

APACHE  
LUCENE  
EUROCON



# The Many Facets of Apache Solr

Yonik Seeley, Lucid Imagination  
yonik@lucidimagination.com, Oct 20 2011

Presented by

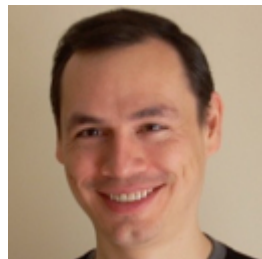
**lucid**  
IMAGINATION



# What I Will Cover

- What is Faceted Search
- Solr's Faceted Search
- Tips & Tricks
- Performance & Algorithms

# My Background



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

# What is Faceted Search

# Faceted Search Example

## Digital cameras

Manufacturer is a **facet**, a way of categorizing the results

The **facet count** or constraint count shows how many results match each value

### Refine your results

#### Manufacturer

- Canon USA (5)
- Sony (2)
- Nikon (2)
- Olympus (6)
- Pentax (2)

#### Resolution

- 6 megapixels (3)
- 8 megapixels and up (14)

#### Zoom range

- 3X to 4X (11)
- 8X to 12X (1)

#### More

- LCD size
- Image stabilizer
- Flash memory
- Still image format
- Maximum ISO

[See all >](#)

Canon, Sony, and Nikon are **constraints**, or facet values

The **breadcrumb** trail shows what constraints have already been applied and allows for their removal

you selected:

\$400 - \$500 ✕

SLR ✕

remove all ✕

17 results

1

2

next

Show 10 results per page

Sort by: Review date

COMPARE SELECTED



Canon EOS Rebel XS (silver, with 18-55mm lens)

\$459 to \$699  
at 15 stores



# Key Elements of Faceted Search

- No hierarchy of options is enforced
  - Users can apply facet constraints in any order
  - Users can remove facet constraints in any order
- No surprises
  - The user is only given facets and constraints that make sense in the context of the items they are looking at
  - The user always knows what to expect before they apply a constraint
- Also known as guided navigation, faceted navigation, faceted browsing, parametric search

# Solr's Faceted Search

# Field Faceting

- Specifies a Field to be used as a **Facet**
- Uses each **term** indexed in that Field as a **Constraint**
- Field must be **indexed**
- Can be used multiple times for multiple fields

```
q           = iPhone
fq         = inStock:true
facet      = true
facet.field = color
facet.field = category
```

# Field Faceting Response

```
http://localhost:8983/solr/select?q=iPhone&fq=inStock:true  
&facet=true&facet.field=color&facet.field=category
```

```
<lst name="facet_counts">  
  <lst name="facet_fields">  
    <lst name="color">  
      <int name="red">17</int>  
      <int name="green">6</int>  
      <int name="blue">2</int>  
      <int name="yellow">2</int>  
    <lst name="category">  
      <int name="accessories">16</int>  
      <int name="electronics">11</int>
```

# Or if you prefer JSON...

```
http://localhost:8983/solr/select?q=iPhone&fq=inStock:true  
&facet=true&facet.field=color&facet.field=category&wt=json
```

```
"facet_counts": {  
  "facet_fields": {  
    "color": [  
      "red", 17,  
      "green", 6,  
      "blue", 2,  
      "yellow", 2]  
    "category": [  
      "accessories", 16,  
      "electronics", 11]
```

# Applying Constraints

Assume the user clicked on “red”...

Simply add another filter query to apply that constraint

```
http://localhost:8983/solr/select?  
q=iPhone&fq=inStock:true&fq=color:red&facet=true&  
facet.field=color&facet.field=category
```

Remove redundant facet.field  
(assuming single valued field)

# facet.field Options

- **facet.prefix** - Restricts the possible constraints to only indexed values with a specified prefix.
- **facet.mincount=0** - Restricts the constraints returned to those containing a minimum number of documents in the result set.
- **facet.sort=count** - The ordering of constraints: count or index
- **facet.offset=0** - Indicates how many constraints in the specified sort ordering should be skipped in the response.
- **facet.limit=100** - The number of constraints to return
- **facet.missing=false** – Return the number of docs with no value in the field

# facet.query

- Specifies a query string to be used as a Facet Constraint
- Typically used multiple times to get multiple (discrete) sets
- Any type of query supported

```
facet.query = rank:[ * TO 20 ]
```

```
facet.query = rank:[ 21 TO * ]
```

# facet.query Results

```
<result numFound="27" ... />
...
<lst name="facet_counts">
  <lst name="facet_queries">
    <int name="rank:[* TO 20]">2</int>
    <int name="rank:[21 TO *]">15</int>
  </lst>
  ...
</lst>
...

