Solr 4
The NoSQL Database

Yonik Seeley

Apachecon Europe 2012

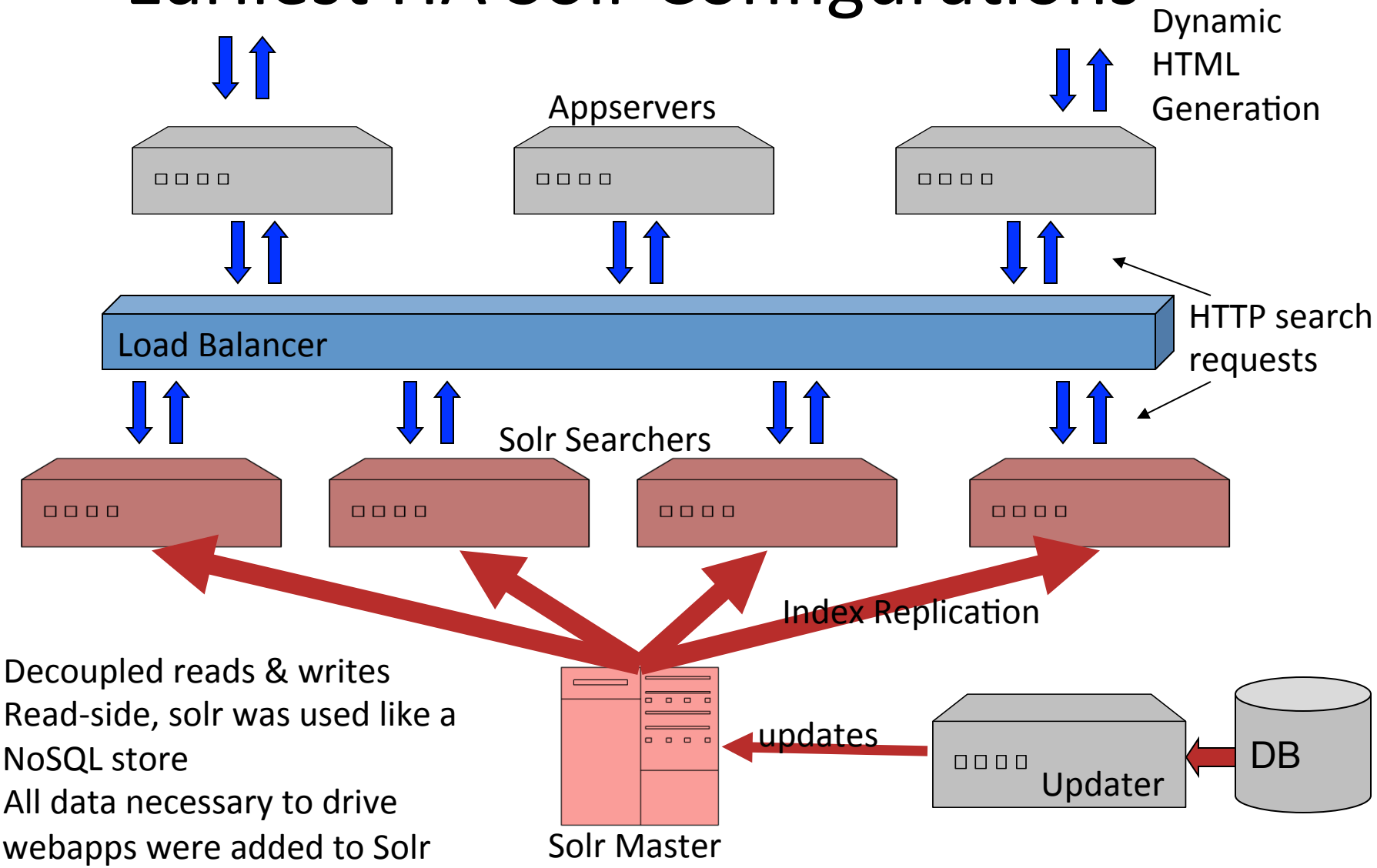
My Background

- Creator of Solr
- Co-founder of LucidWorks
- Expertise: Distributed Search systems and performance
- Lucene/Solr committer, PMC member
- Work: CNET Networks, BEA, Telcordia, among others
- M.S. in Computer Science, Stanford

