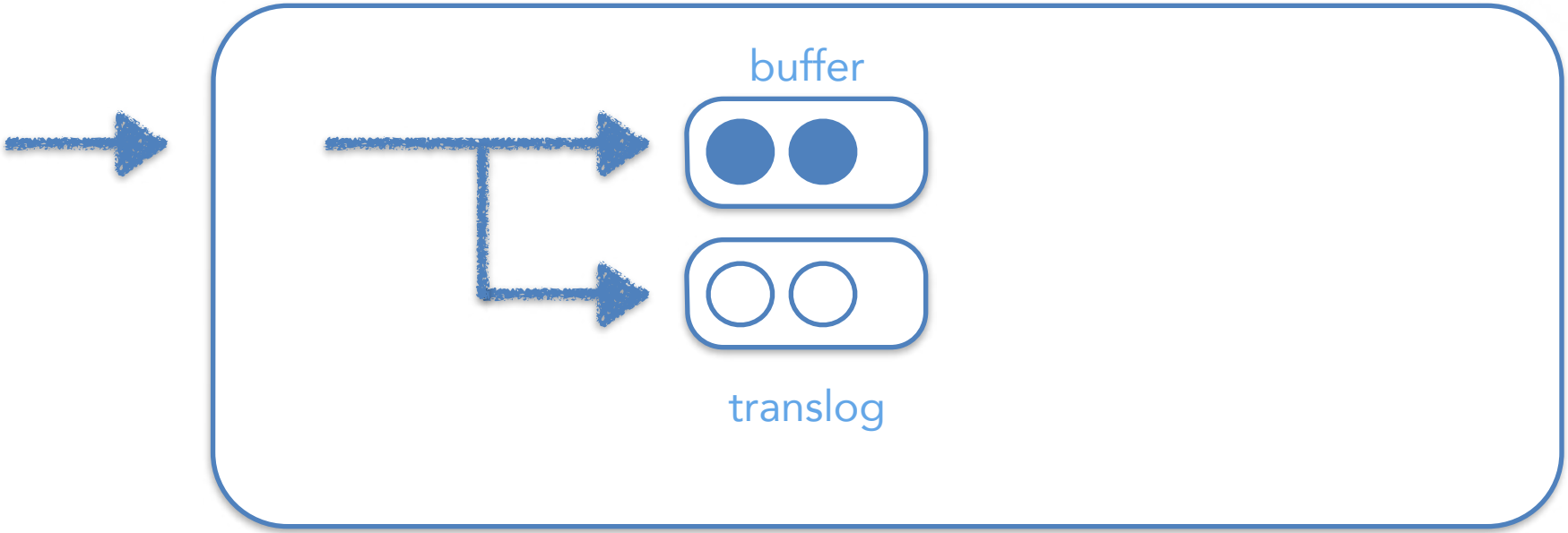
Elasticsearch refresh



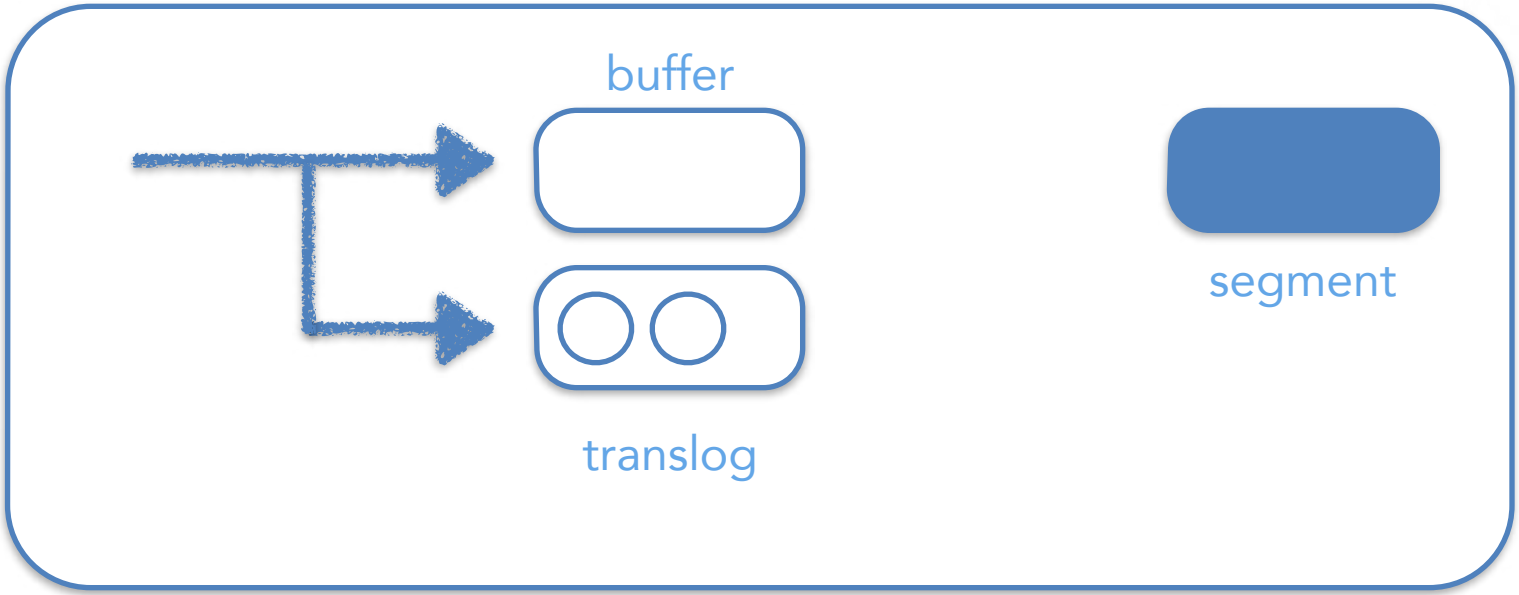
Elasticsearch refresh



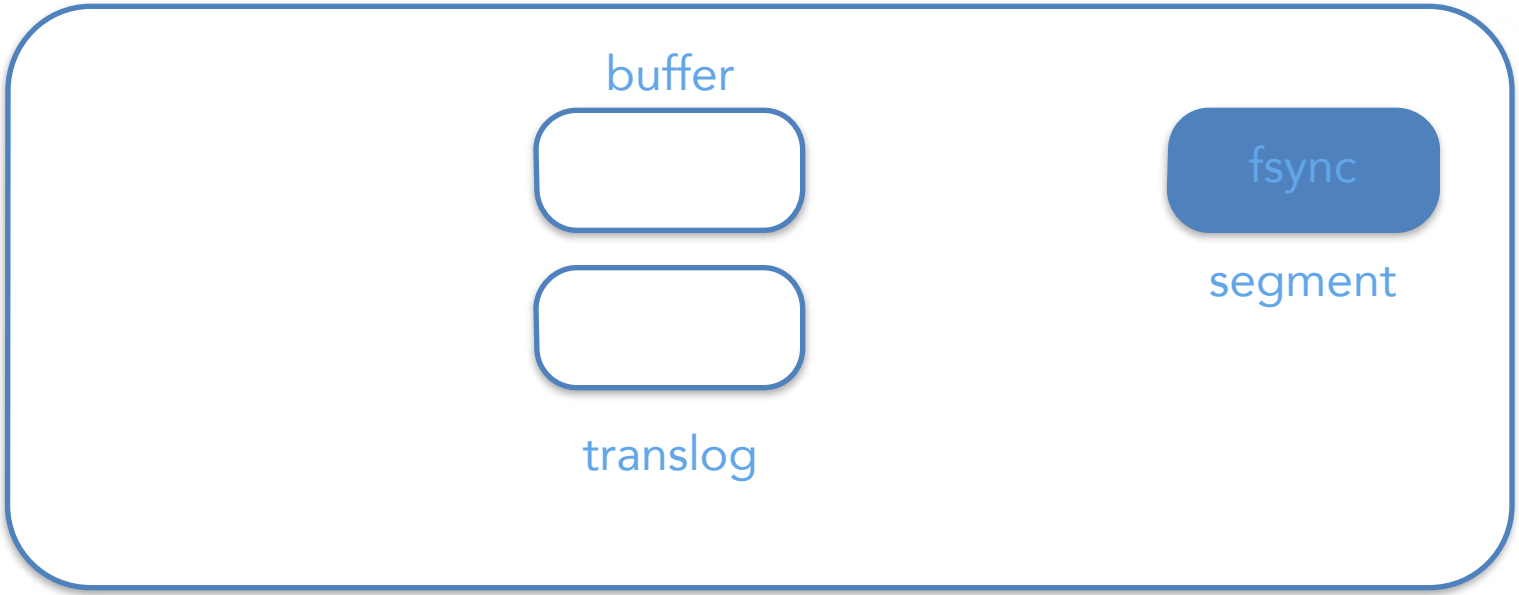
Indexing a document



Elasticsearch refresh



Elasticsearch flush



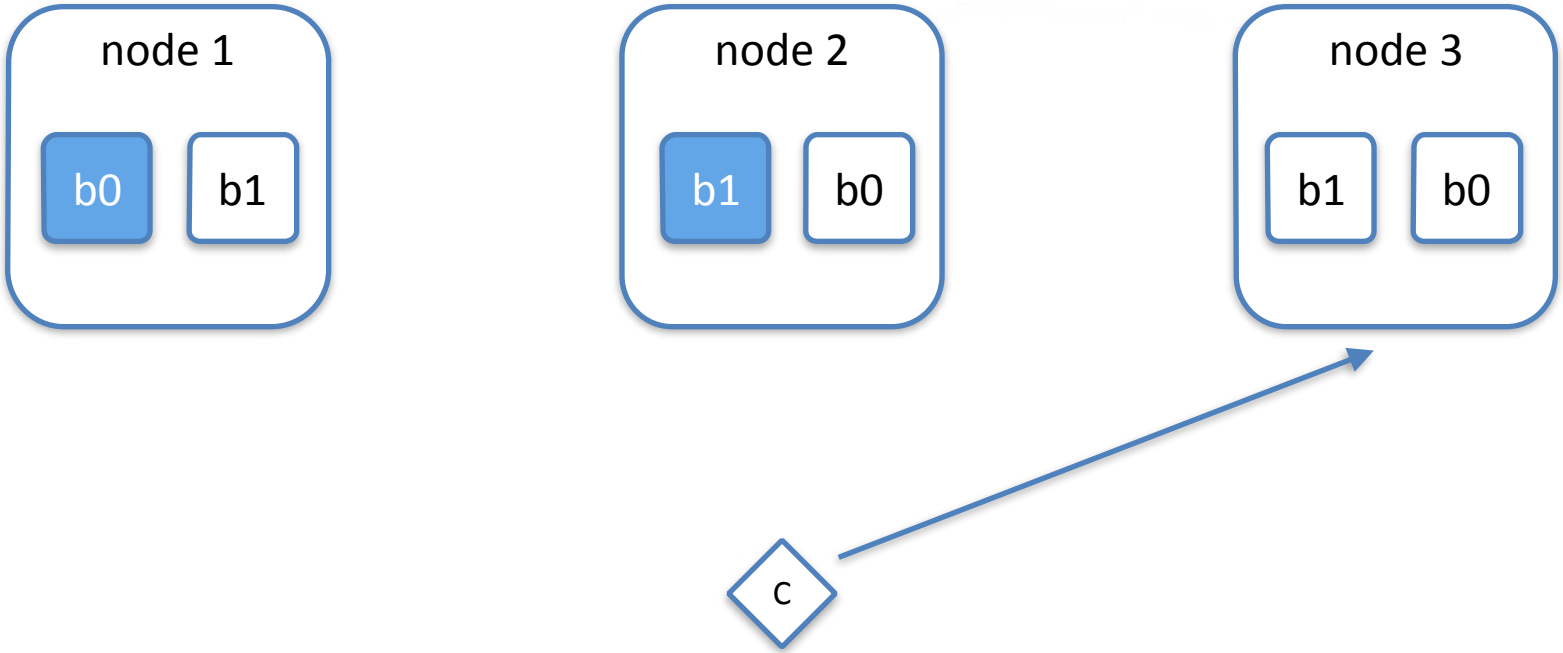
Where are we now?