```

# Spatial faceting

```
q=*:*&facet=true
```

```
pt=45.15,-93.85
```

```
sfield=store
```

```
facet.query={!geofilt d=5}
```

```
facet.query={!geofilt d=10}
```

The lat,lon center point to search from

Name of the field containing lat+lon data

```
"facet_counts":{  
  "facet_queries":{  
    "{!geofilt d=5}":3,  
    "{!geofilt d=10}":6},
```

geospatial query type

# Range Faceting

- Simpler than a sequence of facet.query params

```
http://...&facet=true  
&facet.range=price  
&facet.range.start=0  
&facet.range.end=500  
&facet.range.gap=50
```

```
"facet_counts":{  
  "facet_ranges":{  
    "price":{  
      "counts":[  
        "0.0",5,  
        "50.0",2,  
        "100.0",0,  
        "150.0",2,  
        "200.0",0,  
        "250.0",1,  
        "300.0",2,  
        "350.0",2,  
        "400.0",0,  
        "450.0",1],  
      "gap":50.0,  
      "start":0.0,  
      "end":500.0}}}}
```

# Date Faceting

- **facet.date** is deprecated, use **facet.range** on a date field now
- Creates Constraints based on evenly sized date ranges using the Gregorian Calendar
- Ranges are specified using "Date Math" so they DWIM in spite of variable length months and leap years

```
facet.range           = pubdate
facet.range.start    = NOW/YEAR-1YEAR
facet.range.end      = NOW/MONTH+1MONTH
facet.range.gap      = +1MONTH
```

# Date Faceting Results

```
"facet_counts": {
  "facet_ranges": {
    "pubdate": {
      "counts": [
        "2010-01-01T00:00:00Z", 4,
        "2010-02-01T00:00:00Z", 6,
        "2010-03-01T00:00:00Z", 0,
        "2010-04-01T00:00:00Z", 13,
[ ... ]
        "2011-09-01T00:00:00Z", 5,
        "2011-10-01T00:00:00Z", 2],
      "gap": "+1MONTH",
      "start": "2010-01-01T00:00:00Z",
      "end": "2011-11-01T00:00:00Z" } } }
```

# Range Faceting Options

- **facet.range.hardend=false** - Determines what effective end value is used when the specified "start" and "end" don't divide into even "gap" sized buckets; false means the last Constraint range may be shorter than the others
- **facet.range.other=none** - Allows you to specify what other Constraints you are interested in besides the generated ranges: before, after, between, none, all
- **facet.range.include=lower** – Specifies what bounds are inclusive vs exclusive: lower, upper, edge, outer, all

# Pivot Faceting (trunk)

- Computes a Matrix of Constraint Counts across multiple Facet Fields
- Syntax: `facet.pivot=field1,field2,field3,...`

`facet.pivot=cat,inStock`

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

# Pivot Faceting

<http://...&facet=true&facet.pivot=cat,popularity>

```
"facet_counts":{
```

```
  "facet_pivot":{
```

```
    "cat,popularity":[{
```

```
      "field":"cat",
```

```
      "value":"electronics",
```

```
      "count":14,
```

```
      "pivot":[{
```

```
        "field":"popularity",
```

```
        "value":6,
```

```
        "count":5},
```

```
      {
```

```
        "field":"popularity",
```

```
        "value":7,
```

```
        "count":4},
```

```
(continued)
```

```
{
```

```
  "field":"popularity",
```

```
  "value":1,
```

```
  "count":2}}],
```

```
{
```

```
  "field":"cat",
```

```
  "value":"memory",
```

```
  "count":3,
```

```
  "pivot":[]},
```

```
[...]
```

