ADVANCED DATABASES

CIS 6930

Dr. Markus Schneider

Apache Solr

Group 21

Prarabdh Joshi
Himanshu Vyas
Mark Steele
Jiangjiang Zhu

What is Solr ?

- Solr is an Open Source Search Platform, built on top of Lucene Java Search Library.
- It exposes the Lucene Java API as REST-Full Services
- Indexing in Solr can be done via XML, JSON, CSV or Binary over HTTP protocol.
- Solr provides essential configurations to make data extraction simple even from Rich Documents like pdfs, presentations, Doc files and spreadsheets.
- Queries are made using HTTP GET Method and the results are retrieved in XML, JSON, CSV or Binary Format.

History

- Solr was created by “Yonik Seeley” at CNET Networks in 2004.
- Basically Developed as a In-House project, aimed at adding Search Capabilities to the Company’s Website.
- It Initially had just a Master-Slave architecture, limiting it to small data sets with Scalability Issues.
- In 2006, CNET released it’s source code to Apache Software Foundation under the Lucene Top Level Project.
- In 2008, Solr 1.3 was released with added features including Distributed Search Capabilities.
- The latest version 6 of Solr was released in April 2016, adding support for executing parallel SQL queries and SolrCloud Collections.

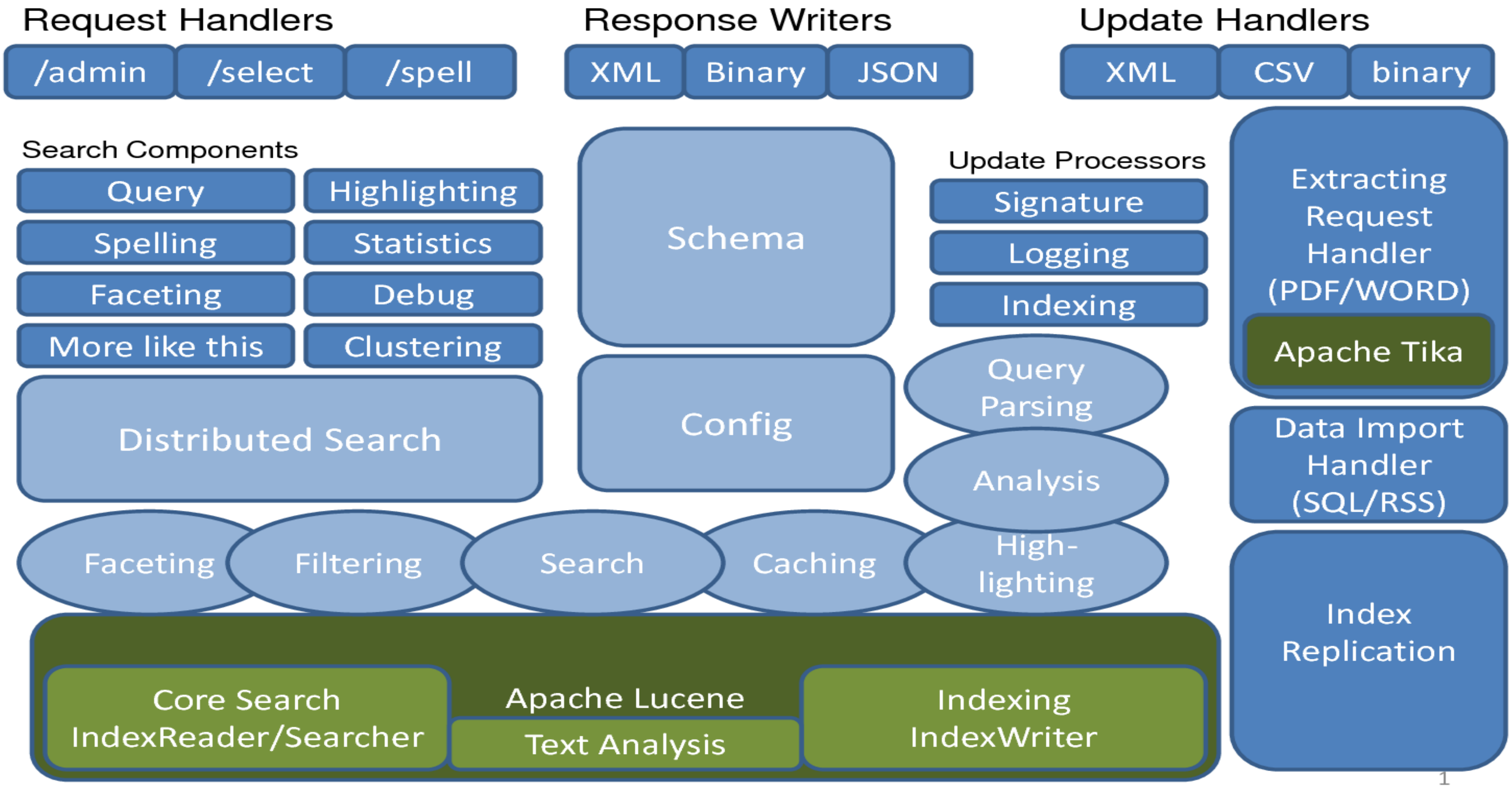
Features in a Nutshell

- Advanced Full Text Search Capability
- Faceted Navigation through the Retrieved Data
- Optimization for High Value Web Traffic
- HTML administration interface
- Distributed Search through Sharding
- Auto Suggest and Auto Completion

More Features

- Automated Indexing of Distributed Documents
- JSON, XML, PHP, Ruby, Python and custom Java binary output formats over the HTTP protocol.
- Built-in security: Authentication, Authorization, SSL
- Near Real Time Search
- High Availability for Writes
- Auto Index Replication
- Extensive Plug In Architecture

Lucene/Solr Architecture



Request Handlers

/admin /select /spell

Response Writers

XML Binary JSON

Update Handlers

XML CSV binary

Search Components

Query Highlighting

Spelling Statistics

Faceting Debug

More like this Clustering

Distributed Search

Schema

Config

Update Processors

Signature

Logging

Indexing

Query Parsing

Analysis

Extracting Request Handler (PDF/WORD)

Apache Tika

Data Import Handler (SQL/RSS)