- Document received via network
- Document received on the correct node
- Document analyzed, stored in translog & buffer

- Client is still waiting for a response

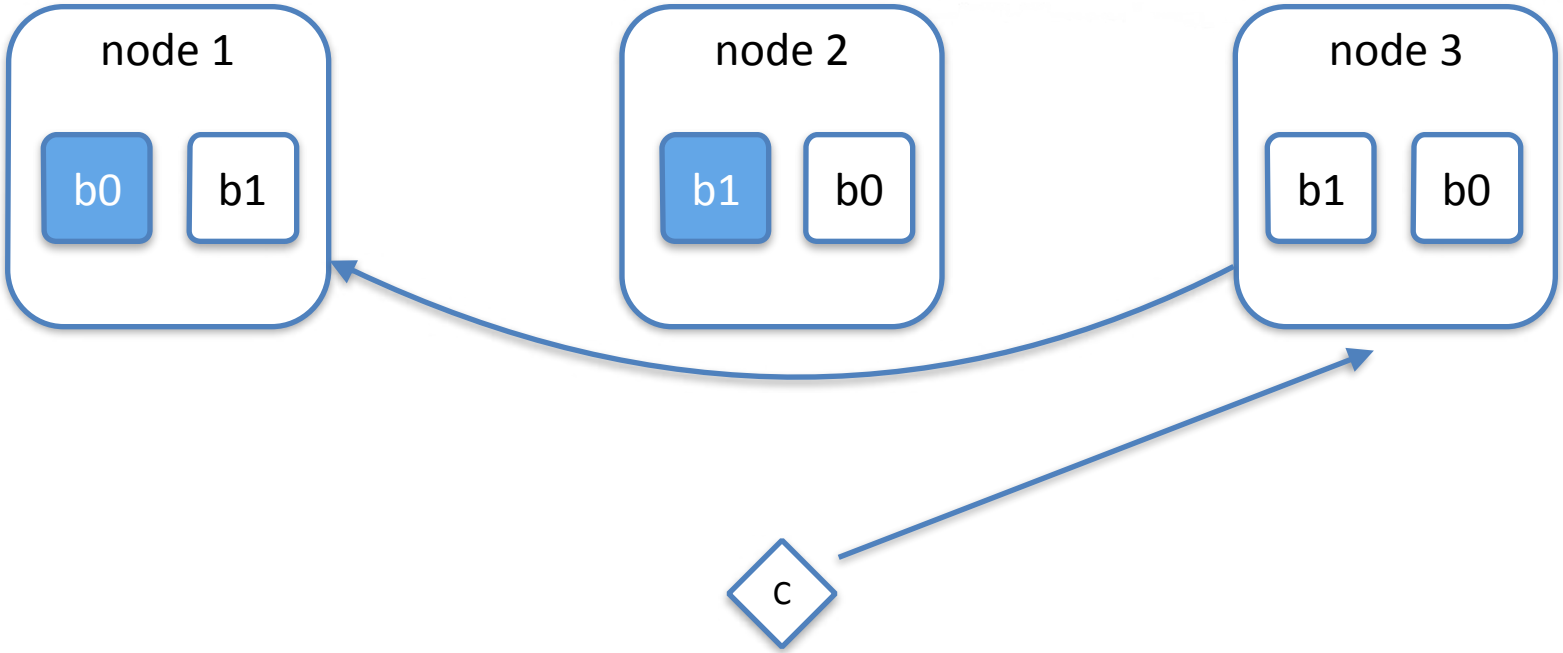
Replication on indexing

Document level replication



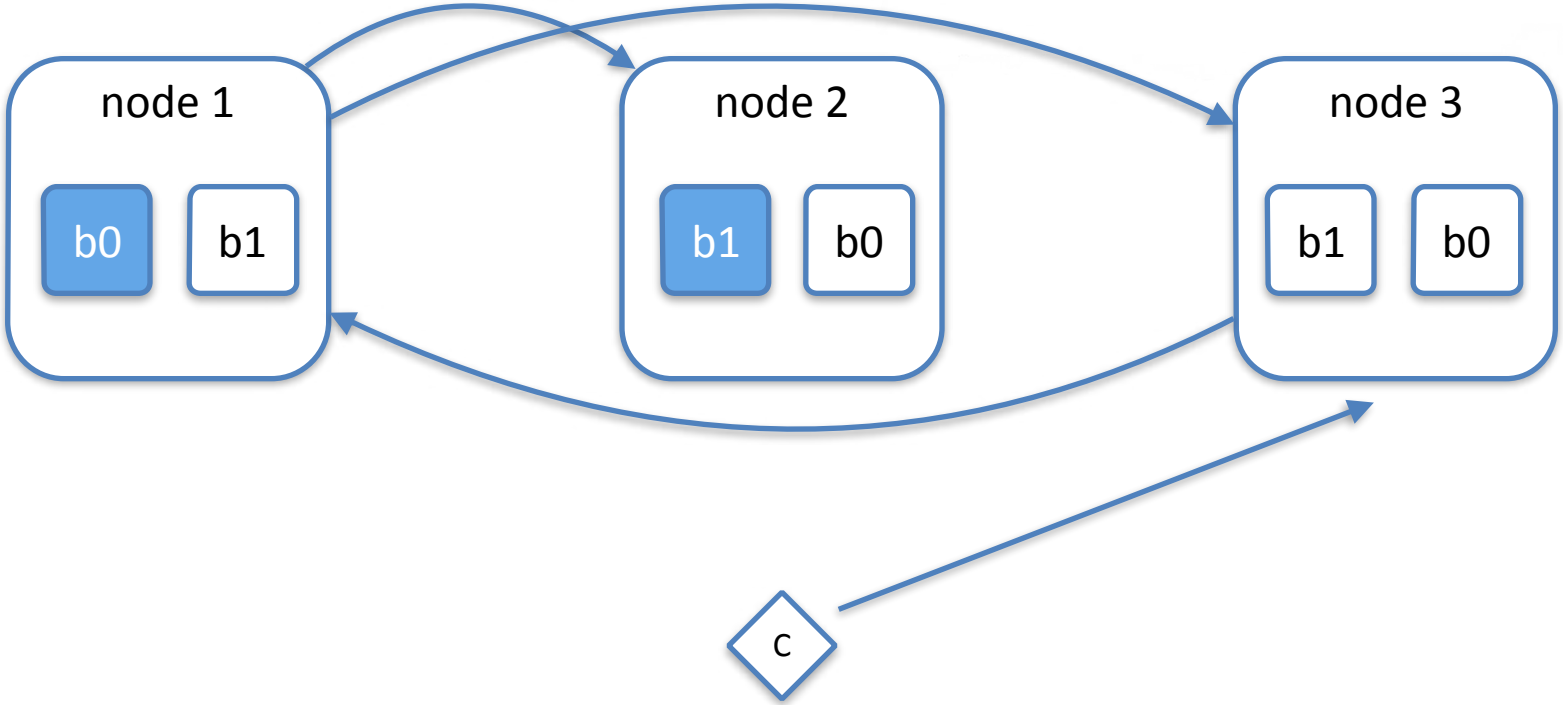
Replication on indexing

Document level replication



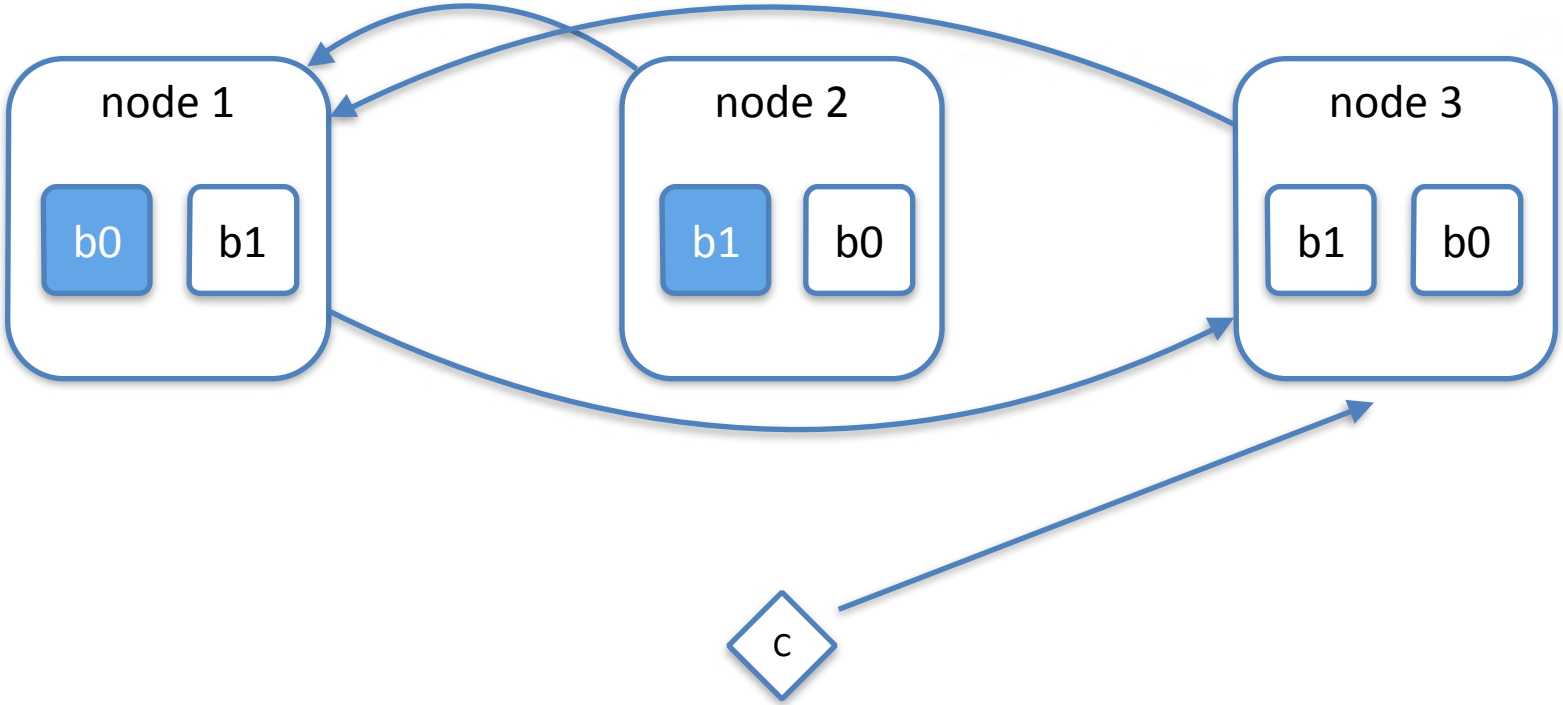
Replication on indexing

Document level replication