14 docs w/  
cat==electronics

5 docs w/  
cat==electronics  
&& popularity==6

# Tips & Tricks

# term QParser

- Default Query Parser does special things with whitespace and punctuation
- Problematic when "filtering" on Facet Field Constraints that contain whitespace, punctuation, or other reserved characters.
- Use the term parser to filter on an exact Term

`fq = {!term f=category}Books & Magazines`

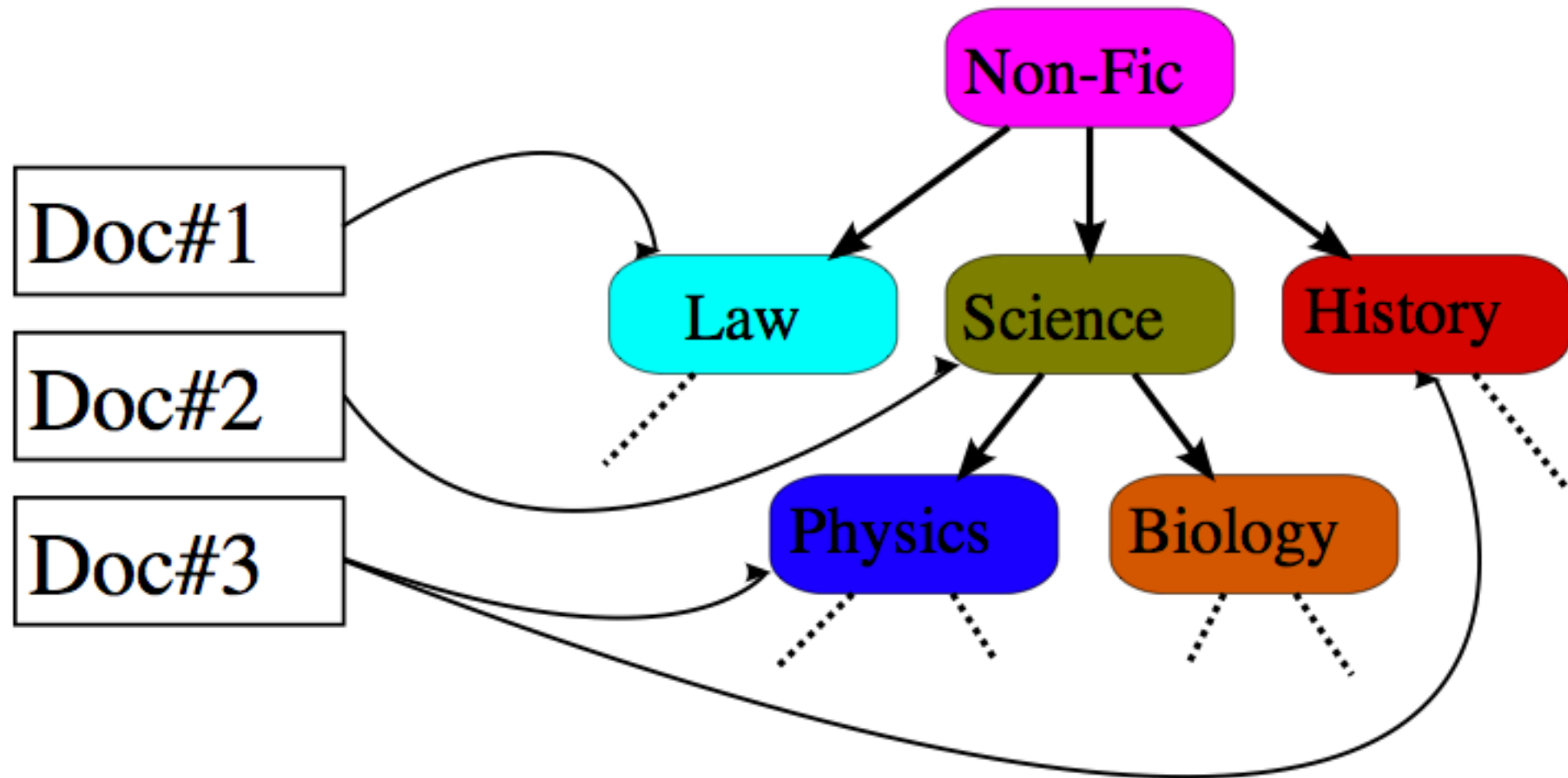
**OR**

`fq = {!term f=category v=$t}`

`t = Books & Magazines`

# Taxonomy Facets

- What If Your Documents Are Organized in a Taxonomy?



