What's New In Apache Solr?

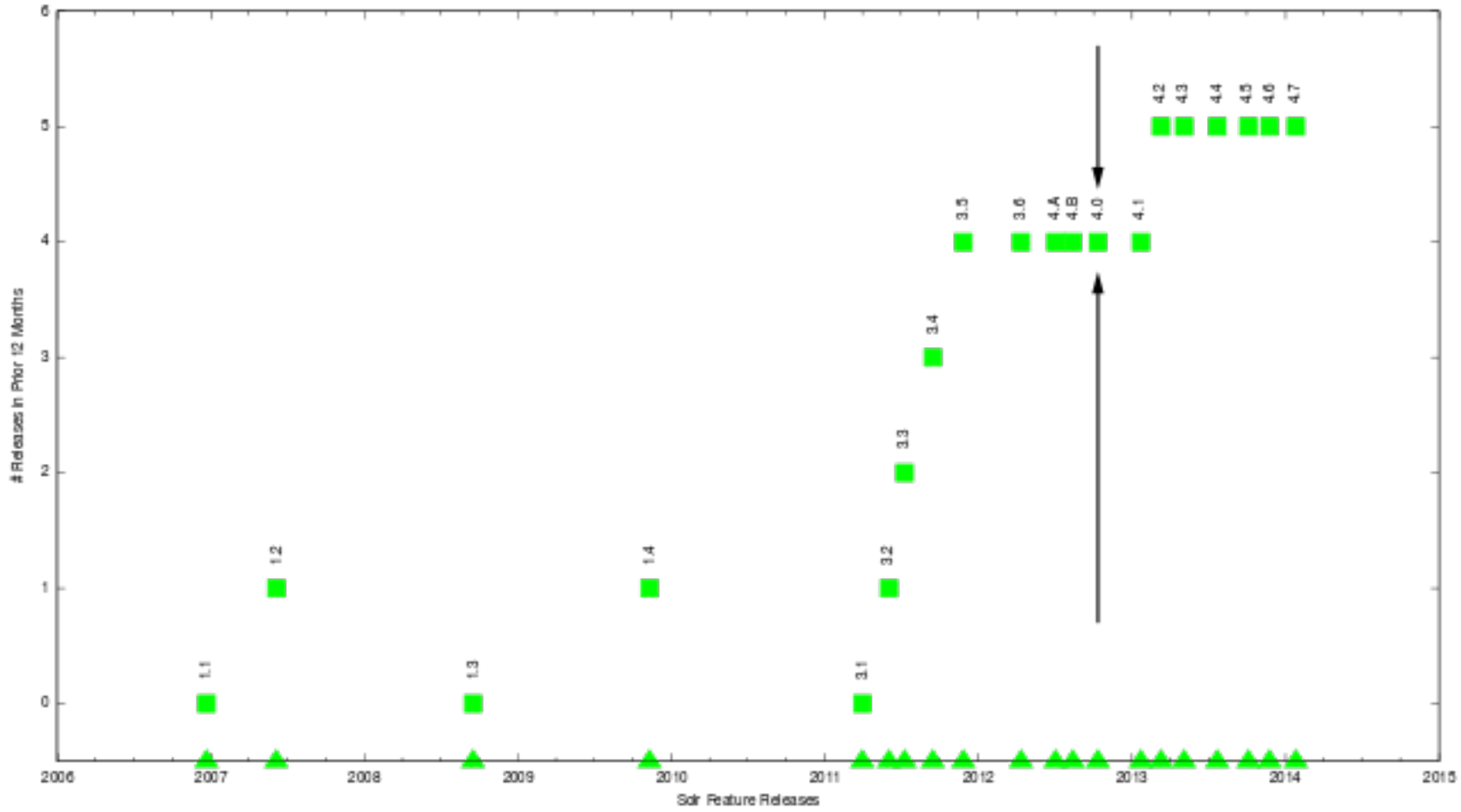
ApacheCon 2014 NA - 2014-04-07

https://people.apache.org/~hossman/ac2014na

https://twitter.com/_hossman

http://www.lucidworks.com/

Acceleration!