Faceting Filtering Search Caching High-lighting

Index Replication

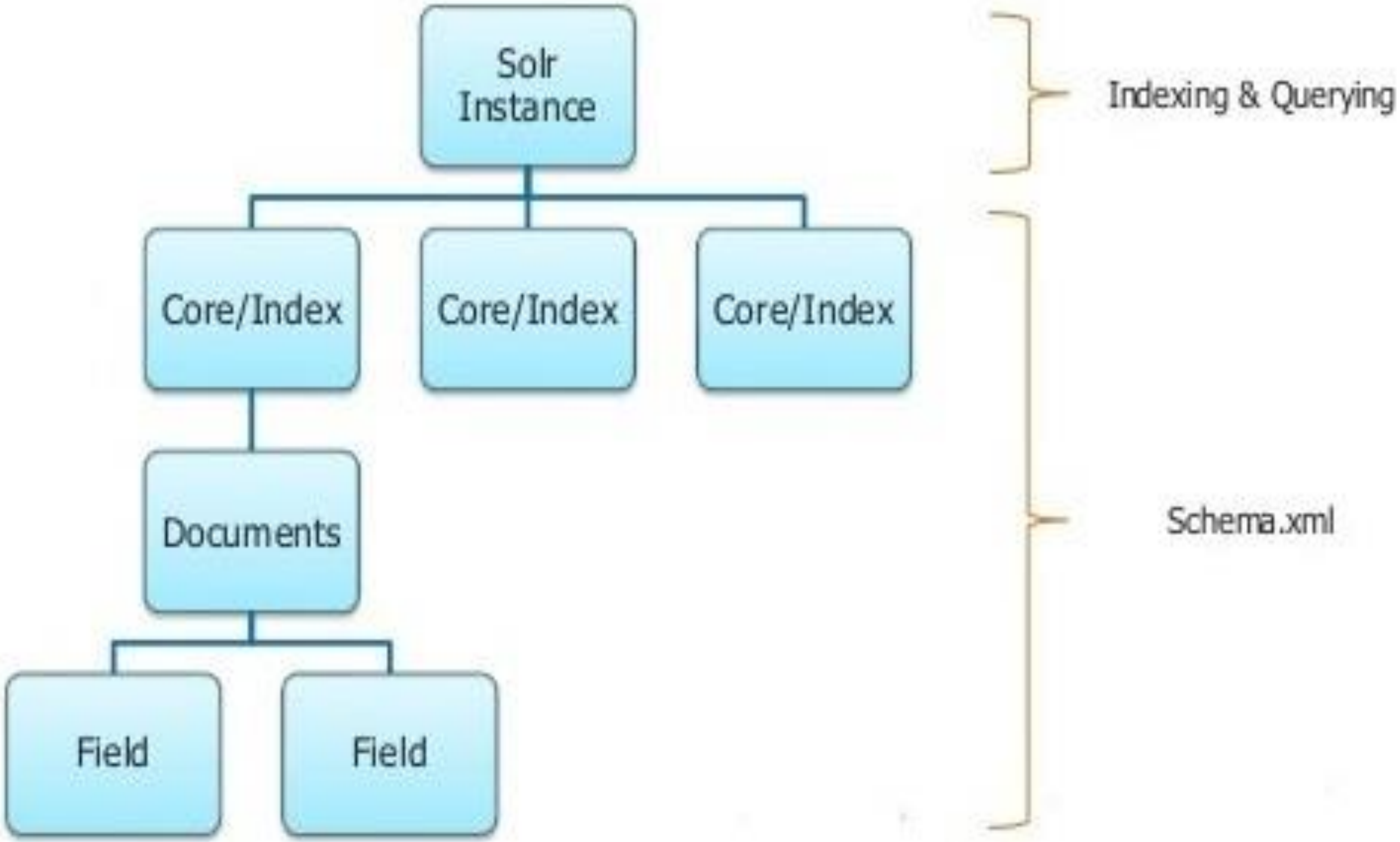
Apache Lucene

Core Search IndexReader/Searcher

Text Analysis

Indexing IndexWriter

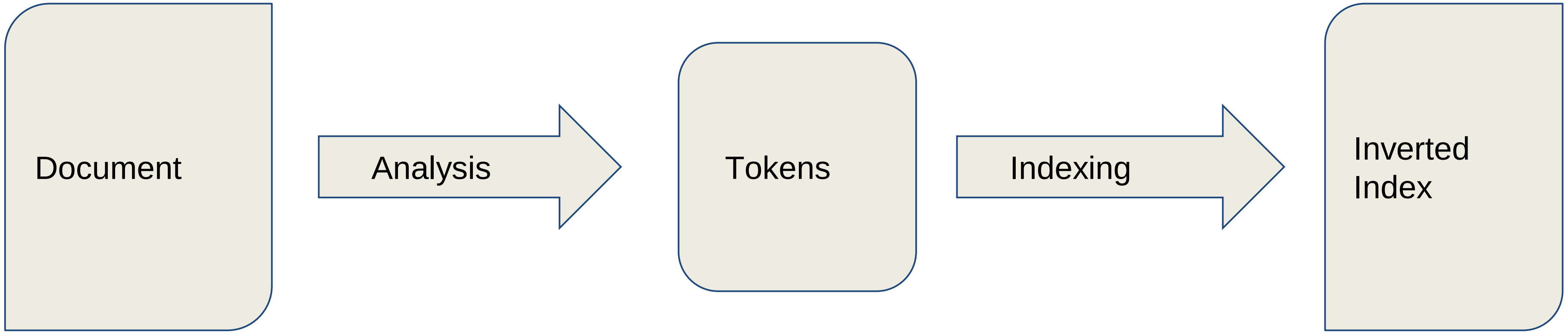
Solr Schema Hierarchy



Why Indexing?

- Indexing Collects, parses and stores Data for Information Retrieval
- It helps in optimizing Speed and Performance for relevant data search
- Without Indexing, Search Engines would scan every Document in the staple, requiring considerable time and computing