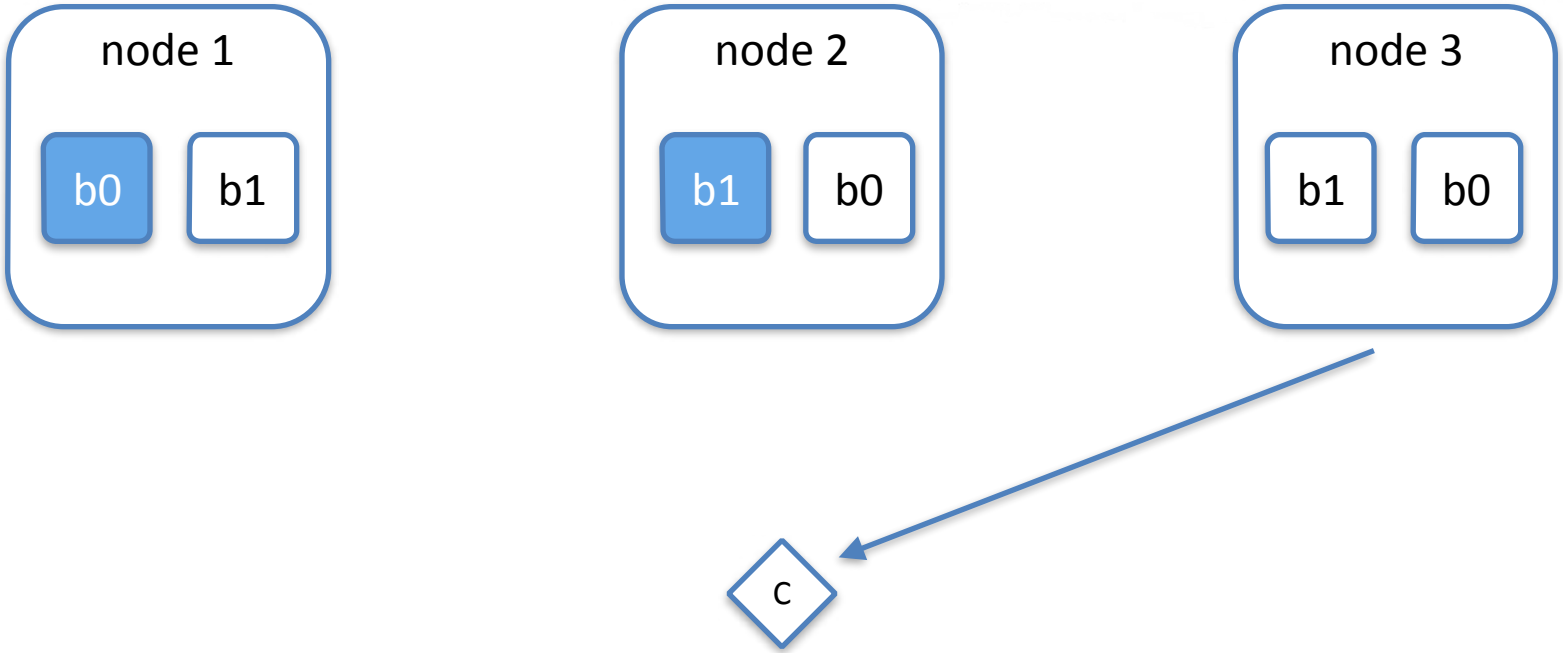
Replication on indexing

Document level replication



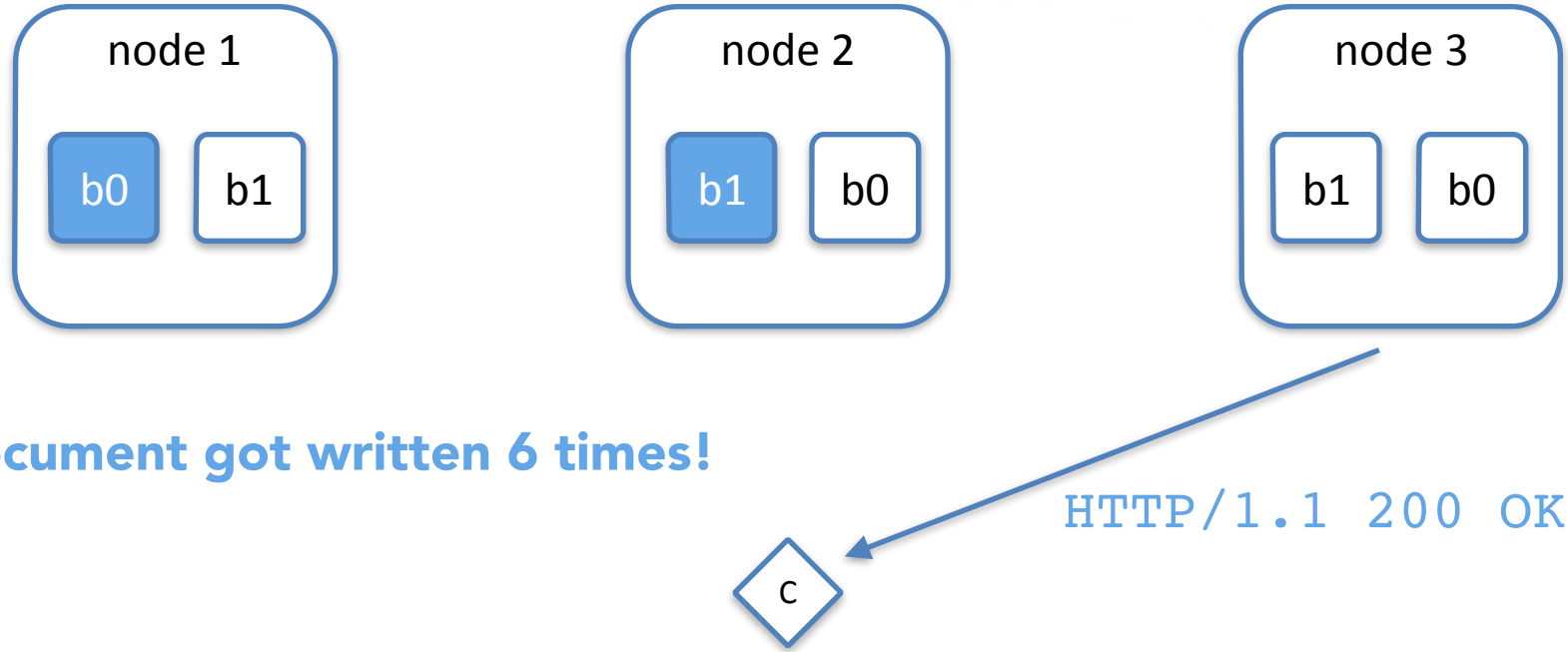
Replication on indexing

Document level replication



Replication on indexing

Document level replication

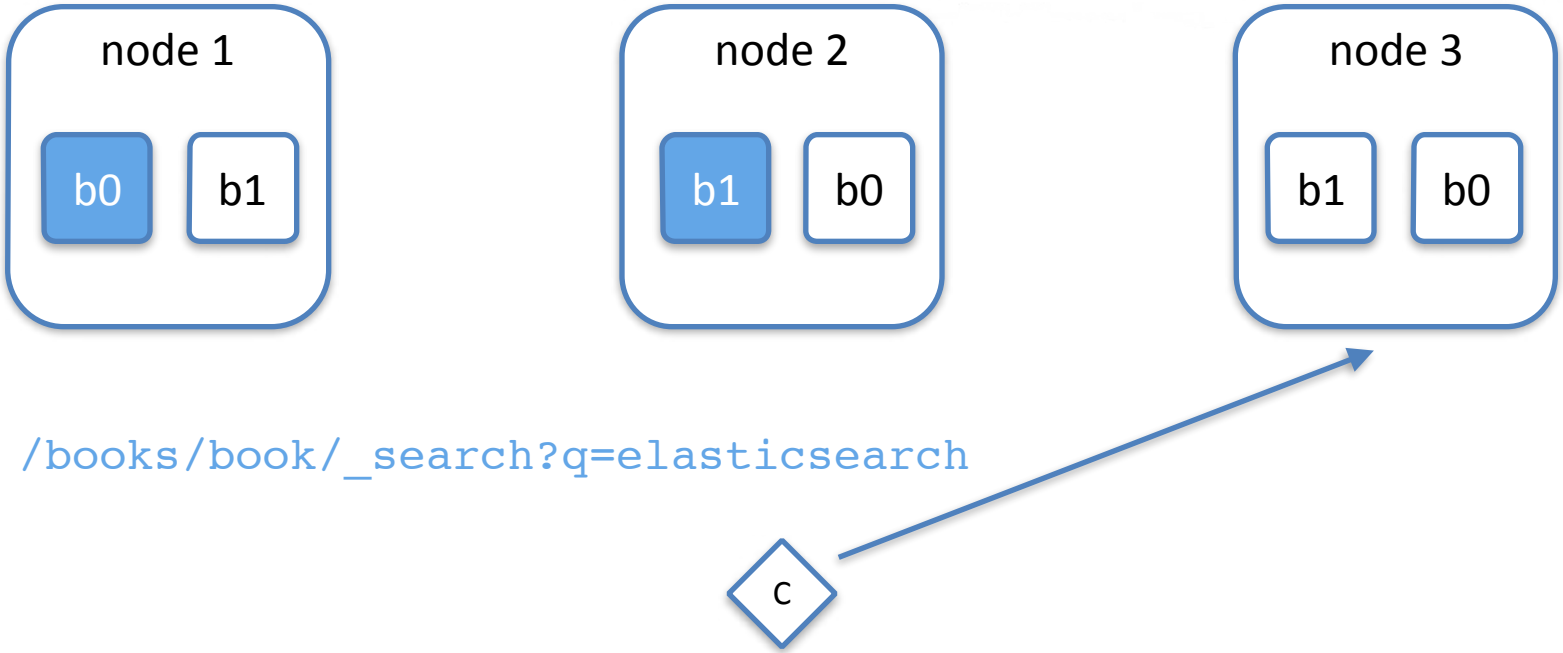


Document got written 6 times!

Searching

Searches: Collecting node

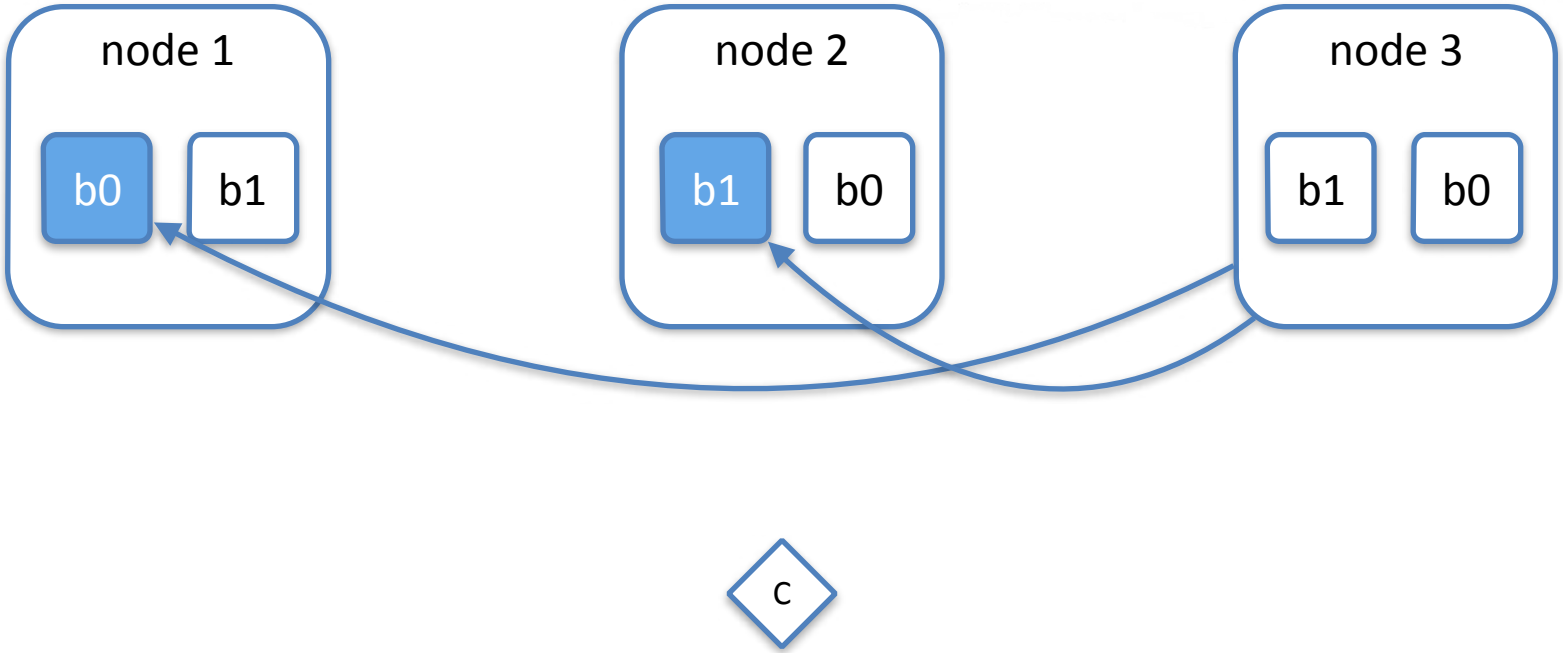
query hits node




```
GET /books/book/_search?q=elasticsearch
```