Graph shows the dates of every Solr feature release (ie: not bug fix releases) along the X axis, with the Y axis showing the number of Solr releases in the 12 months prior to that release -- giving an additional visual aid to the rate of change of frequency of releases.

Adding Data & schema.xml

(Quick Refresher)

Java (SolrJ)

```
SolrInputDocument doc = new SolrInputDocument();
doc.addField("id",11852);
doc.addField("title","Nightfall");
doc.addField("url","http://www.isfdb.org/cgi-bin/title.cgi?11852");
doc.addField("rating",8.7F);
doc.addField("authors","Isaac Asimov");
doc.addField("authors","Robert Silverberg");
solr_server.add(doc);
```

POST XML

```
<add>
 <doc>
  <field name="id">11852</field>
  <field name="title">Nightfall</field>
  <field name="url">http://www.isfdb.org/cgi-bin/title.cgi?11852</field>
  <field name="rating">8.7</field>
  <field name="authors">Isaac Asimov</field>
  <field name="authors">Robert Silverberg</field>
  ...
```

POST JSON

```
[
 {
  "id":11852,
  "title":"Nightfall",
  "url":"http://www.isfdb.org/cgi-bin/title.cgi?11852",
  "rating":8.7,
  "authors":[ "Isaac Asimov",
              "Robert Silverberg" ]
  ...
```

`schema.xml` Fields

```
<field name="id"        type="tint"   indexed="true"  stored="true" />
<field name="title"     type="text"   indexed="true"  stored="true" />
<field name="url"       type="string" indexed="false" stored="true" />
<field name="summary"   type="text"   indexed="true"  stored="false"/>
<field name="rating"    type="tfloat" indexed="true"  stored="true" />
<field name="authors"   type="text"   multiValued="true" />
...
```

Solr also has Dynamic Fields are rule based fields where the field type is determined by glob against the field name -- but there's a limit to how much we can review in this talk, our goal here is to talk about *new* things in Solr.

`schema.xml` Field Types

```
<fieldType name="string" class="solr.StrField" />
<fieldType name="tint"   class="solr.TrieIntField" precisionStep="0" />
<fieldType name="tfloat" class="solr.TrieFloatField" precisionStep="0" />
<fieldType name="text"   class="solr.TextField" indexed="true" stored="false">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt" />
    ...
```

Different field type classes support different options (example: The "Trie*" field types support `precisionStep`) while other generic options such as `indexed`, `stored`, and `multiValued` can be specified on either a field or a field type -- When specified on a field type, these options are inherited by each field that uses that type unless the field explicitly overrides it.

Schema API

More details about Schema API in the Solr Reference Guide.

FieldTypes API

GET .../schema/fieldtypes

```
 {
  "fieldTypes":[{
       "name":"string",
       "class":"solr.StrField",
       "fields":["url",...]},
     {
       "name":"tint",
       "class":"solr.TrieIntField",
       "precisionStep":"0",
       "fields":["id"]},
     {
       "name":"text",
       "class":"solr.TextField",
       "indexed":true,
       "stored":false,
       "indexAnalyzer":{
         "tokenizer":{
           "class":"solr.StandardTokenizerFactory"}
   ...
```

As of Solr 4.7, the field type REST API only supports GET. (read only)

Fields API

```
GET .../schema/fields
```

```
 {
   "fields":[{
       "name":"id",
       "type":"tint",
       "indexed":true,
       "stored":true},
     {
       "name":"url",
       "type":"string",
       "indexed":false,
       "stored":true},
     {
       "name":"authors",
       "multiValued":true,
       "type":"text"},
     ...
```

As of Solr 4.7, the field REST API supports GET and PUT for reading info about fields, and creating new fields. Existing fields can not be modified via the API.

PUT support requires that you use a "Managed Schema". (see below)

Managed Schema

More details about Managed Schemas in the Solr Reference Guide.