Index : Flow



Document

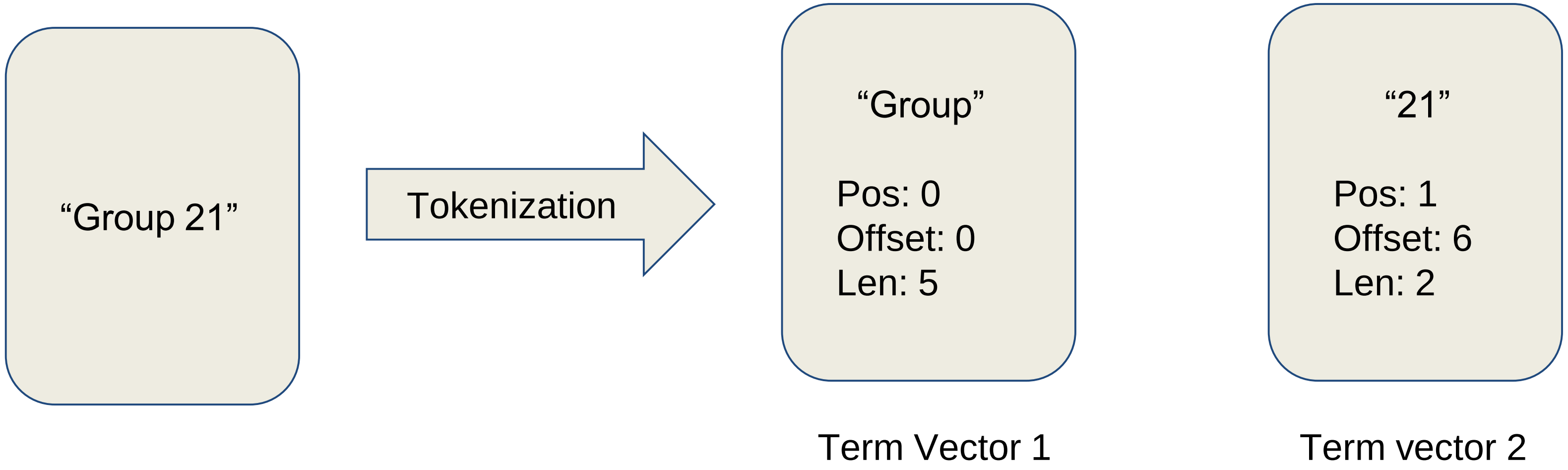
Analysis

Tokens

Indexing

Inverted Index

An Example for Tokenization:



“Group 21”

Tokenization

“Group”

Pos: 0
Offset: 0
Len: 5

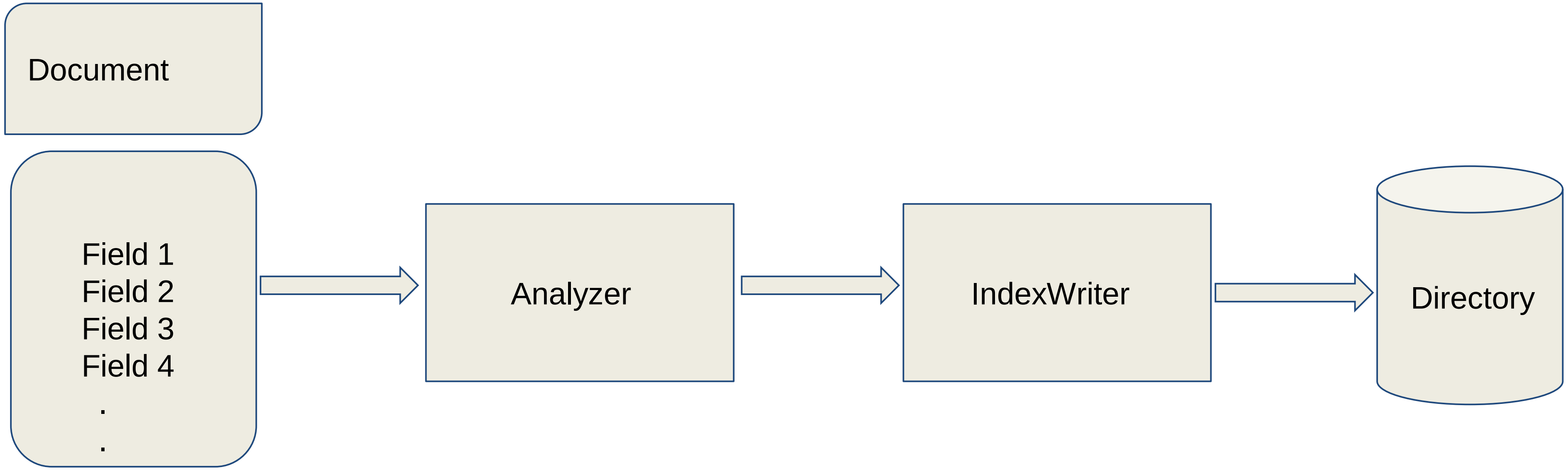
“21”

