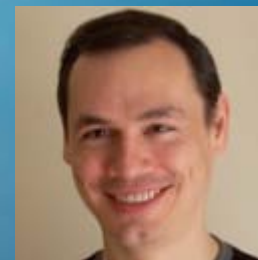




Mastering Solr 1.4

Yonik Seeley
March 4, 2010



Thinking Lucene ▼ Think Lucid.

Agenda

- ▼ Faceting
- ▼ Trie Fields (numeric ranges)
- ▼ Distributed Search
- ▼ 1.5 Preview
- ▼ Q&A
- ▼ See <http://bit.ly/mastersolr> for
 - ▼ Slides for download after the presentation
 - ▼ On-demand viewing of the webinar in ~48 hours

My background



- ▼ Creator of Solr, the Lucene Search Server
- ▼ 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

Getting the most from this talk

- ▶ **Assuming you've been completed the Solr tutorial**
- ▶ **Assuming you're familiar faceting and other high-level Solr/Lucene concepts**
- ▶ **I don't expect you to have deployed Solr in production, but it will provide helpful context**

Faceting Deep Dive

Existing single-valued faceting algorithm

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
- 0
- 2

increment

Existing single-valued faceting algorithm

- ▼ **facet.method=fc**
- ▼ **for each doc in base set:**
 - ▼ `ord = FieldCache.order[doc]`
 - ▼ `accumulator[ord]++`
- ▼ **$O(\text{docs_matching_q})$**
- ▼ **Not used for boolean**

Multi-valued faceting: enum method (1 of 2)

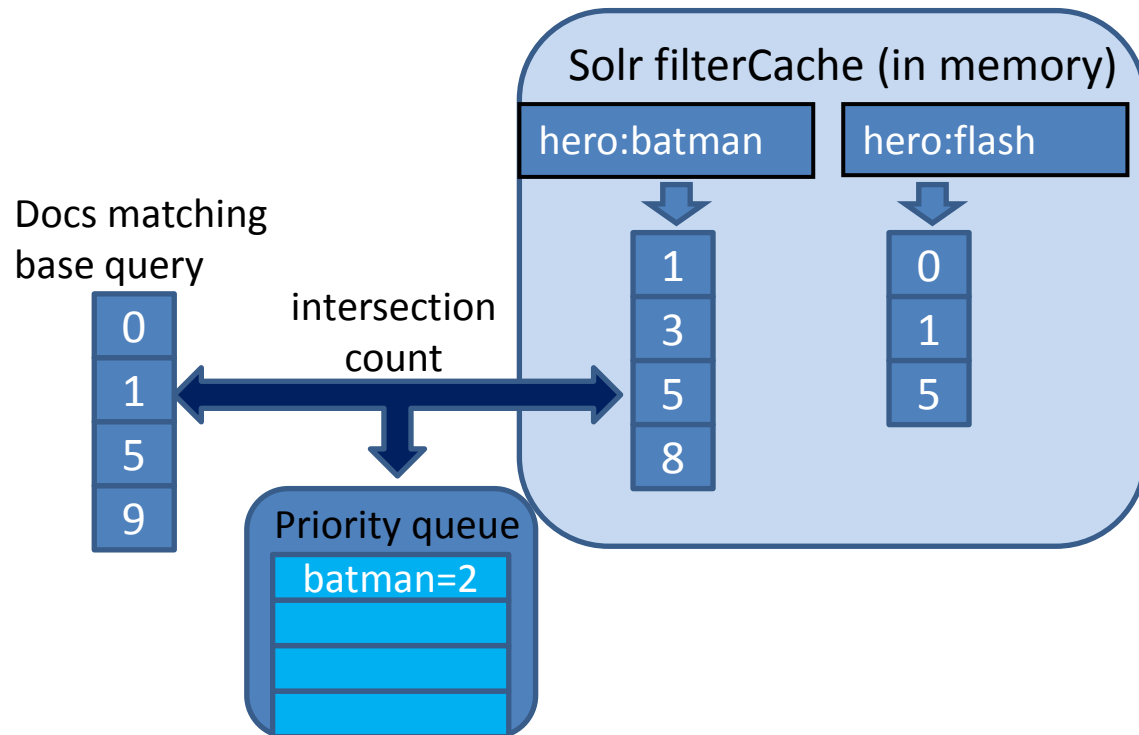
facet.method=enum

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



Multi-valued faceting: enum method (2 of 2)