What is NoSQL

- Non-traditional data stores
- Doesn’t use / isn’t designed around SQL
- May not give full ACID guarantees
  - Offers other advantages such as greater scalability as a tradeoff
- Distributed, fault-tolerant architecture

Earliest HA Solr Configurations



- Decoupled reads & writes
- Read-side, solr was used like a NoSQL store
- All data necessary to drive webapps were added to Solr documents
- Database remained system of record for updaters

Desired Features

(that we started thinking about back in 08)

- Automatic Distributed Indexing
- HA for Writes
- Durable Writes
- Near Real-time Search
- Real-time get
- Optimistic Concurrency

Solr Cloud

- Distributed Indexing designed from the ground up around desired features, not CAP theorem.
  - Note: you must handle P – the real choice is C or A
- Ended up with a CP system (roughly)
  - Eventual consistency is incompatible with optimistic concurrency
- We still do well with **A**vailability
  - All N replicas of a shard must go down before we lose writability for that shard
  - For a network partition, the “big” partition remains active (i.e. Availability isn’t “on” or “off”)
- More: Mark Miller’s SolrCloud Architecture talk tomorrow at 14:45

Solr 4

New Top Level View

- Document Oriented NoSQL Search Platform
  - Data-format agnostic (JSON, XML, CSV, binary)
- Distributed
- Fault Tolerant
  - HA + No single points of failure
- Atomic Updates
- Optimistic Concurrency
- Full-Text search + Hit Highlighting
- Tons of specialized queries: Faceted search, grouping, pseudo-Join, spatial search, functions

Quick Start

1. Unzip the binary distribution (.ZIP file)

   Note: no “installation” required

2. Start Solr

```
$ cd example
$ java –jar start.jar
```

3. Go!

   Browse to http://localhost:8983/solr for the new admin interface

- Dashboard
- Logging
- Core Admin
- Java Properties
- Thread Dump

collection1

Instance

Start	5 minutes ago
Host	192.168.1.102
CWD	/opt/code/lusolr/solr/example
Instance	/opt/code/lusolr/solr/example/solr/collection1
Data	/opt/code/lusolr/solr/example/solr/collection1/data
Index	/opt/code/lusolr/solr/example/solr/collection1/data/index

Versions

solr-spec	5.0.0.2012.08.15.13.17.06
solr-impl	5.0-SNAPSHOT 1373442M - yonik - 2012-08-15 13:17:06
lucene-spec	5.0-SNAPSHOT
lucene-impl	5.0-SNAPSHOT 1373442 - yonik - 2012-08-15 13:15:15

JVM

Runtime	Java HotSpot(TM) 64-Bit Server VM (20.8-b03-424)
Processors	4

System

Physical Memory 73.0%

5.84 GB

8.00 GB

Swap Space 72.4%

2.17 GB

3.00 GB

File Descriptor Count 1.5%

153

10240

JVM-Memory 14.7%

18.24 MB

81.06 MB

123.94 MB

Documentation Issue Tracker IRC Channel Community forum Solr Query Syntax

Add and Retrieve document

```
$ curl http://localhost:8983/solr/update -H 'Content-type:application/json' -d '
[
 { "id" : "book1",
   "title" : "American Gods",
   "author" : "Neil Gaiman"
 }
]'
```

```
$ curl http://localhost:8983/solr/get?id=book1
{
  "doc": {
    "id" : "book1",
    "author": "Neil Gaiman",
    "title" : "American Gods",
    "_version_": 1410390803582287872
  }
}
```

Note: no type of “commit” is necessary to retrieve documents via /get (real-time get)

Atomic Updates

```
$ curl http://localhost:8983/solr/update -H 'Content-type:application/json' -d '
[
 {"id"         : "book1",
  "pubyear_i" : { "add" : 2001 },
  "ISBN_s"    : { "add" : "0-380-97365-1"}
 }
]'
```