Pos: 1
Offset: 6
Len: 2

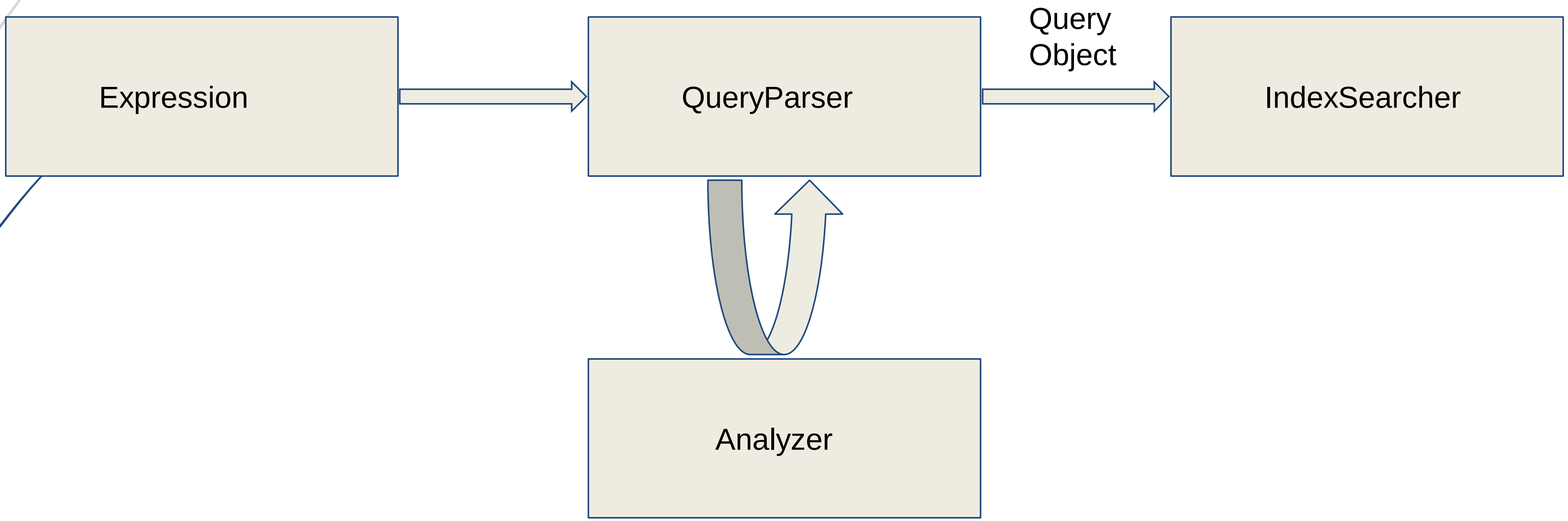
Term Vector 1

Term vector 2

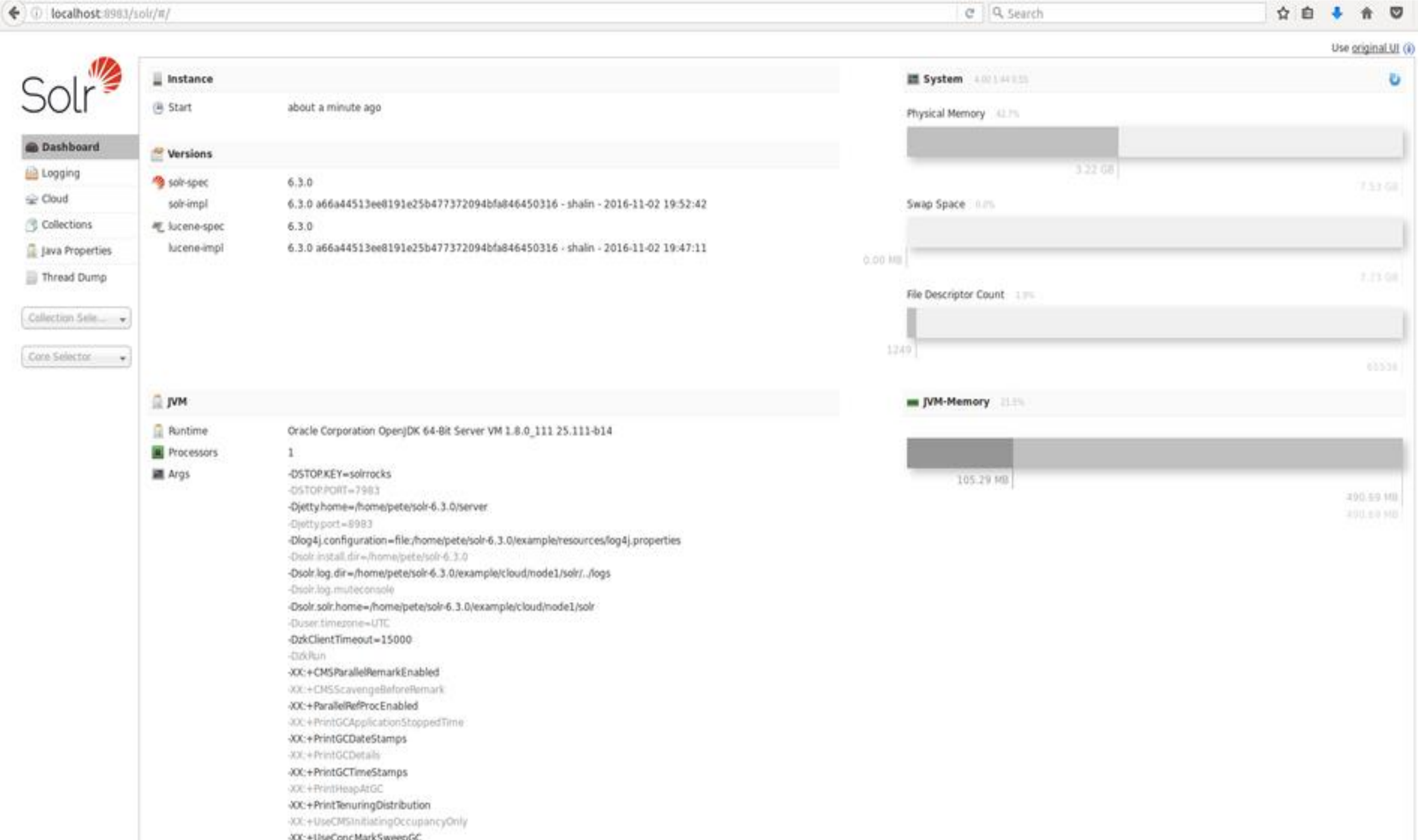
Writing to Index : The Lucene Way



Searching In Lucene



Solr Admin UI



Instance

Start	about a minute ago

Versions

solr-spec	6.3.0
solr-impl	6.3.0 a66a44513ee8191e25b477372094bfa846450316 - shalin - 2016-11-02 19:52:42
lucene-spec	6.3.0
lucene-impl	6.3.0 a66a44513ee8191e25b477372094bfa846450316 - shalin - 2016-11-02 19:47:11

JVM

Runtime	Oracle Corporation OpenJDK 64-Bit Server VM 1.8.0_111 25.111-b14
Processors	1
Args	-DSTOP.KEY=solrrocks -DSTOP.PORT=7983 -Djetty.home=/home/pete/solr-6.3.0/server -Djetty.port=8983 -Dlog4j.configuration=file:/home/pete/solr-6.3.0/example/resources/log4j.properties -Dsolr.install.dir=/home/pete/solr-6.3.0 -Dsolr.log.dir=/home/pete/solr-6.3.0/example/cloud/node1/solr/../logs -Dsolr.log.muteconsole -Dsolr.solr.home=/home/pete/solr-6.3.0/example/cloud/node1/solr -Duser.timezone=UTC -DzkClientTimeout=15000 -DzkRun -XX:+CMSParallelRemarkEnabled -XX:+CMSScavengeBeforeRemark -XX:+ParallelRefProcEnabled -XX:+PrintGCApplicationStoppedTime -XX:+PrintGCDateStamps -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+PrintHeapAtGC -XX:+PrintTenuringDistribution -XX:+UseCMSInitiatingOccupancyOnly -XX:+UseConcMarkSweepGC