# Taxonomy Facets: Data

- Flattened Data

Doc#1: NonFic > Law

Doc#2: NonFic > Sci

Doc#3: NonFic > Hist

NonFic > Sci > Phys

- Indexed Terms (prepend number of nodes in path segment)

Doc#1: 1/NonFic, 2/NonFic/Law

Doc#2: 1/NonFic, 2/NonFic/Sci

Doc#3: 1/NonFic, 2/NonFic/Hist,  
2/NonFic/Sci, 3/NonFic/Sci/Phys

# Taxonomy Facets: Initial Query

```
facet.field      = category
facet.prefix     = 2/NonFic
facet.mincount  = 1
```

```
<result numFound="164" ...
<lst name="facet_fields">
  <lst name="category">
    <int name="2/NonFic/Sci">2</int>
    <int name="2/NonFic/Hist">1</int>
    <int name="2/NonFic/Law">1</int>
```

# Taxonomy Facets: Drill Down

```
fq          = {!term f=category}2/NonFic/Sci
facet.field = category
facet.prefix = 3/NonFic/Sci
facet.mincount = 1
```

```
<result numFound="2" ...
<lst name="facet_fields">
  <lst name="category">
    <int name="3/NonFic/Sci/Phys">1</int>
```

# Multi-Select Faceting

<http://search.lucidimagination.com>

Options

[Clear all facets](#)

---

**PROJECT** [clear projects](#)

- Lucene (3040)
- Solr (1850)
- Nutch (652)
- Tika (6)
- Mahout (31)
- Lucene.Net (181)
- Lucy (4)

---

**SOURCE** [clear sources](#)

- Lucid (4)
- Wiki (16)
- Apache Lucene Web (3)
- Email (4685)
  - user (3882)
  - dev (803)

[Start new search](#)

## Search Result

Found 4,887 results in

[\[WIKI\] Solr Replic](#)  
2009-07-15 09:02  
current **index** version  
steps are as follows,  
<http://wiki.apache.org/>

---

[\[WIKI\] Collection](#)  
2006-05-28 14:54  
schema does not requ  
**replication** Perform the  
<http://wiki.apache.org/>

---

[\[lucene-user\] Ind](#)  
Sent 2002-08-31 by t

- Very generic support
  - Reuses localParams syntax `{!name=val}`
  - Ability to **tag** filters
  - Ability to **exclude** certain filters when faceting, by tag

`q=index replication`

`facet=true`

`fq={!tag=pr}project:(lucene OR solr)`

`facet.field={!ex=pr}project`

`facet.field={!ex=src}source`

# Same Facet, Different Exclusions

- A **key** can be specified for a facet to change the name used to identify it in the response.

```
q = Hot Rod
fq = {!df=colors tag=cx}purple green
facet.field = {!key=all_colors ex=cx}colors
facet.field = {!key=overlap_colors}colors
```

```
"facet_counts": {
  "facet_fields": {
    "all_colors": [
      "red", 19,
      "green", 6,
      "blue", 2],
```

```
    "overlap_colors": [
      "red", 7,
      "green", 6,
      "blue", 1]
    }
  }
```

# “Pretty” facet.field Terms

- Field Faceting uses Indexed Terms
- Leverage **copyField** and TokenFilters that will give you good looking Constraints

```
<tokenizer class="solr.PatternTokenizerFactory"  
  pattern="(,|;)\s*" />  
<filter class="solr.PatternReplaceFilterFactory"  
  pattern="\s+" replacement=" " />  
<filter class="solr.PatternReplaceFilterFactory"  
  pattern=" and " replacement=" & " />  
<filter class="solr.TrimFilterFactory" />  
<filter class="solr.CapitalizationFilterFactory"  
  onlyFirstWord="false" />
```