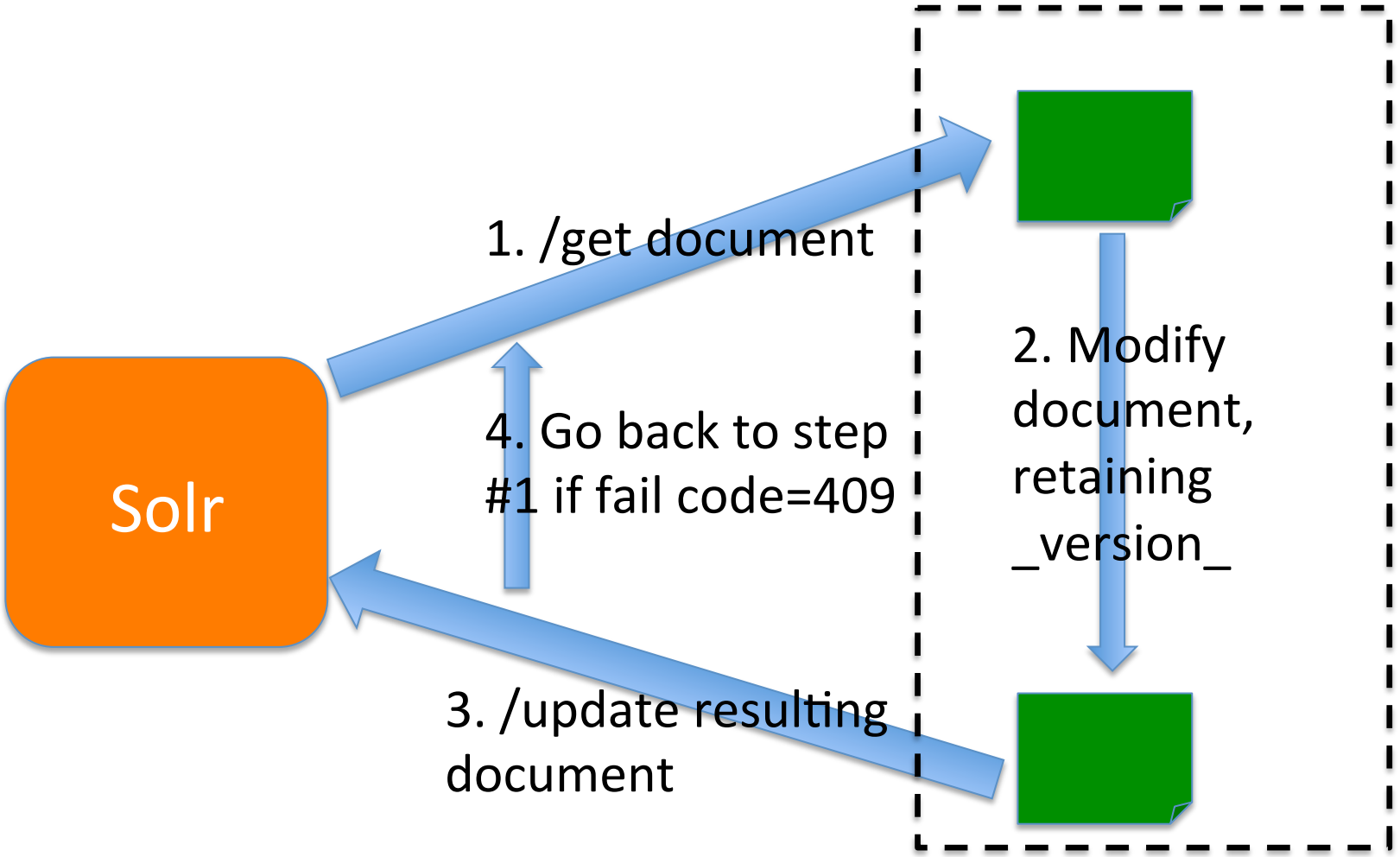
```
$ curl http://localhost:8983/solr/update -H 'Content-type:application/json' -d '
[
 {"id"         : "book1",
  "copies_i"  : { "inc" : 1},
  "cat"       : { "add" : "fantasy"},
  "ISBN_s"    : { "set" : "0-380-97365-0"}
  "remove_s"  : { "set" : null } }
]'
```

Schema-less?