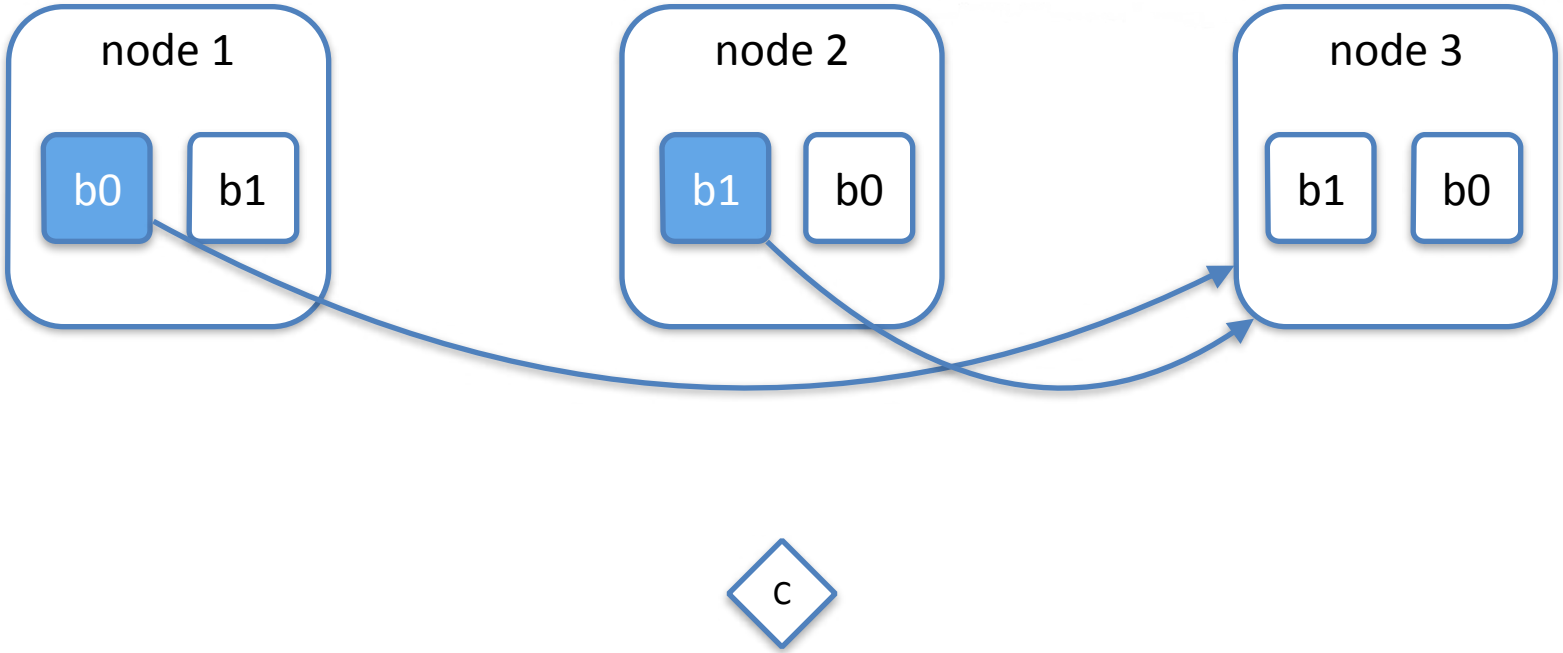
Searches: Collecting node

QUERY phase executed on each shard



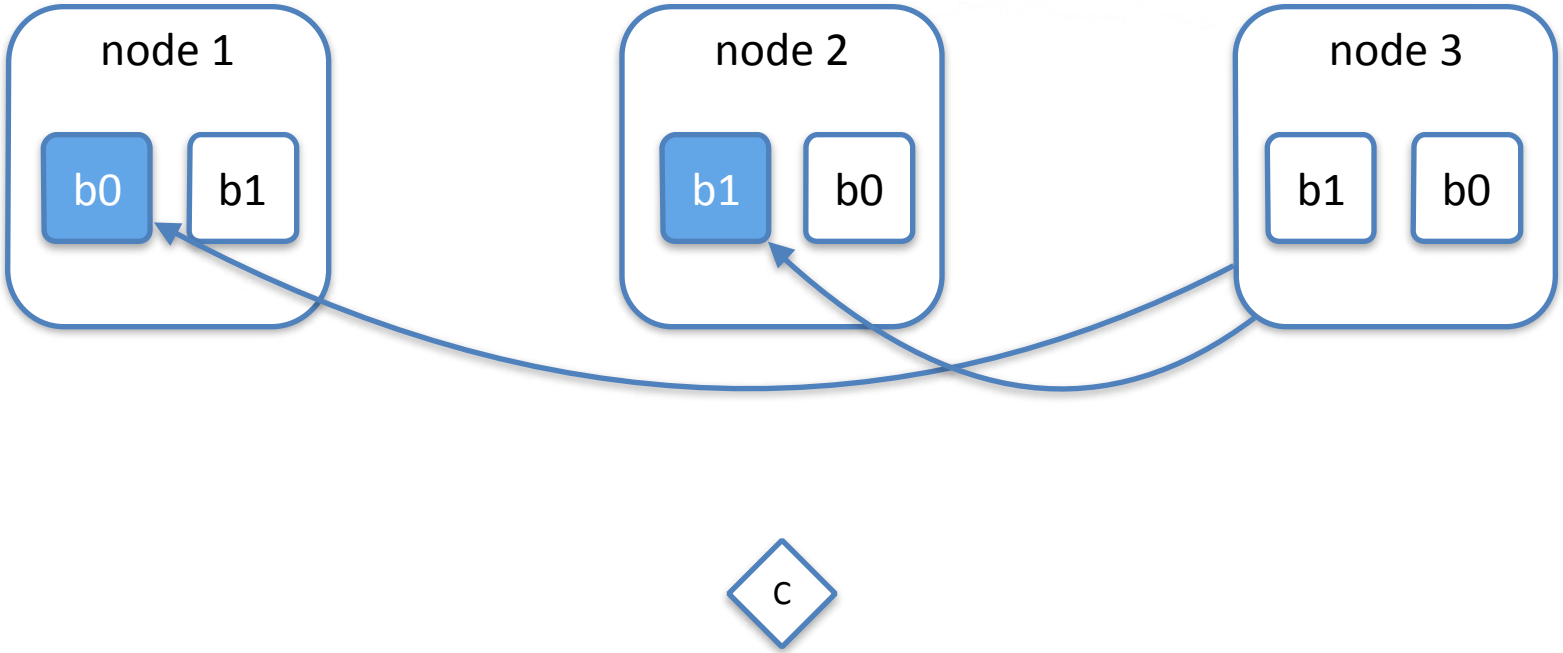
Searches: Collecting node

QUERY results are sorted



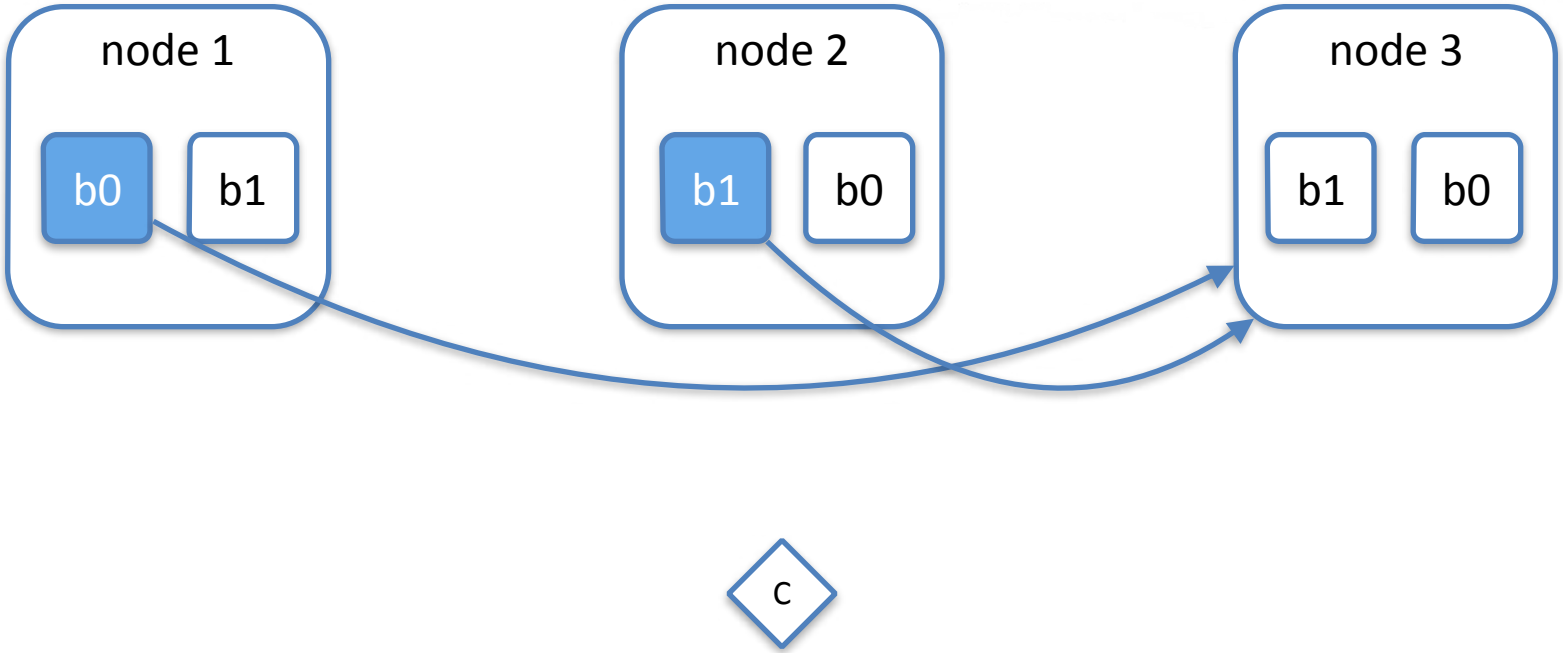
Searches: Collecting node

FETCH phase executed

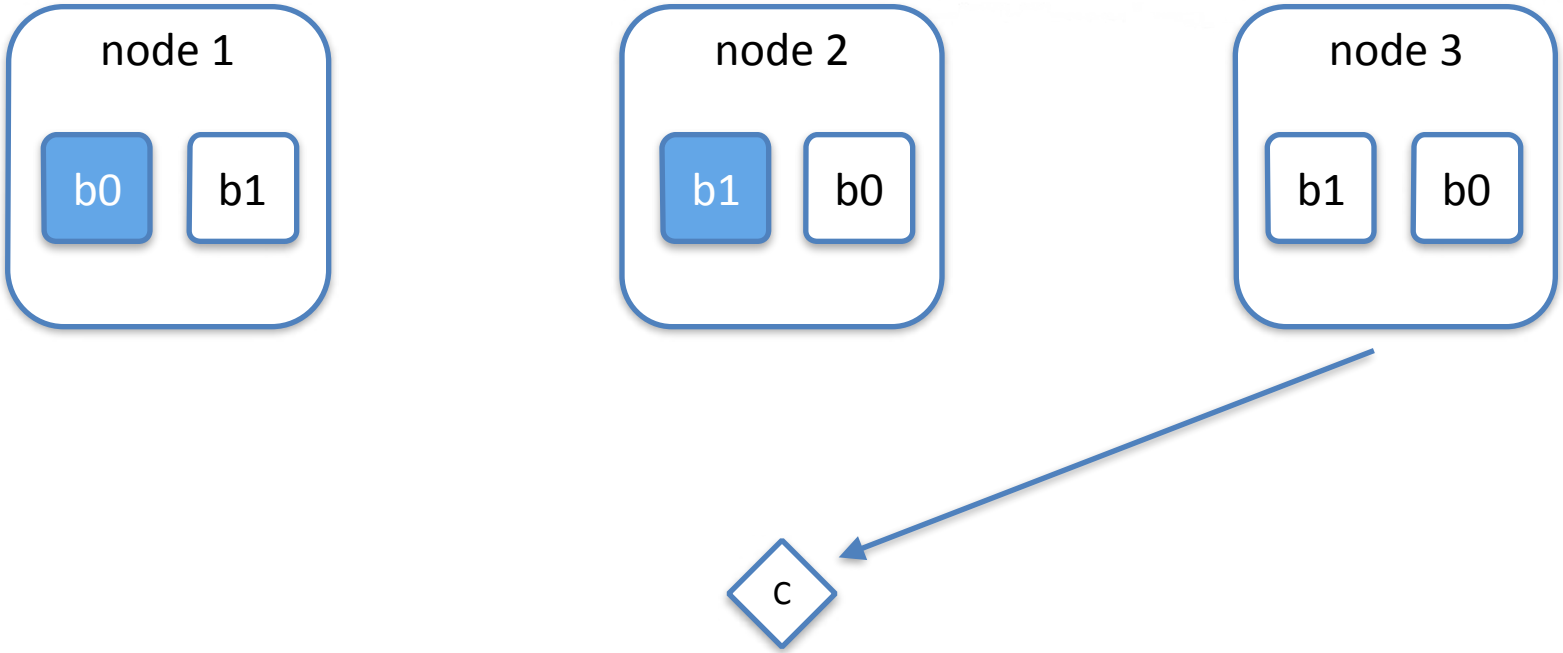


Searches: Collecting node

FETCH results are returned



Searches: Collecting node



Search types

- Full text search
  - Goes on disk, access segments, execute search
- Aggregation
  - Loads data into memory from segments, executes on in-memory data structure