# “Pretty” facet.field Results

```
{ "id" : "MyExampleDoc",  
  "category" : " books  
and magazines;  
computers, "  
}
```

copyField  
in schema

```
<copyField "source"="category" "dest"="category_pretty"/>
```

```
"facet_counts": {  
  "facet_fields": {  
    "category_pretty": [  
      "Books & Magazines", 1,  
      "Computers", 1 ]  
    }  
  }  
}
```

# facet.field Labels

- facet.query params are echoed verbatim when returning the constraint counts
- Optionally, one can declare a facet.query in solrconfig.xml and include a "label" that the presentation layer can parse out for display.

"label" has no meaning to solr

```
facet.query = {!label='Hot!' }  
              +pop:[1 TO *]  
              +pub_date:[NOW/DAY-1DAY TO *]
```

```
"facet_queries" : {  
  "{!label='Hot!' } pop:[1 TO *] ... " : 15  
}
```

# Performance

# facet.method

name	facet.method	description	memory	CPU
enum	<b>enum</b>	Iterates over terms, calculating set intersections	filter-per-term in the filterCache	$\sim O(nTerms)$
field cache	<b>fc</b> (single-valued field)	Iterates over documents, counting terms	Lucene FieldCache Entry... int [maxDoc]+terms	$O(nDocs)$
UnInvertedField	<b>fc</b> (multi-valued field)	Iterates over documents, counting terms	$O(maxDoc * num\ terms\ per\ doc)$	$\sim O(nDocs)$
Per-segment field cache	<b>fcs</b>	Like field-cache, just better for NRT since FieldCacheEntry is at segment level.	Lucene FieldCache Entries... int [maxDoc]+terms	$O(nDocs) + O(nTerms)$

# facet.method=fc (single-valued field)

Mem=int[maxDoc]  
+unique\_values

CPU=O(nDocs in  
base set)

q=Juggernaut  
&facet=true  
&facet.field=hero

Documents  
matching the  
base query  
"Juggernaut"