Mutable Schema

Enabled in `solrconfig.xml`

```
<schemaFactory class="ManagedIndexSchemaFactory">
  <bool name="mutable">true</bool>
  <str name="managedSchemaResourceName">managed-schema</str>
</schemaFactory>
```

`managedSchemaResourceName` is the name of a file that should be used for storing the managed schema metadata. If this file doesn't exist, the Schema factory will look for an existing `schema.xml` file to convert -- making it very easy for existing users to switch to having a managed schema.

`mutable` controls whether the managed schema can be modified at run time. You can initially set it to `true` to allow fields to be created at run time, and then once you are happy with your schema you can set it to `false` to prevent errant changes to your schema.

Schema-Less?

More details about Schemaless Mode in the Solr Reference Guide.

Personally, I think "Schemaless" is a poor name for this type of setup - a better way to think about it is "Data Driven Schema".

Add Fields Automatically

```
<processor class="solr.AddSchemaFieldsUpdateProcessorFactory">
  <str name="defaultFieldType">text</str>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Boolean</str>
    <str name="fieldType">boolean</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.util.Date</str>
    <str name="fieldType">tdate</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Integer</str>
    <str name="fieldType">tint</str>
  </lst>
  <lst name="typeMapping">
    <str name="valueClass">java.lang.Number</str>
    <str name="fieldType">tdouble</str>
  </lst>
</processor>
```

`AddSchemaFieldsUpdateProcessorFactory` can be defined in `solrconfig.xml` -- either your default `updateRequestProcessorChain`, or in a specific named chain, so you could choose to apply it only to updates from certain clients.

The `typeMapping` rules are applied in order defined.

Value Type Helpers

```
<processor class="solr.ParseBooleanFieldUpdateProcessorFactory"/>
<processor class="solr.ParseLongFieldUpdateProcessorFactory"/>
<processor class="solr.ParseDoubleFieldUpdateProcessorFactory"/>
<processor class="solr.ParseDateFieldUpdateProcessorFactory">
  <str name="defaultTimeZone">Europe/Paris</str>
  <str name="locale">fr_FR</str>
  <arr name="format">
    <str>'le' EEEE dd MMMM yyyy</str>
    <str>'le' dd MMM. yyyy 'à' HH 'h' mm</str>
    ...
```

These processors can be configured prior to `AddSchemaFieldsUpdateProcessorFactory` if you expect updates from non-Java clients where the underlying data type may not be preserved.

With formats like JSON, Solr automatically can tell when a field should be text, vs. boolean, vs. a number -- but not whether a certain string should be parsed as a date, or if a certain numeric value should be treated as a float vs an int. These processors (and the order they are executed in) can help resolve these ambiguities.

These processors can also be helpful when clients send you "unclean" formatted data -- for example, sending numeric values that have been formatted as Strings in a particular locale convention. For example, client code might format numbers using `ru_RU` string formatting conventions, indexing `"12 345,899"` instead of `12345.899`. The `ParseDoubleFieldUpdateProcessorFactory` can be configured with locale information to parse that for you.

Queries & Pagination

(Quick Refresher)

More details about Pagination of Results in the Solr Reference Guide.

Search, Filter, Sort, Paginate

```
    q = title:Nightfall      # Affects score
   fq = rating:[5.0 TO *]    # Constrains result set, non-scoring
 sort = score desc           # Order of result list
start = 0                    # Offset in result list
 rows = 20                   # Size of result list slice
```

More details about Common Query Parameters in the Solr Reference Guide.