System 4.00 1.44 0.55

Physical Memory 42.7%

3.22 GB

7.53 GB

Swap Space 0.0%

0.00 MB

7.73 GB

File Descriptor Count 1.9%

1249

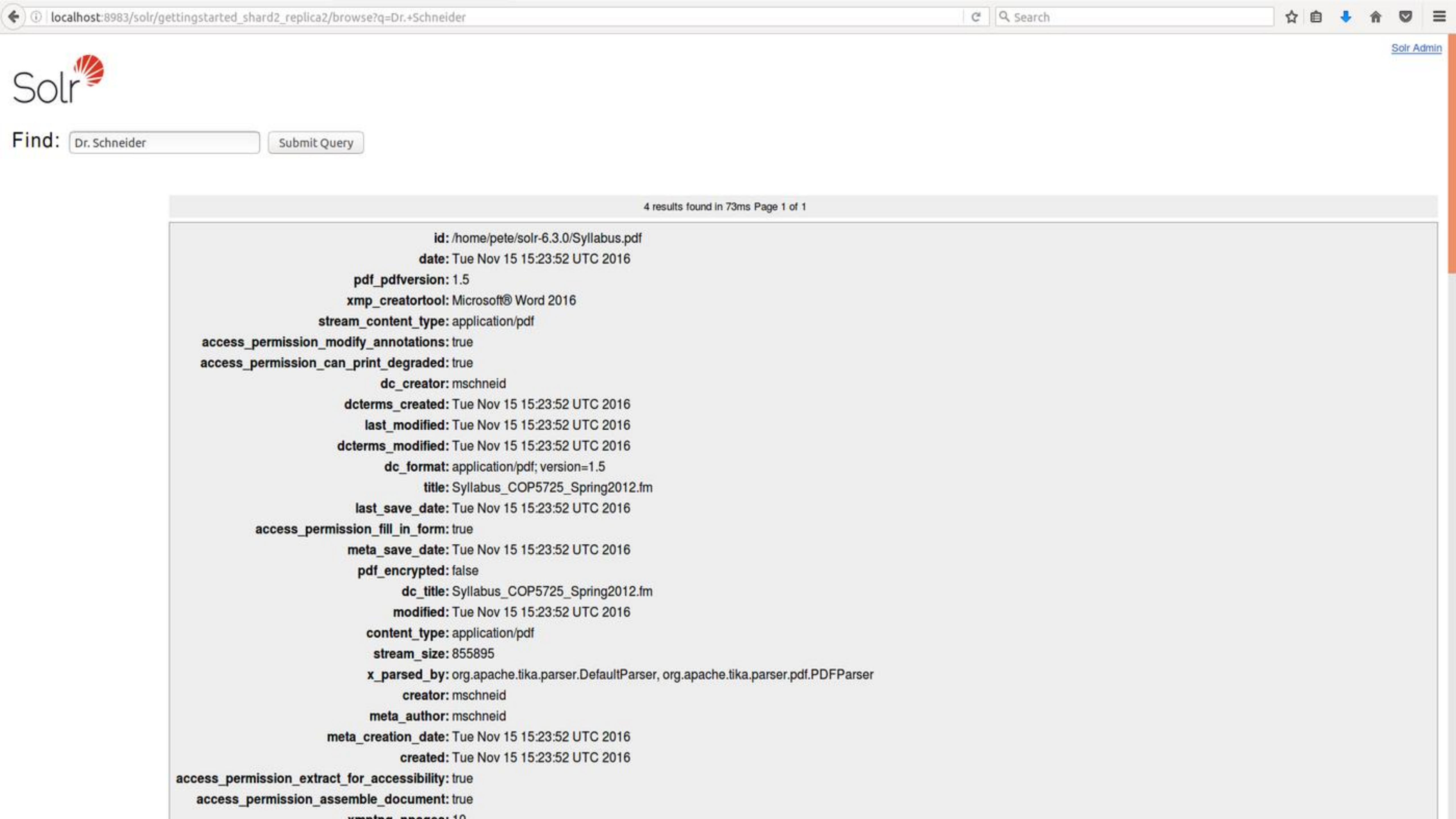
65536

JVM-Memory 21.5%

105.29 MB

490.69 MB

490.69 MB



localhost:8983/solr/gettingstarted_shard2_replica2/browse?q=Dr.+Schneider

Find: Dr. Schneider

4 results found in 73ms Page 1 of 1

id: /home/pete/solr-6.3.0/Syllabus.pdf
date: Tue Nov 15 15:23:52 UTC 2016
pdf_pdfversion: 1.5
xmp_creatortool: Microsoft® Word 2016
stream_content_type: application/pdf
access_permission_modify_annotations: true
access_permission_can_print_degraded: true
dc_creator: mschneid
dcterms_created: Tue Nov 15 15:23:52 UTC 2016
last_modified: Tue Nov 15 15:23:52 UTC 2016
dcterms_modified: Tue Nov 15 15:23:52 UTC 2016
dc_format: application/pdf; version=1.5
title: Syllabus_COP5725_Spring2012.fm
last_save_date: Tue Nov 15 15:23:52 UTC 2016
access_permission_fill_in_form: true
meta_save_date: Tue Nov 15 15:23:52 UTC 2016
pdf_encrypted: false
dc_title: Syllabus_COP5725_Spring2012.fm
modified: Tue Nov 15 15:23:52 UTC 2016
content_type: application/pdf
stream_size: 855895
x_parsed_by: org.apache.tika.parser.DefaultParser, org.apache.tika.parser.pdf.PDFParser
creator: mschneid
meta_author: mschneid
meta_creation_date: Tue Nov 15 15:23:52 UTC 2016
created: Tue Nov 15 15:23:52 UTC 2016
access_permission_extract_for_accessibility: true
access_permission_assemble_document: true