Memory: Fielddata

```
GET /books/book/_search?search_type=count
{
  "aggs" : {
    "author-top10" : {
      "terms" : {
        "field" : “authors.raw"
      }
    }
  }
}
```

Memory: Fielddata

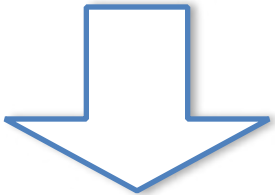
This does not work for aggs

Term	*Document ID*
clinton	***1***
gormley	***1***
tong	***1***
zachary	***1***

Memory: Fielddata

Uninverting the index from a raw field

Term	Document ID
Clinton Gormley	1
Zachary Tong	1



Document ID	Term
1	Clinton Gormley, Zachary Tong

Memory: Fielddata

- Used for sorting & aggregations
- All values of a field are loaded into the heap
- Per segment data structure

- Can take a lot of memory

Doc values

On-Disk vs. in-memory

```
PUT /books/book/_mapping
{
  "book" : {
    "properties" : {
      "authors": {
        "type": "string", "analyzer": "standard",
        "fields": { "raw": { "type": "string", "index": "not_analyzed",
"doc_values" : true } }
      }
    }
  }
}
```

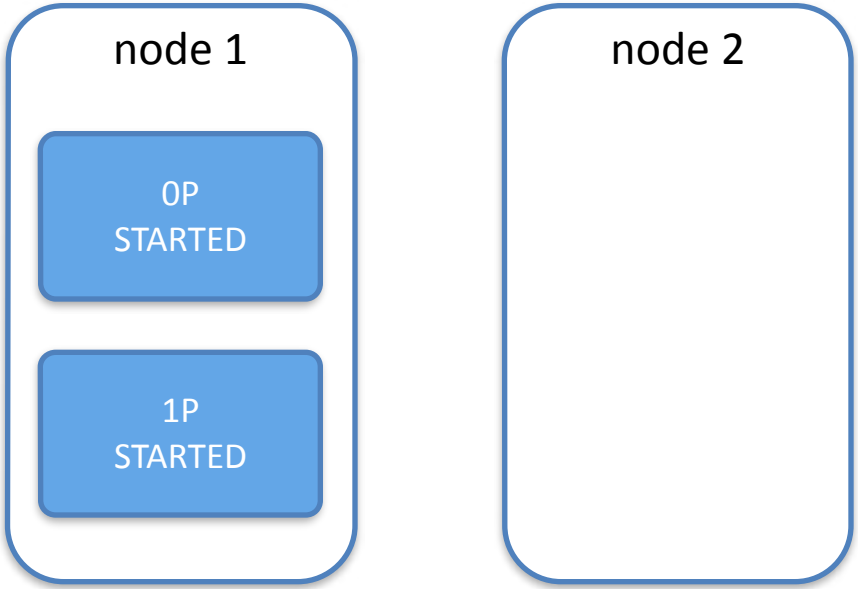
Where are we now?

- Original source is sent to collecting node during queries
- Document lives in a segment as part of a shard/index
- Document fields live in memory as part of fielddata
- Segments are put into the filesystem cache by the OS

Scaling your cluster

Relocation

Let's offload our fast node



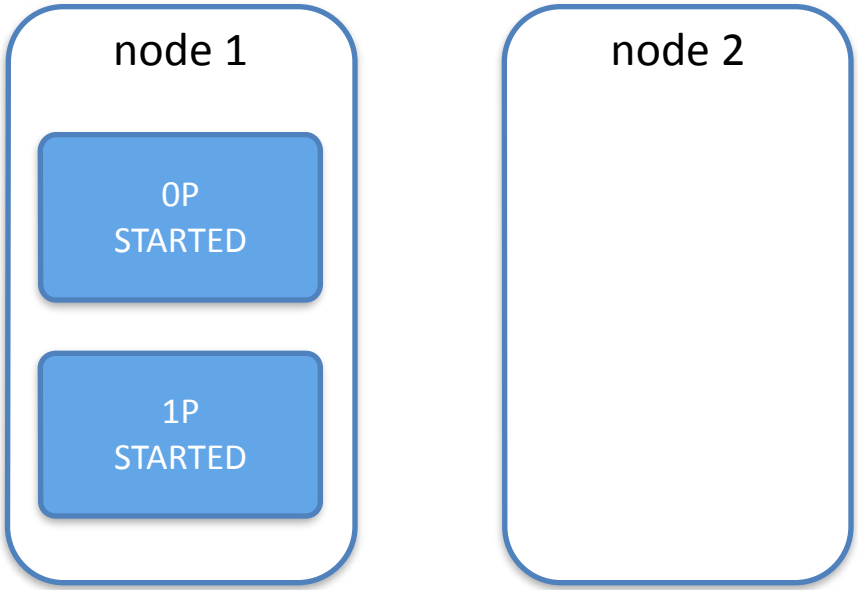
node.box_type: ssd

node.box_type: hdd

```
PUT /books/
{
  "settings" : {
    "index.routing.allocation.include.box_type" : "ssd"
  }
}
```

Relocation

Let's offload our fast node



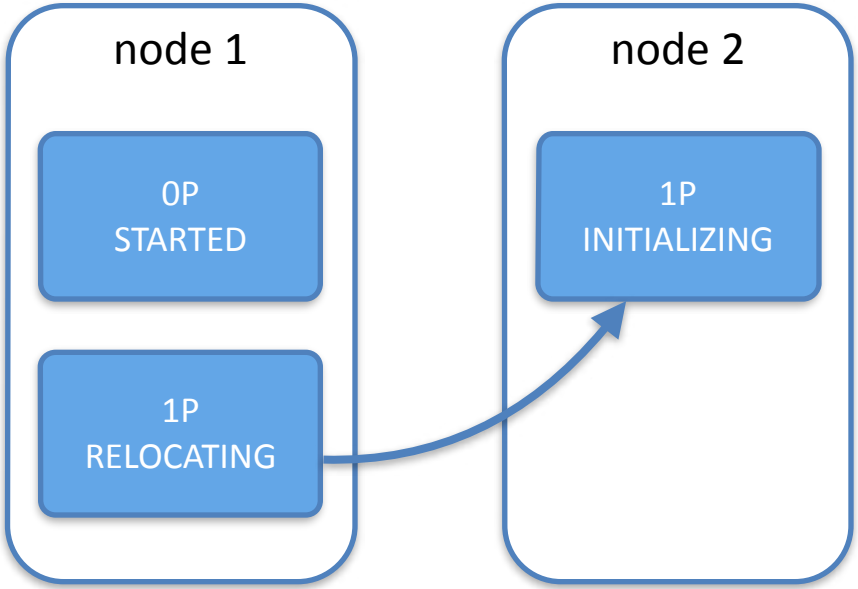
node.box_type: ssd

node.box_type: hdd

```
PUT /books/_settings
{
  "index.routing.allocation.include.box_type" : "hdd"
}
```

Relocation

Let's offload our fast node



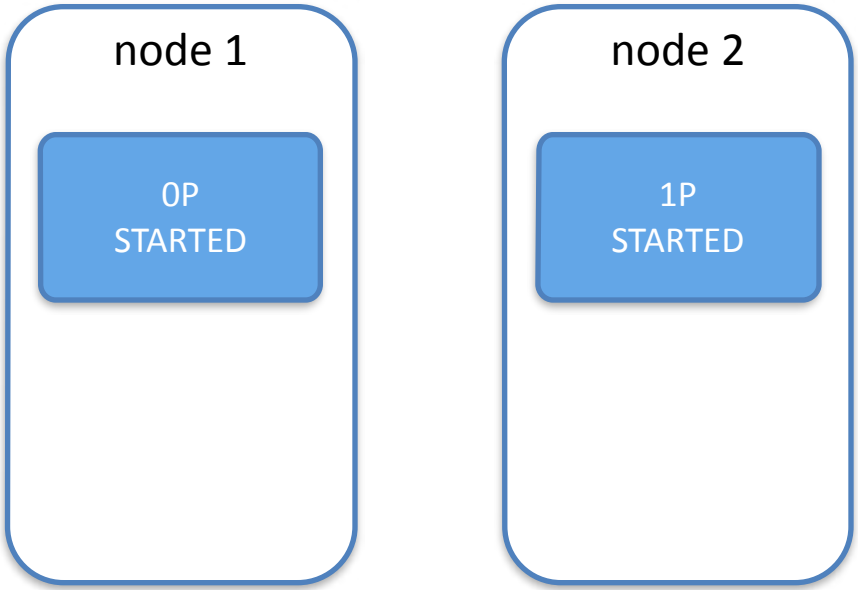
node.box_type: ssd

node.box_type: hdd

```
PUT /books/_settings
{
  "index.routing.allocation.include.box_type" : "hdd"
}
```

Relocation

Let's offload our fast node



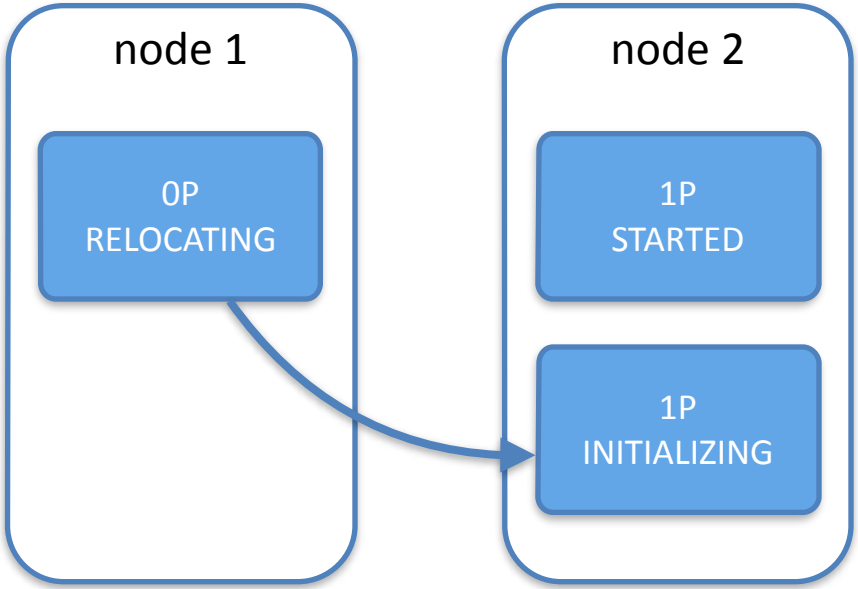
node.box_type: ssd

node.box_type: hdd

```
PUT /books/_settings
{
  "index.routing.allocation.include.box_type" : "hdd"
}
```

Relocation

Let's offload our fast node



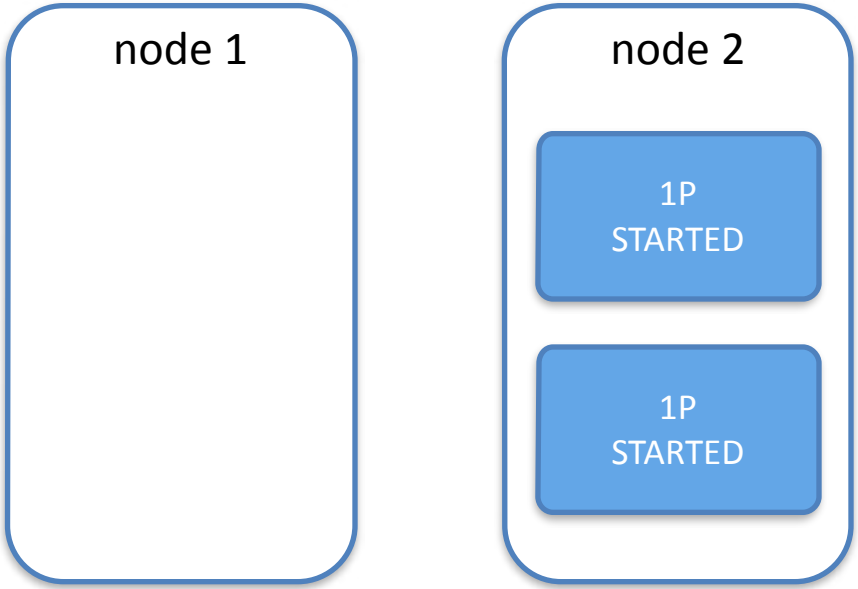
node.box_type: sdd

node.box_type: hdd

```
PUT /books/_settings
{
  "index.routing.allocation.include.box_type" : "hdd"
}
```