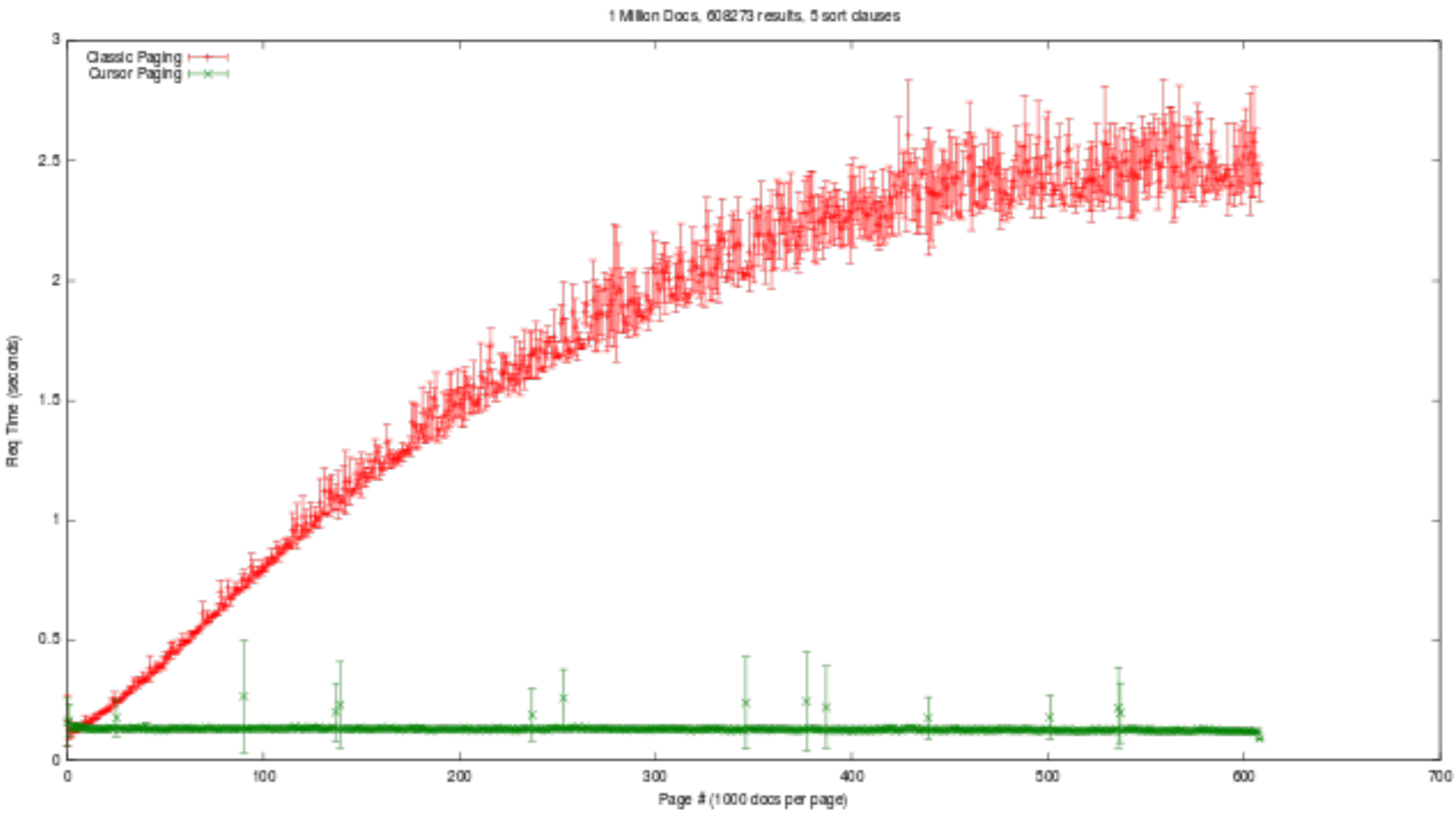
Page #1

```
...&sort=score+desc&rows=20&start=0

 {
   "response":{"numFound":32145678,"start":0,"docs":[
     {
       "id":11852,
       "title":"Nightfall",
       "url":"http://www.isfdb.org/cgi-bin/title.cgi?11852",
       "rating":8.7,
       "authors":[ "Isaac Asimov",
                   "Robert Silverberg" ]},
     ...
```

Page #2

```
...&sort=score+desc&rows=20&start=20

 {
   "response":{"numFound":32145678,"start":20,"docs":[
     {
       "id":15475,
       "title":"The Legend of Nightfall",
       "url":"http://www.isfdb.org/cgi-bin/title.cgi?15475",
       "rating":4.3,
       "authors":[ "Mickey Zucker Reichert" ]},
     ...
```

Pagination Performance?