- ▼ $O(n_terms_in_field)$
- ▼ Short circuits based on `term.df`
- ▼ `filterCache` entries `int[ndocs]` or `BitSet(maxDoc)`
- ▼ `filterCache` concurrency + efficiency upgrades in 1.4
- ▼ Size `filterCache` appropriately
- ▼ Prevent `filterCache` use for small terms with `facet.enum.cache.minDf`

Multi-valued faceting: new UnInvertedField

- ▶ **facet.method=fc**
- ▶ **Like single-valued FieldCache method, but with multi-valued FieldCache**
- ▶ **Good for many unique terms, relatively few values per doc**
 - ▶ Best case: 50x faster, 5x smaller (100K unique values, 1-5 per doc)
 - ▶ $O(n_docs)$, but optimization to count the inverse when $n > \text{maxDoc}/2$
- ▶ **Memory efficient**
 - ▶ Terms in a document are delta coded variable width term numbers
 - ▶ Term number 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 1/128th of string values in memory

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
version:	1.0
description:	Concurrent LRU Cache(maxSize=10000, initialSize=10, minSize=9000, acceptableSize=9500, cleanupThread=false)
stats:	<pre> 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} </pre>

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
version:	1.0
description:	
status:	
item_cat:	{field=cat,memSize=5376,tindexSize=52,time=2,phase1=2,nTerms=16,bigTerms=10,termInstances=6,uses=44}
item_cat :	{field=cat,memSize=5376,tindexSize=52,time=2,phase1=2,nTerms=16,bigTerms=10,termInstances=6,uses=44}
	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}

Migrating from 1.3 to 1.4 faceting

▼ filterCache

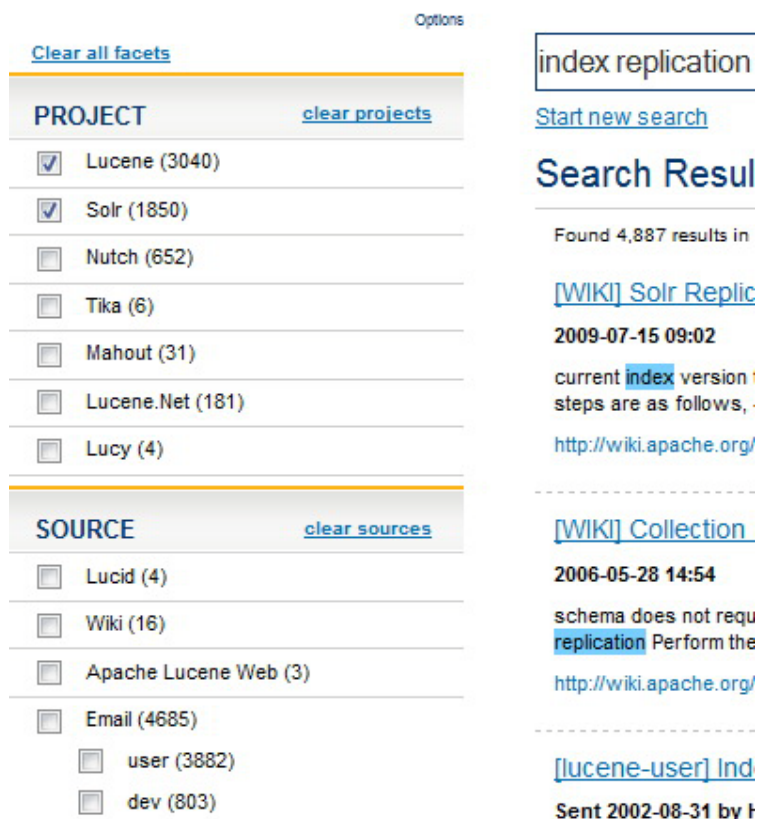
- ▼ Lower size (need room for normal fq filters and “big terms”)
- ▼ Lower or eliminate autowarm count

▼ 1.3 enum method can sometimes be better

- ▼ f.<fieldname>.facet.method=enum
- ▼ Field has many values per document and many unique values
- ▼ Huge index without faceting time constraints

Multi-select faceting

<http://search.lucidimagination.com>



The screenshot shows a search interface with two faceted categories: PROJECT and SOURCE. The PROJECT category is selected, showing Lucene (3040) and Solr (1850) as checked items, and Nutch (652), Tika (6), Mahout (31), Lucene.Net (181), and Lucy (4) as unchecked. The SOURCE category shows Lucid (4), Wiki (16), Apache Lucene Web (3), Email (4685) with sub-items user (3882) and dev (803) as unchecked. A search bar contains 'index replication' and a search result for 'index replication' is displayed, including a date '2009-07-15 09:02' and a snippet about current index version steps.