- Schema-less: No such thing for anything based on Lucene
  - Fields with same name must be indexed the same way
- Closest approximation: guess the type when you see a new field
  - Can be troublesome (adding 0 then 1.5)
  - Most clients still need to know the "schema"
- Dynamic fields: convention over configuration
  - Only pre-define types of fields, not fields themselves
  - No guessing. Any field name ending in **`_i`** is an integer

Optimistic Concurrency

- Conditional update based on document version



Version semantics

- Specifying _version_ on any update invokes optimistic concurrency

version	Update Semantics
> 1	Document version must exactly match supplied _version_
1	Document must exist
< 0	Document must not exist
0	Don't care (normal overwrite if exists)

Optimistic Concurrency Example

```
$ curl http://localhost:8983/solr/get?id=book2
{ "doc” : {
    "id":"book2",
    "title":["Neuromancer"],
    "author":"William Gibson",
    "copiesIn_i":7,
    "copiesOut_i":3,
    "_version_":123456789 }}
```

Get the document

```
$ curl http://localhost:8983/solr/update -H 'Content-type:application/json' -d '
[  {
    "id":"book2",
    "title":["Neuromancer"],
    "author":"William Gibson",
    "copiesIn_i":6,
    "copiesOut_i":4,
    "_version_":123456789 }
]'
```

Modify and resubmit, using the same _version_

Alternately, specify the _version_ as a request parameter

```
curl http://localhost:8983/solr/update?_version_=123456789 -H 'Content-type:application/json' -d
[…]
```

Optimistic Concurrency Errors

- HTTP Code 409 (Conflict) returned on version mismatch

$ curl **-i** http://localhost:8983/solr/update -H 'Content-type:application/json' -d '
[{"id":"book1", "author":"Mr Bean", "_version_":54321}]'

```
HTTP/1.1 409 Conflict
Content-Type: text/plain;charset=UTF-8
Transfer-Encoding: chunked

{
  "responseHeader":{
    "status":409,
    "QTime":1},
  "error":{
    "msg":"version conflict for book1 expected=12345
           actual=1408814192853516288",
    "code":409}}
```

Simplified JSON Delete Syntax

- Singe delete-by-id

```
{"delete":”book1"}
```

- Multiple delete-by-id

```
{"delete":[”book1”,”book2”,”book3”]}
```

- Delete with optimistic concurrency

```
{"delete":{"id":”book1", "_version_":123456789}}
```

- Delete by Query

```
{"delete":{”query":”tag:category1”}}
```

Durable Writes

- Lucene flushes writes to disk on a “commit”
- Solr 4 maintains it’s own transaction log
  - Services real-time get requests
  - Recover from crash (log replay on restart)
  - Supports distributed “peer sync”
- Writes forwarded to multiple shard replicas
  - A replica can go away forever w/o collection data loss
  - A replica can do a fast “peer sync” if it’s only slightly out of data
  - A replica can do a full index replication (copy) from a peer

Near Real Time (NRT) softCommit

- softCommit opens a new view of the index without flushing + fsyncing files to disk
  - Decouples update visibility from update durability
- commitWithin now implies a soft commit
- Current autoCommit defaults from solrconfig.xml:

```
 <autoCommit>
       <maxTime>15000</maxTime>
       <openSearcher>false</openSearcher>
 </autoCommit>

<!--  <autoSoftCommit>
         <maxTime>5000</maxTime>
       </autoSoftCommit> -->
```

New Queries in Solr 4

New Spatial Support

- Multiple values per field
- Index shapes other than points (circles, polygons, etc)
- Well Known Text (WKT) support via JTS
- Indexing:

  "geo”:”43.17614,-90.57341”

  “geo”:”Circle(4.56,1.23 d=0.0710)”

  “geo”:”POLYGON((-10 30, -40 40, -10 -20, 40 20, 0 0, -10 30))”

- Searching:

  fq=geo:"Intersects(-74.093 41.042 -69.347 44.558)"

  fq=geo:"Intersects(POLYGON((-10 30, -40 40, -10 -20, 40 20, 0 0, -10 30))) "

Relevancy Function Queries

function	semantics	example
docfreq(field,term)	Constant: number of documents containing the term	docfreq(text,'solr')
termfreq(field,term)	Number of times term appears in a doc	termfreq(text,'solr')
totaltermfreq(field,term)	Constant: number of times the term appears in the field over the whole index	ttf(text,'solr')
sumtotaltermfreq(field)	Constant: sum of ttf of every term (i.e. number of indexed tokens in field)	sttf(text,'solr')
idf(field,term)	Constant: inverse document frequency using the Similarity for the field	idf(text,'solr')
tf(field,term)	term frequency scoring factor using the Similarity for the field	tf(text,'solr')
norm(field)	Length normalization for the document (including any index-time boost)	norm(text)
maxdoc()	Constant: number of documents	maxdoc()
numdocs()	Constant: number of non-deleted documents	numdocs()