The Red line here shows the performance of classic pagination (continuously increasing the `start` parameter) compared to using `cursorMark` (the Green line) to fetch a large number of result sets using non-trivial sort criteria.

Graph generated from performance data available in a SearchHub blog post I wrote in December 2013. (Which includes additional graphs and details of methodology)

RED IS BAD

Cursors To The Rescue!

Start Cursor

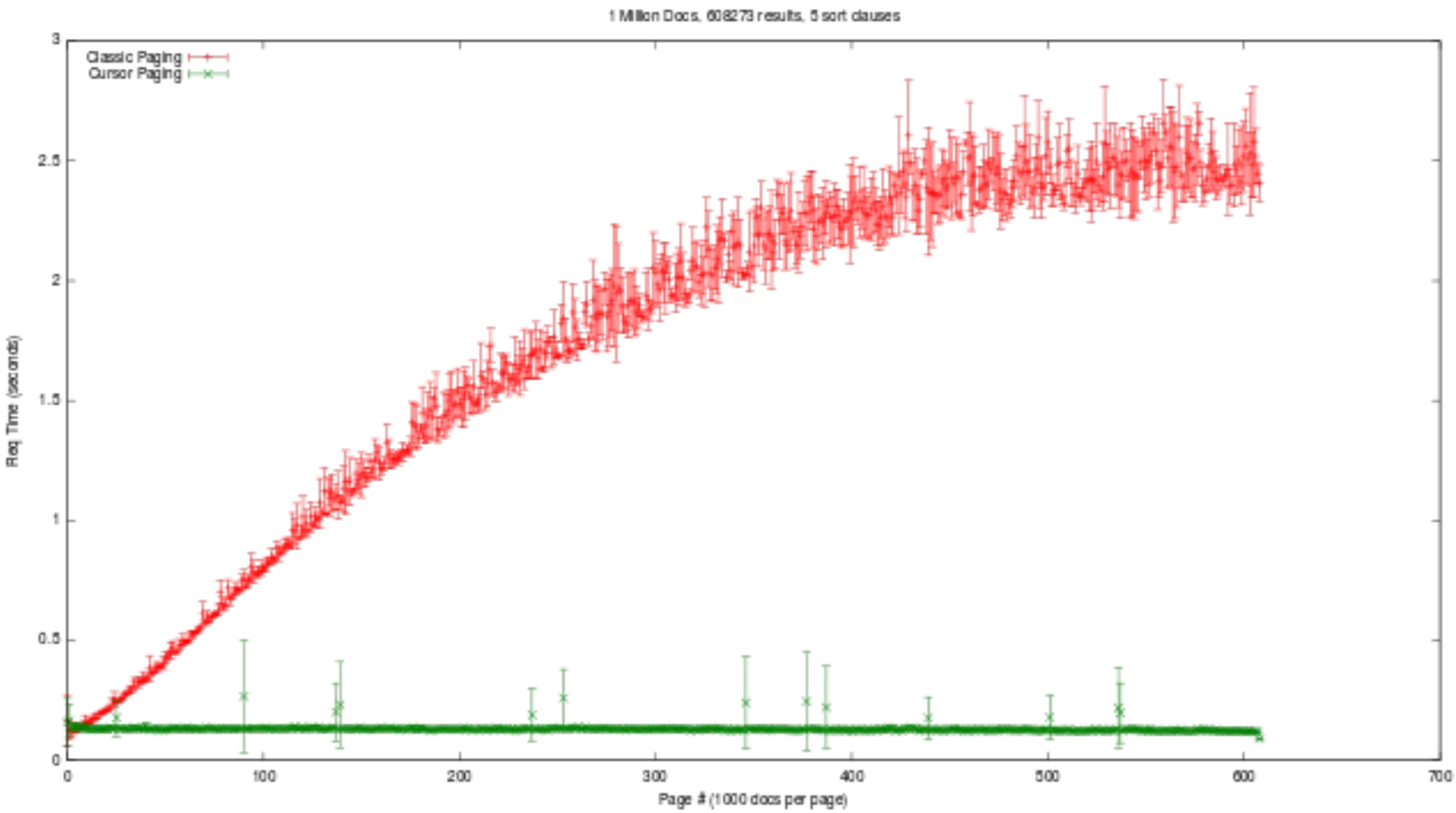
```
...&sort=score+desc,id+desc&cursorMark=*
```

```
{
  "response":{"numFound":32145678,"start":0,"docs":[
    {
      "id":11852,
      "title":"Nightfall",
      "url":"http://www.isfdb.org/cgi-bin/title.cgi?11852",
      "rating":8.7,
      "authors":[ "Isaac Asimov",
                  "Robert Silverberg" ]},
    ...
    ]},
  "nextCursorMark":"AoEjR0JQ"}
```