Relocation

Let's offload our fast node



node.box_type: ssd

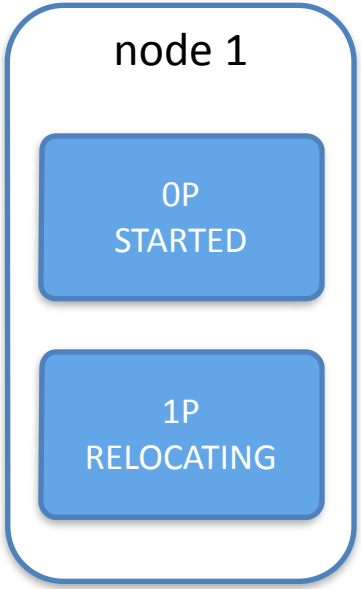
node.box_type: hdd

```
PUT /books/_settings
{
  "index.routing.allocation.include.box_type" : "hdd"
}
```

Relocation

What does RELOCATING mean?

- Point in time snapshot is copied (segments)
- Write operations occur and create new segments
- All write operations are stored in translog
- This translog is sent to the other node and replayed



Relocation

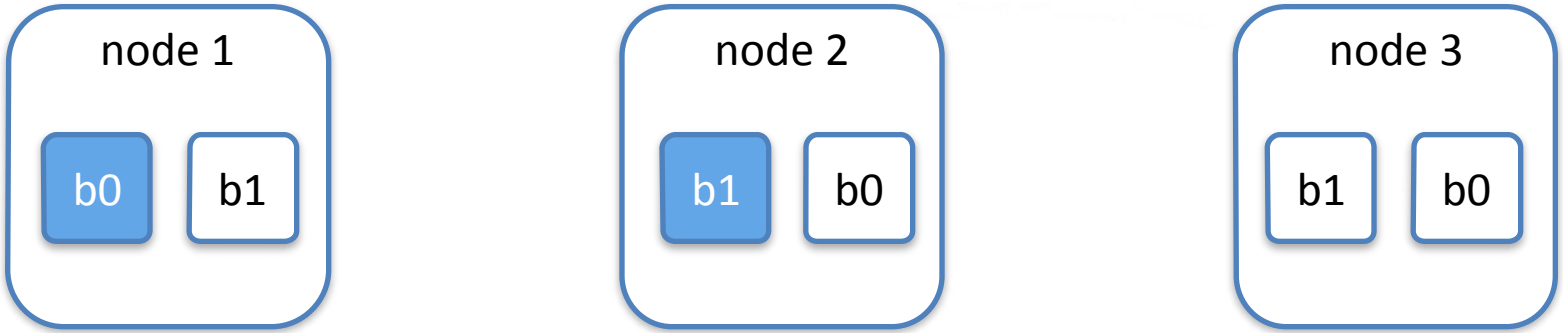
- Process of copying whole shards across the cluster
- Triggered by
  - adding/removing nodes from the cluster
  - exceeded disk thresholds
  - shard allocation filtering

Where are we now?

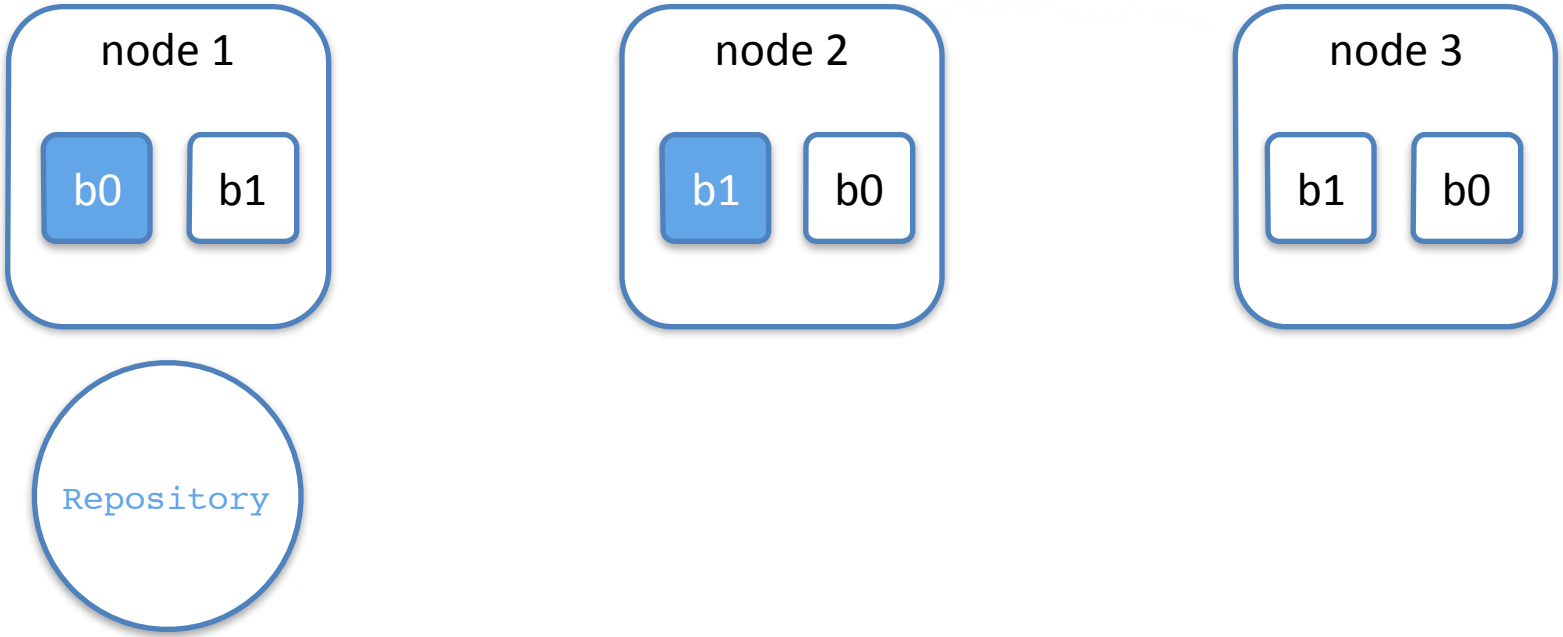
- Relocation moves shards across the cluster
- Document is duplicated by being copied from one node to another, as part of a segment
- Segments are kept open until operation is complete

Backup your data

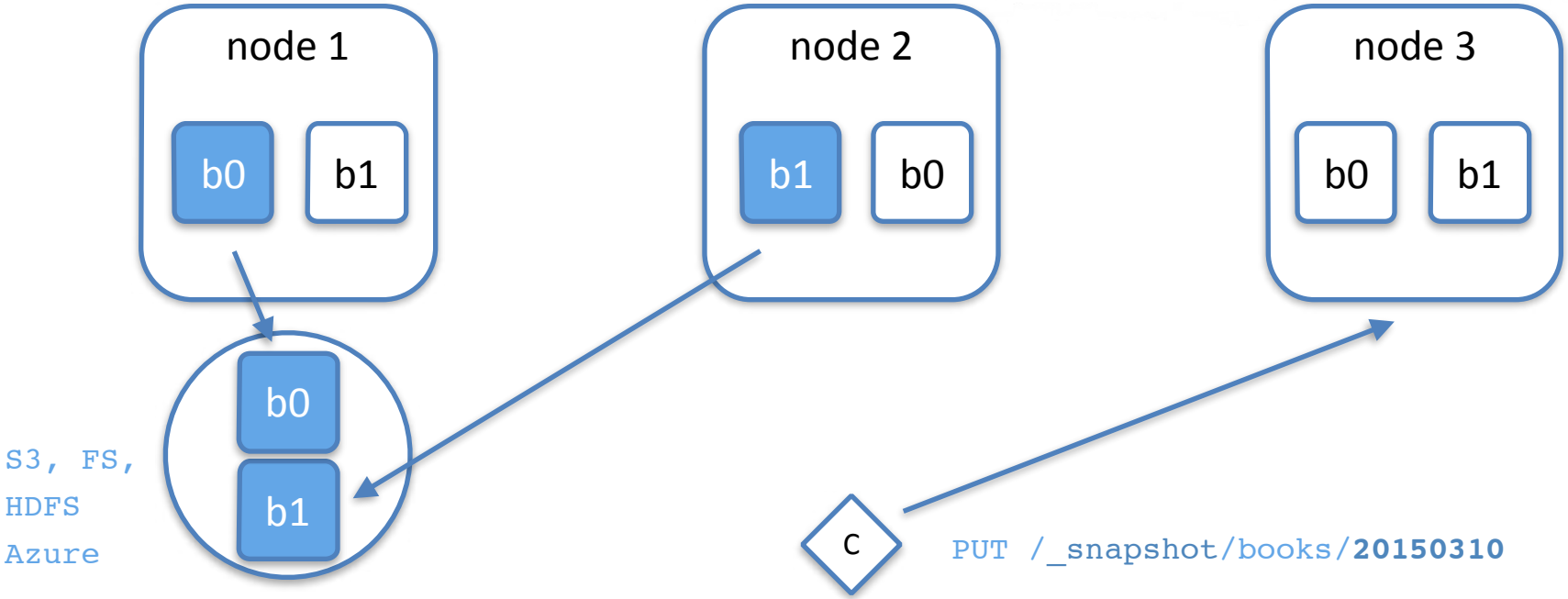
How to backup?



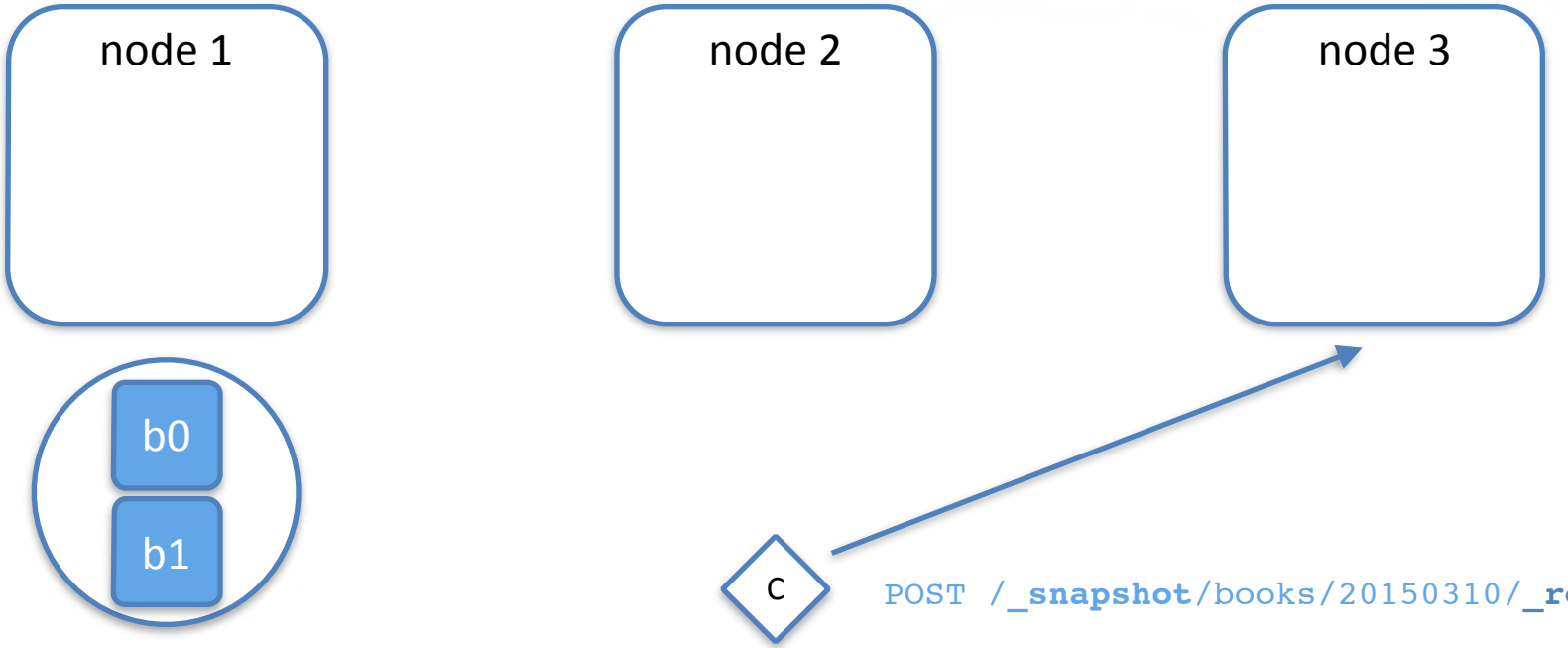
How to backup?