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=proj}project:(lucene OR solr)`

`&facet.field={!ex=proj}project`

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

New Trie Fields (numeric ranges)

New Trie* fields

- ▶ **Numeric, Date fields index at multiple precisions to speed up range queries**
- ▶ **Base10 Example: 175 is indexed as hundreds:1 tens:17 ones:175**

TrieRangeQuery:[154 TO 183] is executed as
tens:[16 TO 17] OR ones:[154 TO 159] OR ones:[180 TO 183]
- ▶ **Best case: 40x speedup of range queries**
- ▶ **Configurable precisionStep per field (expressed in bits)**
 - ▶ precisionStep=8 means index first 8, then first 16, first 24, and 32 bits
 - ▶ precisionStep=0 means index normally
- ▶ **Extra bonus: more memory efficient FieldCache entries**

Trie* Index Overhead

100,000 documents, random 32 bit integers from 0-1000

Precision Step	Index Size*	Index Size Multiplier
0	223K	1
8	588K	2.6
6	838K	3.7
4	1095K	4.9

100,000 documents, random 32 bit integers

Precision Step	Index Size*	Index Size Multiplier
0	1.17M	1
8	3.03M	2.6
6	3.86M	3.3
4	5.47M	4.7

*Index Size reflects only the portion of the index related to indexing the field

Schema migration

▼ Use int, float, long, double, date for normal numerics

- ▼ `<fieldType name="int" class="solr.TrieIntField" precisionStep="0" omitNorms="true" positionIncrementGap="0"/>`

▼ Use tint, tfloat, tlong, tdouble, tdate for faster range queries

- ▼ `<fieldType name="tint" class="solr.TrieIntField" precisionStep="8" omitNorms="true" positionIncrementGap="0"/>`

- ▼ Date faceting also uses range queries

▼ date, tdate, NOW can be used in function queries

- ▼ Date boosting: `recip(ms(NOW,mydatefield),3.16e-11,1,1)`

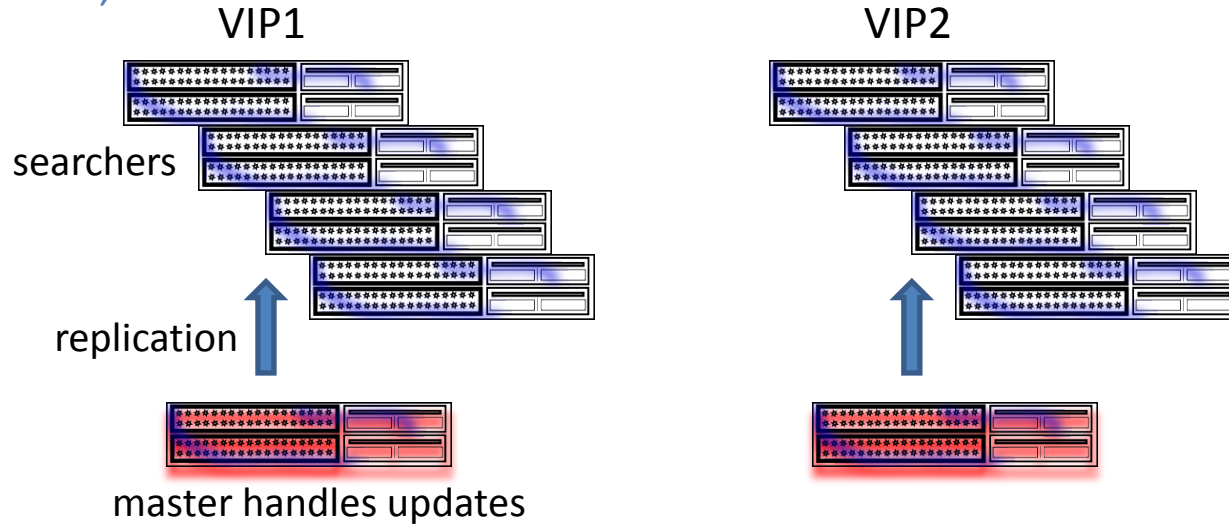
Distributed Search

- ▼ **Split into multiple shards**
 - ▼ When single query latency too long
 - Super-linear speedup possible
 - ▼ Optimal performance when free RAM > shard size
 - Minimum: RAM > (shard index size – stored_fields)
- ▼ **Use replication for HA & increased capacity (queries/sec)**