Next Cursor

```
...&sort=score+desc,id+desc&cursorMark=AoEjR0JQ
```

```
{
  "response":{"numFound":32145678,"start":0,"docs":[
    {
      "id":15475,
      "title":"The Legend of Nightfall",
      "url":"http://www.isfdb.org/cgi-bin/title.cgi?15475",
      "rating":4.3,
      "authors":[ "Mickey Zucker Reichert" ]},
    ...
    ]},
  "nextCursorMark":"AoEpVkRCREIxQTE2"}
```



The Red line here shows the performance of classic pagination (continuously increasing the `start` parameter) compared to using `cursorMark` (the Green line) to fetch a large number of result sets using non-trivial sort criteria.

Graph generated from performance data available in a SearchHub blog post I wrote in December 2013. (Which includes additional graphs and details of methodology)

GREEN IS GOOD

Sorting, Faceting, & DocValues

Inverted Indexes

☑ **Fast Term Search** ☺

Input

D1: Nightfall
D2: A Time to Rend
D3: The Legend of Nightfall
D4: Legends from the End of Time
D5: Time of Legends
D6: Legends
D7: About Time

Index

a → D2
about → D7
end → D4
from → D4
legend → D3, D4, D5, D6
nightfall → D1, D3
of → D3, D4, D5
rend → D2
the → D3, D4
time → D2, D4, D5
to → D2

These slides visually represents the basic principle of an Inverted Index: optimizing the look-up of "terms" to find "documents" (not the other way around) but they are extremely simplified in terms of what the actual data structures used in Lucene & Solr look like.

There is a lot more going on, particularly in terms of how the term data is "packed" and encoded on disk, and what in memory structures are maintained to "skip" over terms during look-up. For the purposes of this presentation however, the key basics are covered.

Inverted Indexes

☑ Fast Range Search ☺

Input

D1: 8.7
D2: 3.2
D3: 4.3
D4: 4.3
D5: 1.8
D6: 5.7
D7: 4.5

Index

1.8 → D5
3.2 → D2
4.3 → D3, D4
4.5 → D7
5.7 → D6
8.7 → D1

Inverted Indexes

☐ **Fast Sorting** ☹
☐ **Fast Faceting** ☹

Input	Index	FieldCache
D1: 8.7	1.8 → D5	D1 → 8.7
D2: 3.2	3.2 → D2	D2 → 3.2
D3: 4.3	4.3 → D3, D4	D3 → 4.3
D4: 4.3	4.5 → D7	D4 → 4.3
D5: 1.8	5.7 → D6	D5 → 1.8
D6: 5.7	8.7 → D1	D6 → 5.7
D7: 4.5		D7 → 4.5

The FieldCache is built at request time by "un-inverting" the Indexed terms.