Boolean/Conditional Functions

- Constants `true`, `false`
- exists(field|function) returns true if a value exists for a document
- if(expression,trueValue,falseValue)

  if(exists(myField),100,add(f2,f3))

  – Note: for numerics 0==false, for strings empty==false
- def(field|function,defaultValue)

  def(rating,5) // returns the rating, or 5 if this doc has none
- Other boolean functions: not(x), and(x,y), or(x,y), xor(x,y)

Pseudo-Fields

Returns other info along with document stored fields

- Function queries

```
fl=name,location,geodist(),add(myfield,10)
```

- Fieldname globs

```
fl=id,attr_*
```

- Multiple “fl” (field list) values

```
&fl=id,attr_*
&fl=geodist()
&fl=termfreq(text,’solr’)
```

- Aliasing

```
fl=id,location:loc,_dist_:geodist()
```

- fl=id,[explain],[shard] TODO: [hl] for highlighting

Pseudo-Fields Example

```
$ curl http://localhost:8983/solr/query?
   q=solr
   &fl=id,apache_mentions:termfreq(text,’apache’)
   &fl=my_constant:”this is cool!”
   &fl=inStock, not(inStock)
   &fl=other_query_score:query($qq)
   &qq=text:search

{ "response":{"numFound":1,"start":0,"docs":[
      {
        "id":"SOLR1000",
        "apache_mentions":1,
        "my_constant":"this is cool!",
        "inStock":true,
        "not(inStock)":false,
        "other_query_score":0.84178084
      }]}
```

Pseudo-Join

id: blog1
name: Solr 'n Stuff
owner: Yonik Seeley
Started: 2007-10-26

id: **blog2**
name: lifehacker
owner: Gawker Media
started: 2005-1-31

id: post1
blog_id: blog1
author: Yonik Seeley
title: Solr relevancy function queries
body: Lucene's default ranking […]

id: post2
blog_id: blog1
author: Yonik Seeley
title: Solr result grouping
body: Result Grouping, also called […]

id: post3
blog_id: **blog2**
author: Whitson Gordon
title: How to Install **Netflix** on Almost Any Android Device

Restrict to blogs mentioning netflix:

fq={!**join** from=**blog_id** to=**id**}body:netflix

- How it works:
  - Finds all documents matching "netflix"
  - Maps to different docs by following **blog_id** to **id**

Pseudo-Join Examples

- Only show posts from blogs started after 2010

  &fq={!join from=id to=blog_id}started:[2010 TO *]

- If any post in a blog mentions “embassy”, then search all posts in that blog for “bomb” (self-join)