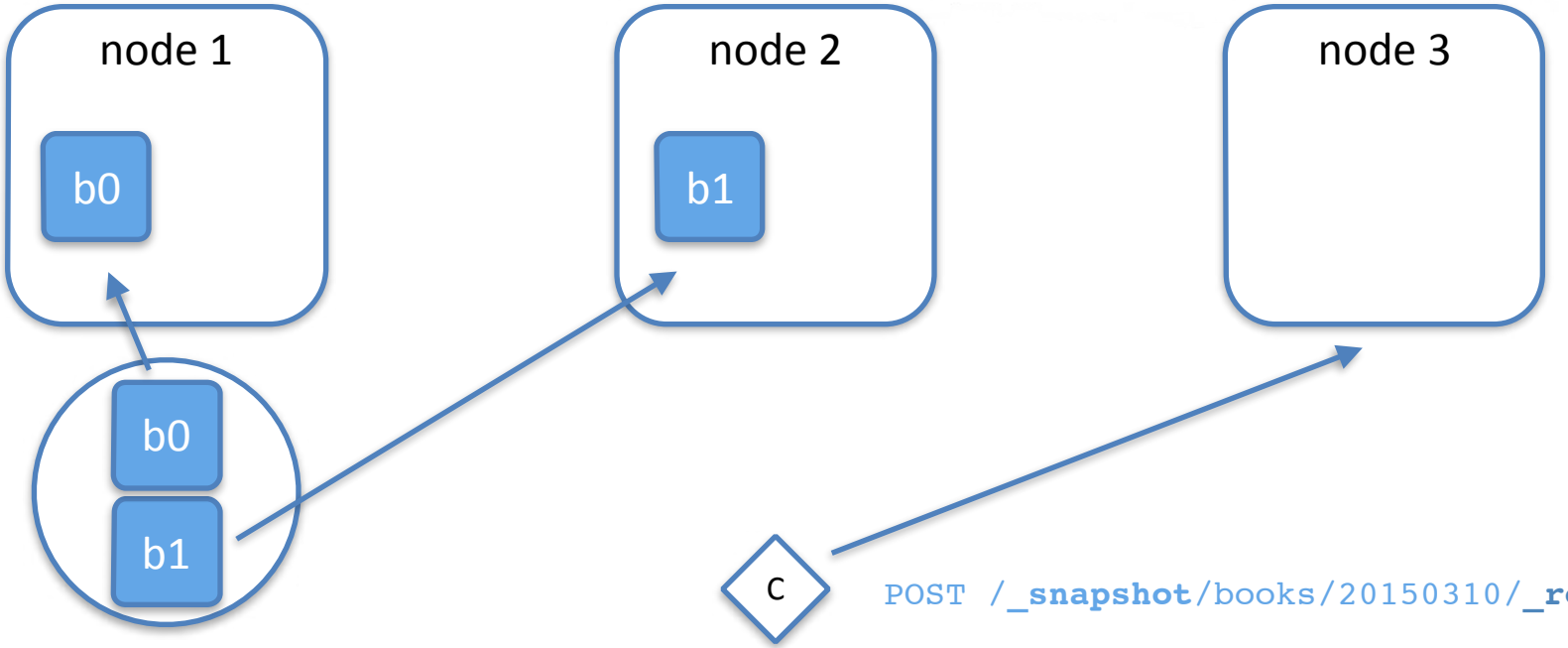
How to backup?