0  
2  
7

lookup

Lucene FieldCache Entry  
(StringIndex) for the "hero"  
field

order: for each  
doc, an index into  
the lookup array

lookup: the  
string values

5  
3  
5  
1  
4  
5  
2  
1

(null)  
batman  
flash  
spiderman  
superman  
wolverine

accumulator

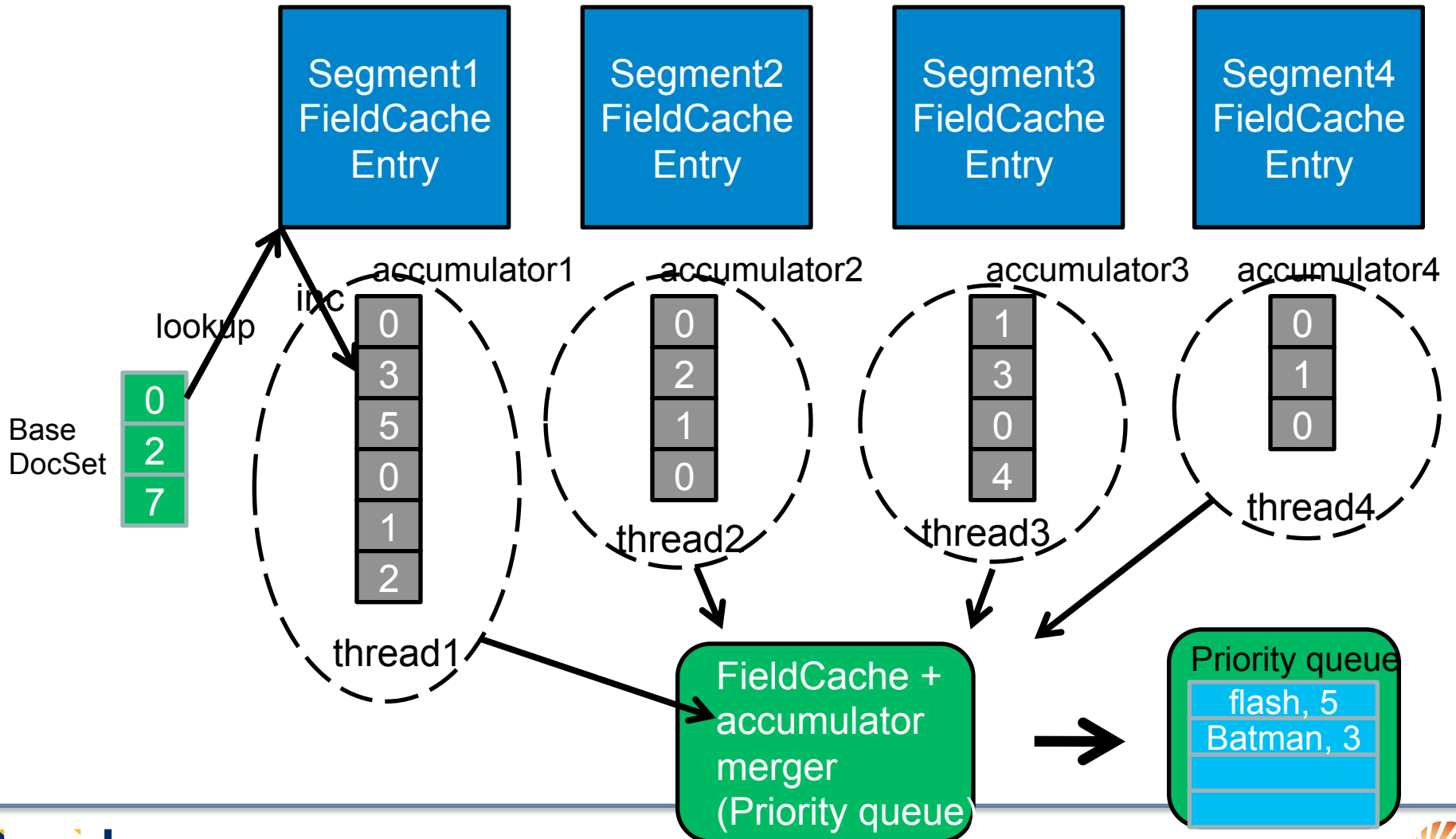
0  
1  
0  
0  
0  
2

increment

Priority queue

flash, 5  
Batman, 3

# facet.method=fcs (trunk) (per-segment single-valued)



# facet.method=fcs

- Controllable multi-threading

```
facet.method=fcs
```

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

- Disadvantages

- Larger memory use (FieldCaches + accumulators)
- Slower (extra FieldCache merge step needed) –  $O(nTerms)$

- Advantages

- Rebuilds FieldCache entries only for new segments (NRT friendly)
- Multi-threaded

# Per-segment faceting performance comparison

Test index: 10M documents, 18 segments, single valued field

Base DocSet=100 docs, facet.field on a field with 100,000 unique terms

**A**

Time for request*	facet.method=fc	facet.method=fcs
static index	3 ms	244 ms
quickly changing index	1388 ms	267 ms

Base DocSet=1,000,000 docs, facet.field on a field with 100 unique terms

**B**

Time for request*	facet.method=fc	facet.method=fcs
static index	26 ms	34 ms
quickly changing index	741 ms	94 ms

\*complete request time, measured externally

# facet.method=fc (multi-valued field)

- UnInvertedField - like single-valued FieldCache algorithm, but with multi-valued FieldCache
- Good for many unique terms, relatively few values per doc
  - Best case: 50x faster, 5x smaller than “enum” (100K unique values, 1-5 per doc)
  - $O(n\_docs)$ , but optimization to count the inverse when  $n > \maxDoc/2$
- Memory efficient
  - Terms in a document are delta coded variable width ords (vints)
  - Ord list for document packed in an int or in a shared byte[]
  - Hybrid approach: “big terms” that match >5% of index use filterCache instead
  - Only 1/128<sup>th</sup> of string values in memory