Distributed Search & Sharding

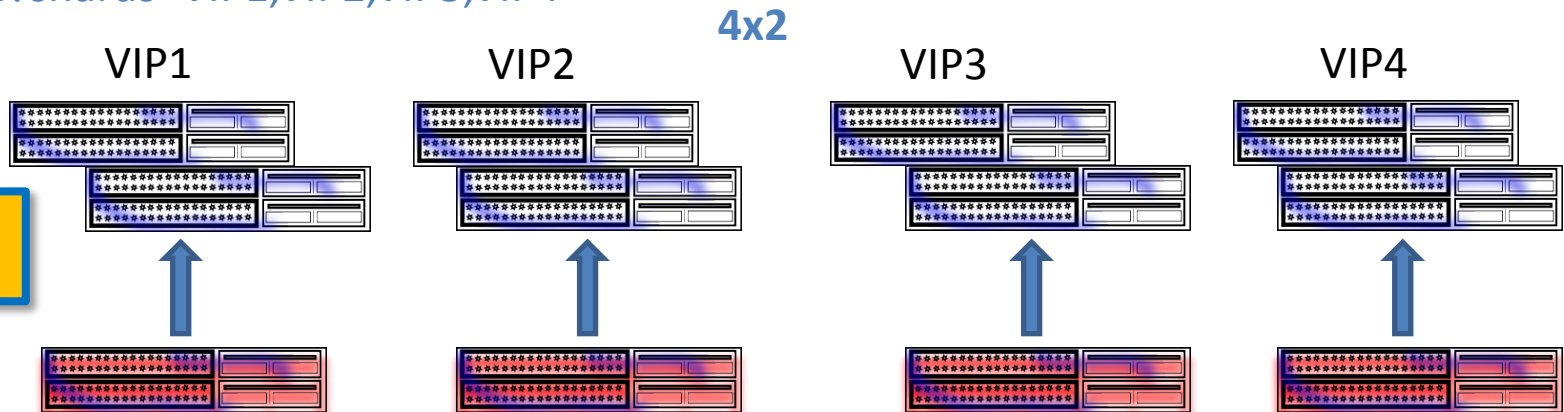
<http://...?shards=VIP1,VIP2>

2x4



<http://...?shards=VIP1,VIP2,VIP3,VIP4>

4x2



Distributed Search: Use Cases

▼ 2 shards x 4 replicas

- ▼ Fewer masters to manage (and fewer total servers)
- ▼ Less increased load on other replicas when one goes down (33% vs 100%)
- ▼ Less network bandwidth

▼ 4 shards x 2 replicas

- ▼ Greater indexing bandwidth

Index Partitioning

▶ **Partition by date**

- ▶ Easy incremental scalability - add more servers over time as needed
- ▶ Easy to remove oldest data
- ▶ Enables increased replication factor for newest data

▶ **Partition by document id**

- ▶ Works well for updating
- ▶ Not easily resizable

▶ **Partitioning to allow querying a subset of shards**

- ▶ Increases system throughput, decreases network bandwidth
- ▶ Partition by userId for mailbox search
- ▶ Partition by region for geographic search

1.5 Preview

1.5 Preview: SolrCloud

▶ **Baby steps toward simplifying cluster management**

▶ **Integrates Zookeeper**

- ▶ Central configuration (solrconfig.xml, etc)
- ▶ Tracks live nodes
- ▶ Tracks shards of collections

▶ **Removes need for external load balancers**

- ▶ shards=localhost:8983/solr|localhost:8900/solr,localhost:7574/solr|localhost:7500/solr

▶ **Can specify logical shard ids**

- ▶ shards=NY_shard,NJ_shard

▶ **Clients don't need to know shards:**

<http://localhost:8983/solr/collection1/select?distrib=true>

1.5 Preview: Spatial Search

▼ **PointType**

- ▼ Compound values: 38.89,-77.03
- ▼ Range queries and exact matches supported

▼ **Distance Functions**

- ▼ haversine

▼ **Sorting by function query**

▼ **Still needed**

- ▼ Ability to return sort values
- ▼ Distance filtering

Q&A

Resources

- ▼ **Apache Solr web site**
 - ▼ <http://lucene.apache.org/solr>
- ▼ **LucidWorks: free Certified Distribution of Solr + Reference Guide**
 - ▼ <http://www.lucidimagination.com/Downloads>
- ▼ **Search all of Lucene/Solr (wiki, mailing lists, issues, ref man, etc)**
 - ▼ <http://search.lucidimagination.com>
- ▼ **Download slides (in ~4 hours) & re-play this talk (~48 hours)**
 - ▼ <http://bit.ly/mastersolr>
- ▼ **Thanks for coming!**