DocValues

☑ **Fast Sorting** ☺
☑ **Fast Faceting** ☺

Input	DocValues
D1: 8.7	D1 → 8.7
D2: 3.2	D2 → 3.2
D3: 4.3	D3 → 4.3
D4: 4.3	D4 → 4.3
D5: 1.8	D5 → 1.8
D6: 5.7	D6 → 5.7
D7: 4.5	D7 → 4.5

Unlike the FieldCache, DocValues are built when constructing your index.

As with the previous slides, this is a simple visually representation of the basic principle behind DocValues: optimizing the look-up of "document ids" to find "values" (not the other way around) but they are extremely simplified in terms of what the actual data structures used in Lucene & Solr look like. DocValues (particularly the default DocValues format) are heavily optimized for space & speed, and designed to let the bulk of the data remain on disk, with only small data structures loaded into the JVM memory.

Generally speaking: using DocValues instead of relying on the FieldCache for faceting and/or sorting on a field should reduce JVM RAM usage and increase request speed, particularly on the first request and in "NRT" situations. If you *also* need to search on the field as well, you will still want to index it -- and having both the inverted index and the docvalues for a field will certainly result in an increase in indexing time and use more disk than just one or the other.

DocValues in `schema.xml`

```
<field name="rating"     type="tint"   indexed="true"  docValues="true"  />
<field name="title_sort" type="string" indexed="false" docValues="true"  />
<field name="title"      type="text"   indexed="true"  docValues="false" />
```

More details about using DocValues in the Solr Reference Guide.

Solr Cloud

(Just Scratching the Surface)

There have been some huge advances in Solr Cloud related functionality since 4.0, but I'm only going to briefly mention some of the major highlights, since there are several Solr Cloud specific talks happening today & tomorrow that will go into much more depth:

- Introduction to SolrCloud
- Solr's SolrCloud, The State of the Union
- Building Google-in-a-box: using Apache SolrCloud and Bigtop to index your bigdata
- Deploying and managing SolrCloud in the cloud

New In Solr Cloud

- Custom Sharding & Routing via `router.name`:
  - `compositeId`: Hash based, optionally driven by id prefix
  - `implicit`: Infer shard from where client sent document
    ....or `_route_` param
- Shard Splitting w/o Downtime:
  - Divide shards in two on the fly as your index grows
  - Optionally split by ranges or route key via `split.key`
- Hadoop Integration:
  - Keeping indexes in HDFS
  - Building Indexes with Map Reduce

More details about SolrCloud in the Solr Reference Guide:

- Document Routing
- Shard Splitting
- Running Solr on HDFS
- Building indexes with Map-Reduce (Not yet covered in Reference Guide)

Documentation!

(No, Seriously: *Good* Documentation)

Solr Reference Guide

https://lucene.apache.org/solr/documentation.html

- Online "Live" documentation maintained in Confluence
  - Supports public comments for questions & feedback
- Formally released PDFs for each major feature release of Solr
  - Available from the Apache mirror network

"Live" Documentation meaning it can be updated by project members as features are committed, as opposed to the "released" documentation which is snapshotted in time.

Anything Else?

Lots More, Not Enough Time

- Release announcement highlights
  - https://lucene.apache.org/solr/solrnews.html
- Detailed Changes
  - https://lucene.apache.org/solr/4_7_0/changes/Changes.html

Every release announcement includes a list of "Highlights" from the developers to to draw attention to some of the more significant new features.

The authoritative copy of `CHANGES.txt` lives in SVN, but with each release we publish an HTML-ified version that makes it easy to drill down in the lists of New Features, Bug Fixes, etc....

Q & A

- Me
  - https://twitter.com/_hossman
- My Company
  - http://www.lucidworks.com/
- These Slides
  - https://people.apache.org/~hossman/ac2014na
- Solr Docs
  - https://lucene.apache.org/solr/documentation.html
- Mailing Lists & IRC
  - https://lucene.apache.org/solr/discussion.html
- Join The Revolution in DC, November 11-14
  - http://www.lucenerevolution.org/2014/call-for-speakers