  q=bomb

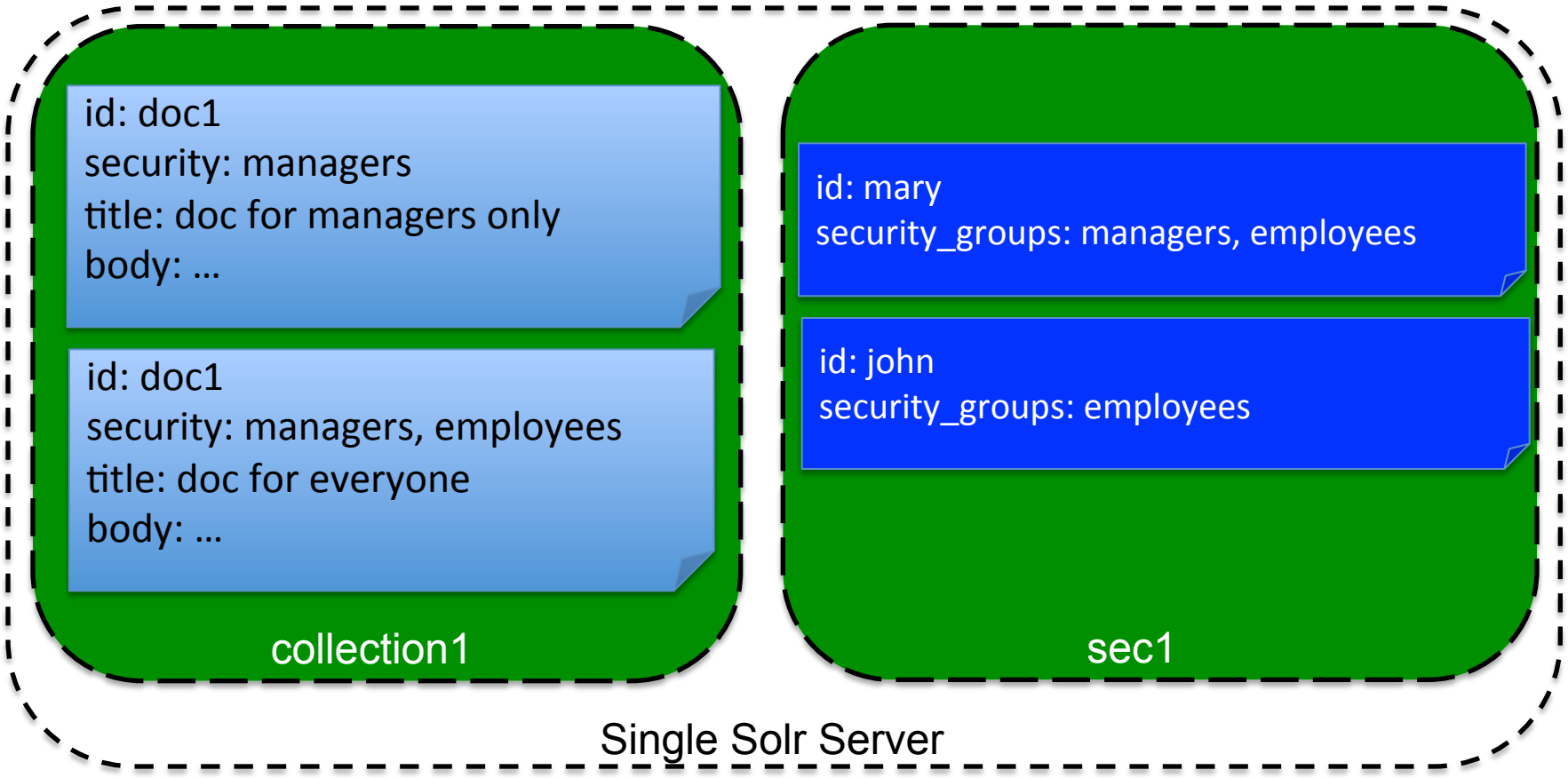
  &fq={!join from=blog_id to=blog_id}embassy

- If any blog post mentions “embassy”, then search all emails with the same blog owner for “bomb”

  q=email_body:bomb

  &fq={!join from=owner_email_user to=email_user}{!join from=blog_id to=id}embassy

Cross-Core Join



http://localhost:8983/solr/collection1/select?q=foo
&fq={!join **fromIndex**=sec1 from=security_groups to=security}user:john

Pseudo-Join vs Grouping

Pseudo-Join	Result Grouping / Field Collapsing
O(n_terms_in_join_fields)	O(n_docs_in_result)
Single or multi-valued fields	Single-valued fields only
Filters only (no info currently passed from the “from” docs to the “to” docs).	Can order docs within a group and groups by top doc within that group using normal sort criteria.
Chainable (one join can be the input to another)	Not currently chainable – can only group one field deep
Affects which documents match a request, so naturally affects facet numbers (e.g. you can search posts and get numbers of blogs)	New for 4.0: via optional post-group faceting, grouping can affect faceting by optionally using only the most relevant document per group.