# facet.method=fc fieldValueCache

- Implicit cache with UnInvertedField entries
  - Not autowarmed – use static warming request
  - <http://localhost:8983/solr/admin/stats.jsp> (mem size, time to create, etc)

name:	fieldValueCache
class:	org.apache.solr.search.FastLRUCache
version:	1.0
description:	Concurrent LRU Cache(maxSize=10000, initialSize=10, minSize=9000, acceptableSize=9500, cleanupThread=false)
stats:	lookups : 45 hits : 43 hitratio : 0.95 inserts : 1 evictions : 0 size : 1 warmupTime : 0 cumulative_lookups : 45 cumulative_hits : 43 cumulative_hitratio : 0.95 cumulative_inserts : 1 cumulative_evictions : 0 item_cat : {field=cat,memSize=5376,tindexSize=52,time=2,phase1=2,nTerms=16,bigTerms=10,termInstances=6,uses=44}

# Faceting: fieldValueCache

- Implicit cache with UnInvertedField entries
  - Not autowarmed – use static warming request
  - <http://localhost:8983/solr/admin/stats.jsp> (mem size, time to create, etc)

```
name: fieldValueCache
class: org.apache.solr.search.FastLRUCache
```

```
item_cat:
{field=cat,memSize=5376,tindexSize=52,time=2,phase1=2,
nTerms=16,bigTerms=10,termInstances=6,uses=44}
```

```
inserts : 1
evictions : 0
size : 1
warmupTime : 0
cumulative_lookups : 45
cumulative_hits : 43
cumulative_hitratio : 0.95
cumulative_inserts : 1
cumulative_evictions : 0
```

```
item_cat : {field=cat,memSize=5376,tindexSize=52,time=2,phase1=2,nTerms=16,bigTerms=10,termInstances=6,uses=44}
```

# Multi-valued faceting: facet.method=enum

- facet.method=enum
- For each term in field:
  - Retrieve filter
  - Calculate intersection size

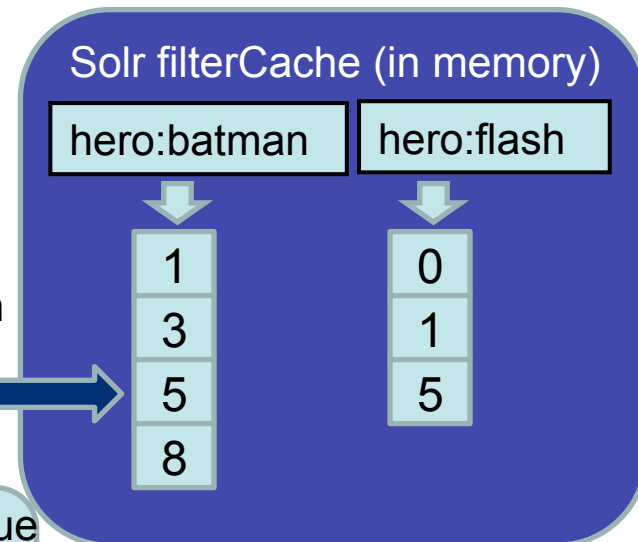
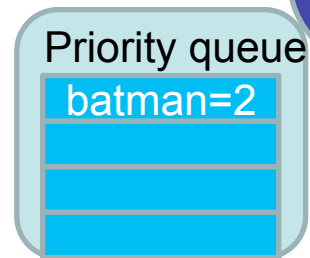
Lucene Inverted Index (on disk)

batman	1	3	5	8
flash	0	1	5	
spiderman	2	4	7	
superman	0	6	9	
wolverine	1	2	7	8

Docs matching  
base query

0  
1  
5  
9

intersection  
count



# facet.method=enum

- $O(n\_terms\_in\_field)$
- Short circuits based on term.df
- filterCache entries `int[ndocs]` or `BitSet(maxDoc)`
- Size filterCache appropriately
  - **Either autowarm filterCache, or use static warming queries (via newSearcher event) in solrconfig.xml**
- `facet.enum.cache.minDf` - prevent filterCache use for small terms
  - **Also useful for huge index w/ no time constraints**

# Q&A