Restore from a snapshot

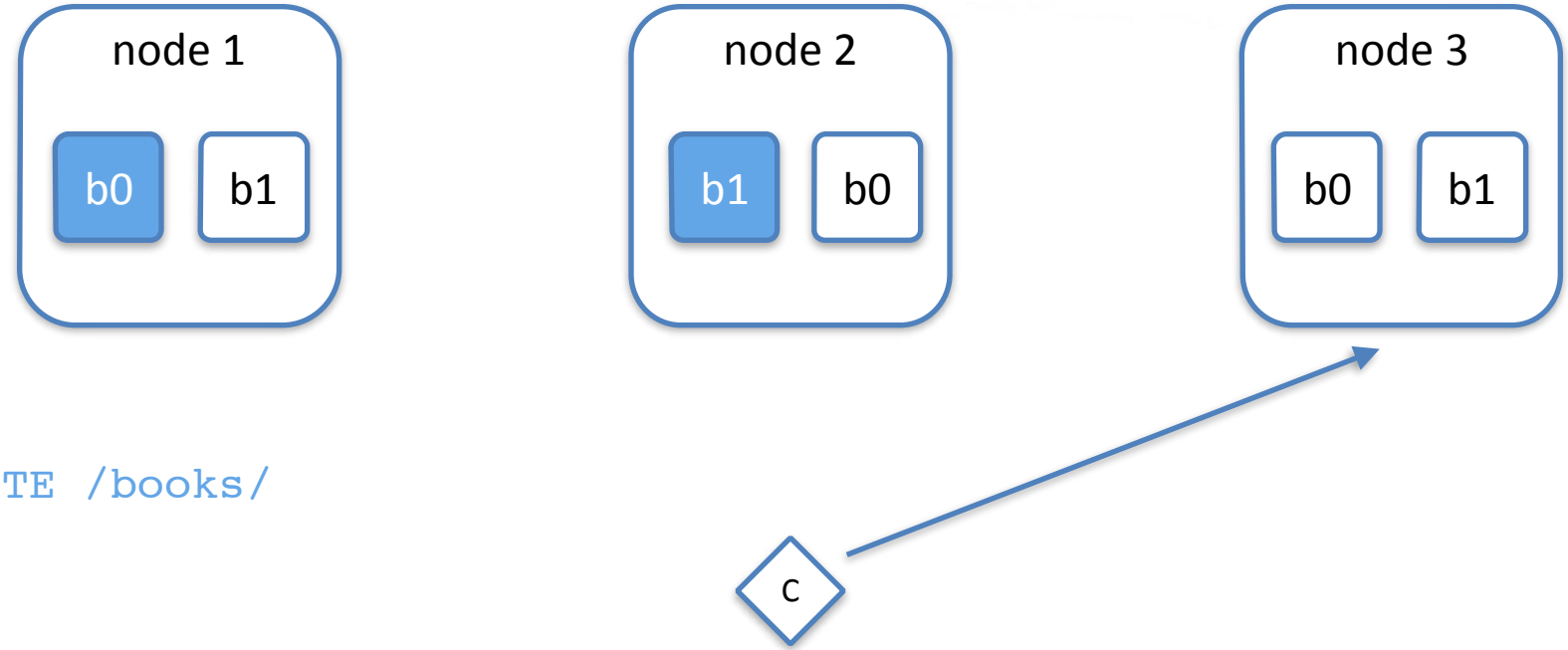


Restore from a snapshot



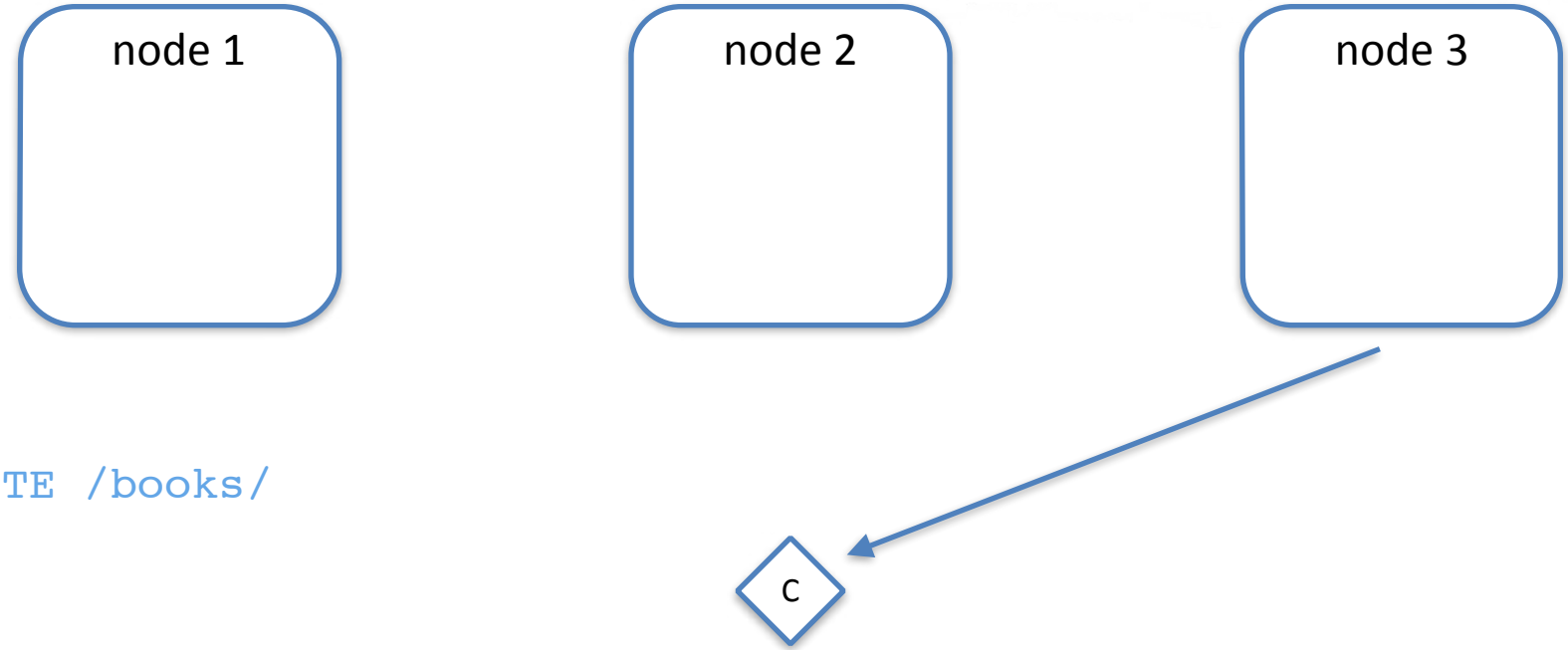
Deleting data

Index deletion



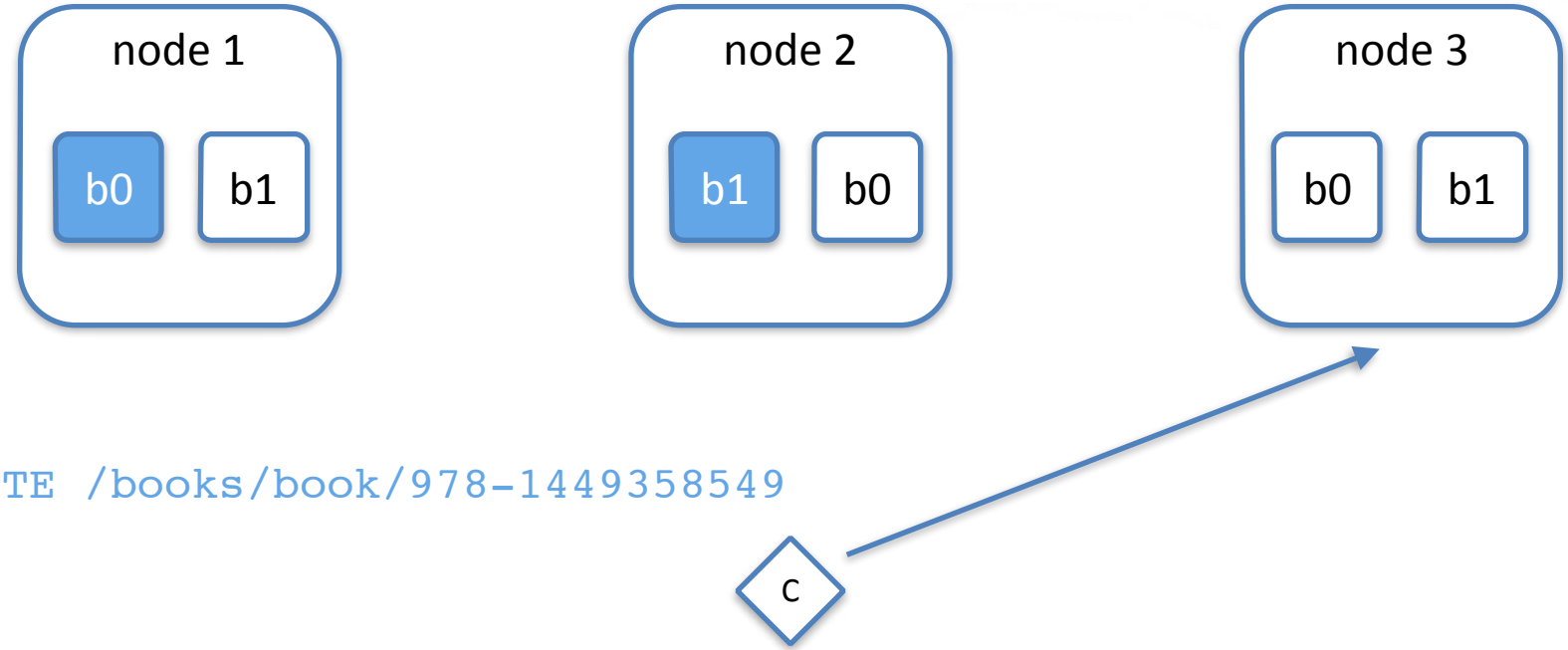

```
DELETE /books/
```

Index deletion




```
DELETE /books/
```

Document deletion

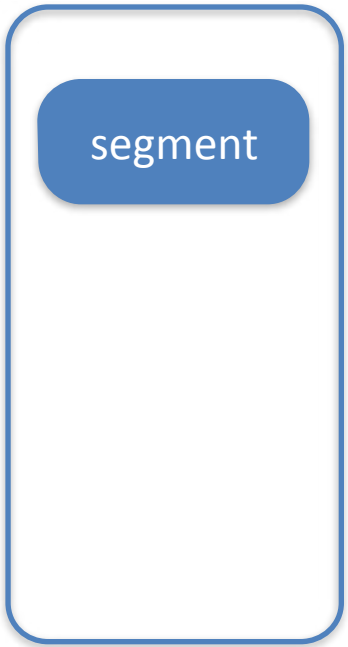


Document deletion

- Segments are immutable

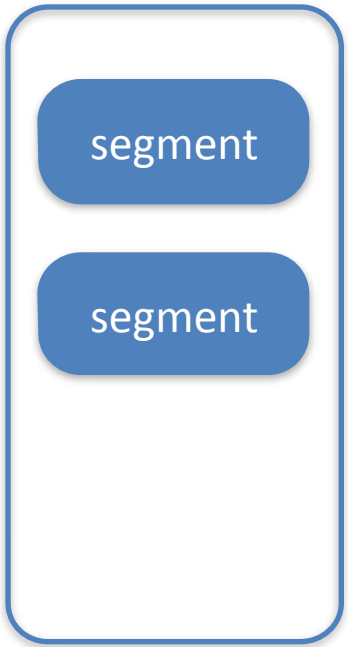
- Deletes are soft, document ids are added in a tombstone file to be marked for deletion
- Actual deletion happens on next merge of that segment

Lucene merges

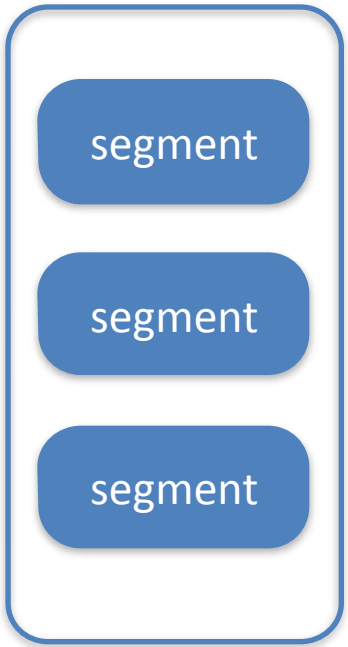


- Segments are created all the time when indexing data
- Merging creates one big segment from several smaller ones

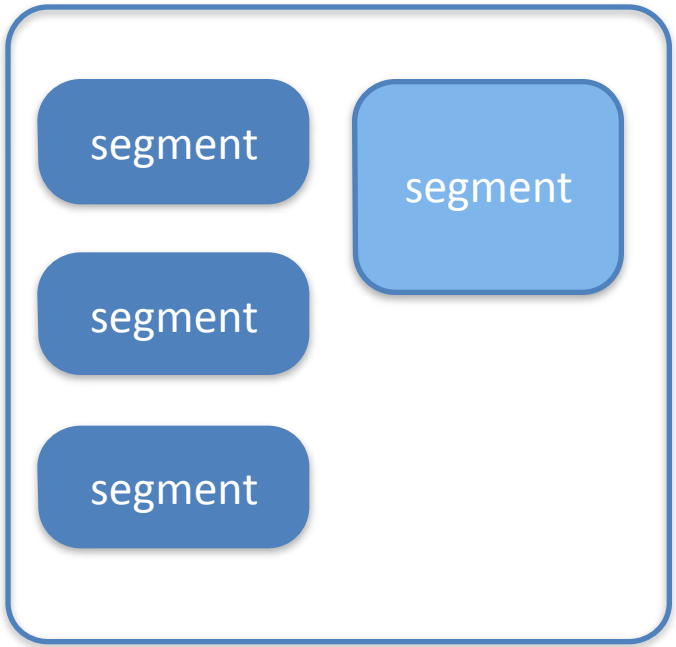
Lucene merges



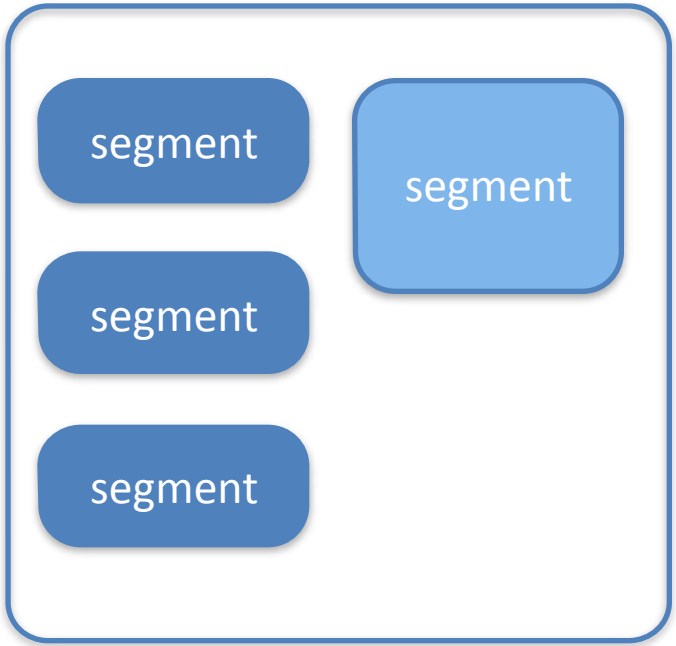
Lucene merges



Lucene merges

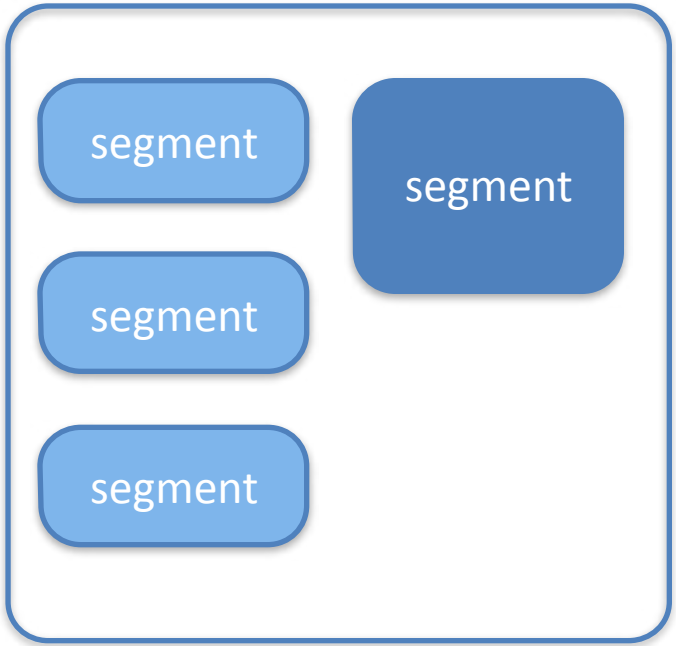


Lucene merges

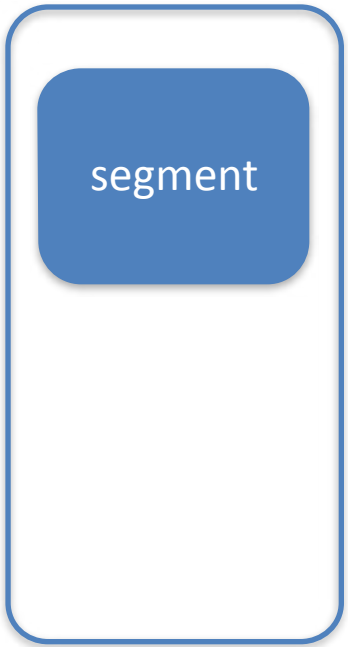


document exists twice!

Lucene merges



Lucene merges



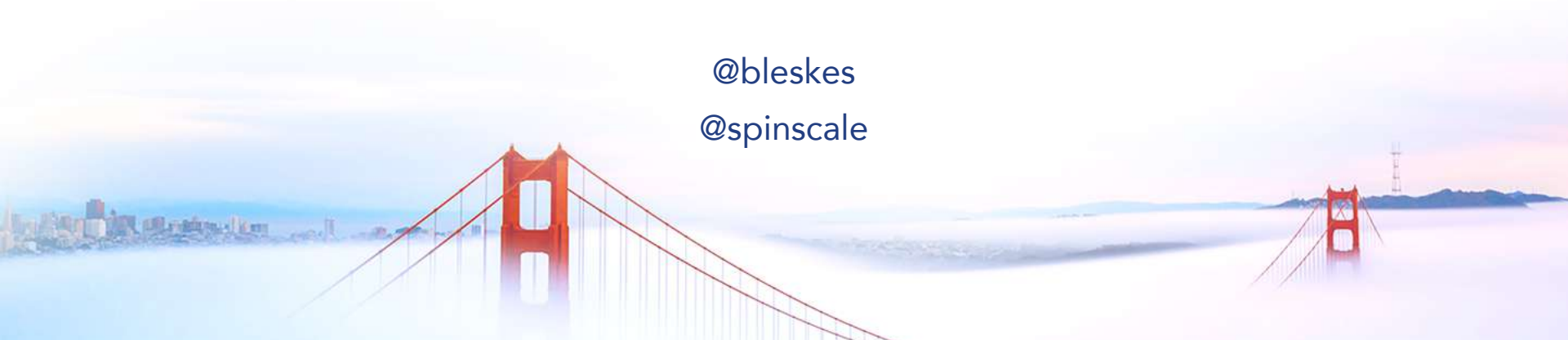
Don’t be afraid of data duplication!

Don’t be afraid of data duplication!

... if you are aware of its lifecycle!

elastic{ON}15

Thanks for listening!
Questions?



@bleskes

@spinscale