Pivot Faceting

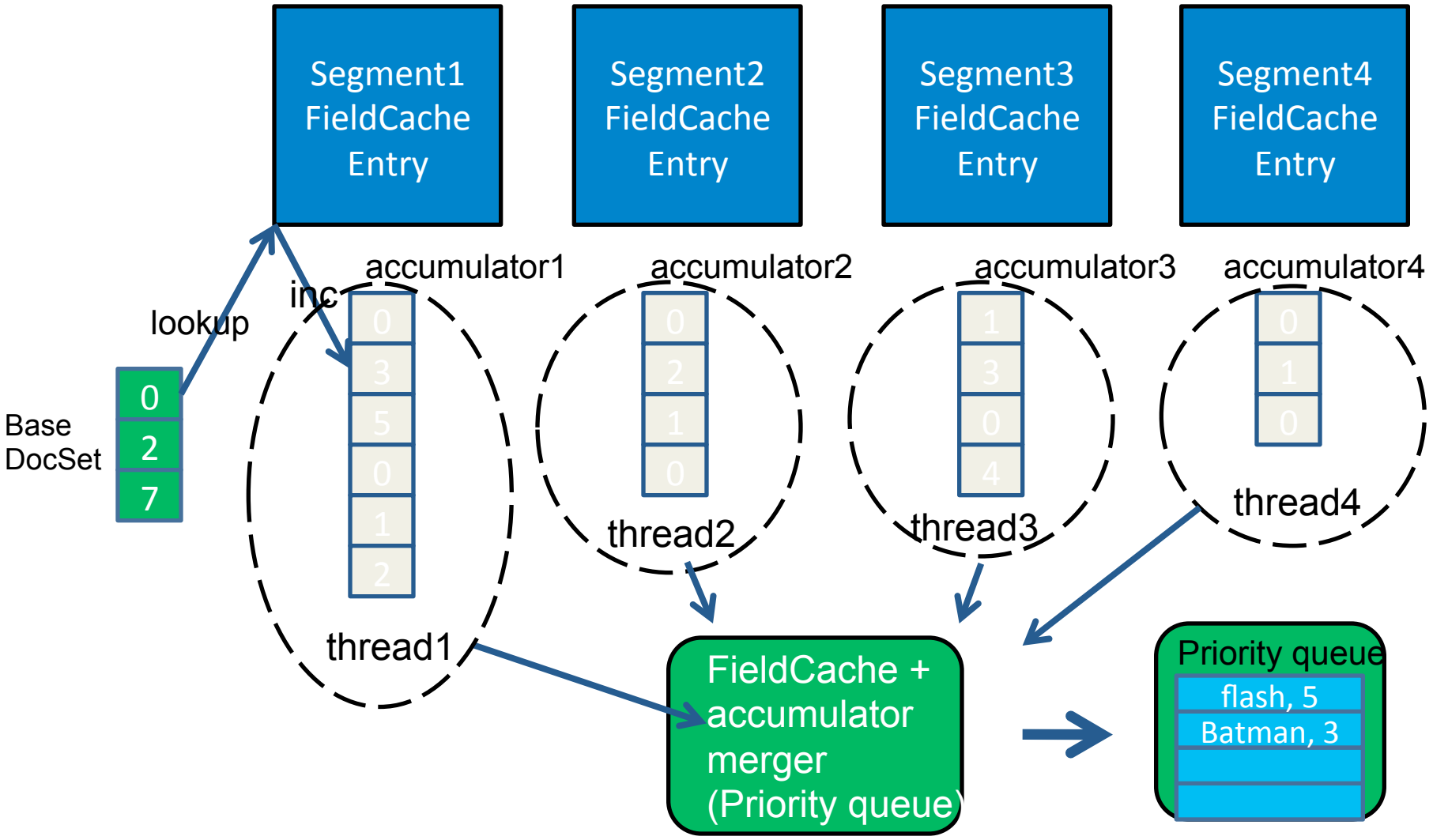
- Finds the top N constraints for field1, then for each of those, finds the top N constraints for field2, etc
- Syntax: facet.pivot=field1,field2,field3,…

```
facet.pivot=cat,inStock
```

	#docs	#docs w/ inStock:true	#docs w/ instock:false
cat:electronics	14	10	4
cat:memory	3	3	0
cat:connector	2	0	2
cat:graphics card	2	0	2
cat:hard drive	2	2	0

Per-segment single-valued faceting

`facet.method=fcs`



facet.method=fcs

- Controllable multi-threading

  ```
  facet.method=fcs
  facet.field={!threads=4}myfield
  ```

- Disadvantages
  - Slower unless committing multiple times per second (aggressive NRT)
  - Extra FieldCache merge step needed - O(nTerms)
  - Larger memory use (FieldCaches + accumulators)
- Advantages
  - Rebuilds FieldCache entries only for new segments (NRT friendly)
  - Multi-threaded

Questions?