NOSQL DATABASE EXAMPLES

H·BASE
CouchDB
relax
Redis
Neo4j
the graph database
Cassandra
mongoDB
RAVENDB
membase
riak
elasticsearch.
ORACLE
Coherence
Apache
Solr

Solr Data Model

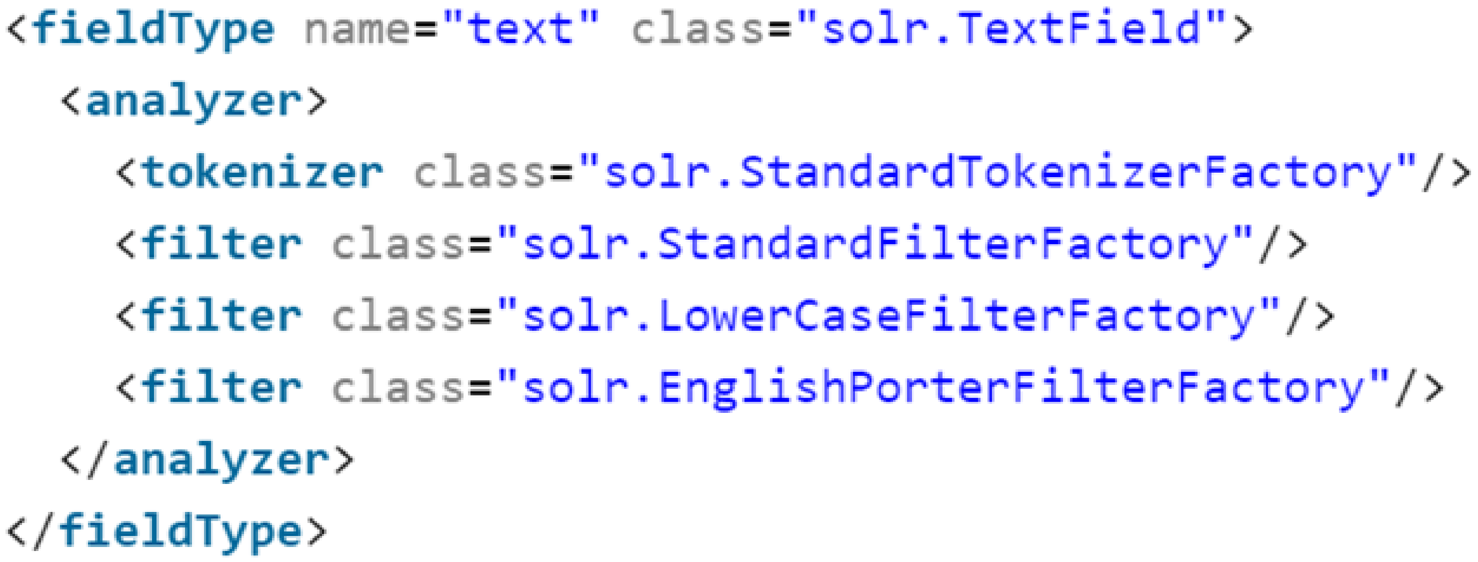
Fields

- Can be compared to a RDBMS column
- Fields can contain different kinds of data.
- Field types tell Solr how to interpret data

```
   <fields>
       <field name="id" type="string" indexed="true"
    stored="true" required="true" />
              <field name="name" type="text" indexed="true"
stored="true"/>
       …
   </fields>
```

FieldType

- Determines type of a field e.g. string, text etc.
- Associated with Lucene class
- Indexing rules are defined for FieldType



```
<fieldType name="text" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StandardFilterFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPorterFilterFactory"/>
  </analyzer>
</fieldType>
```

The Document

- Represents basic and atomic unit of information in Solr
- Composed of fields

Similarities with RDBMS record

- A document can have primary key
- A document has a structure consisting of one or more fields

Differences with RDBMS record

- Fields can be multivalued whereas a column in a database table can have only one value
- Fields either have a value or don't exist at all. There's no notion of NULL value in Solr.
- Field names can be static or dynamic, but table columns in a database must be explicitly declared in advance

The Inverted Index

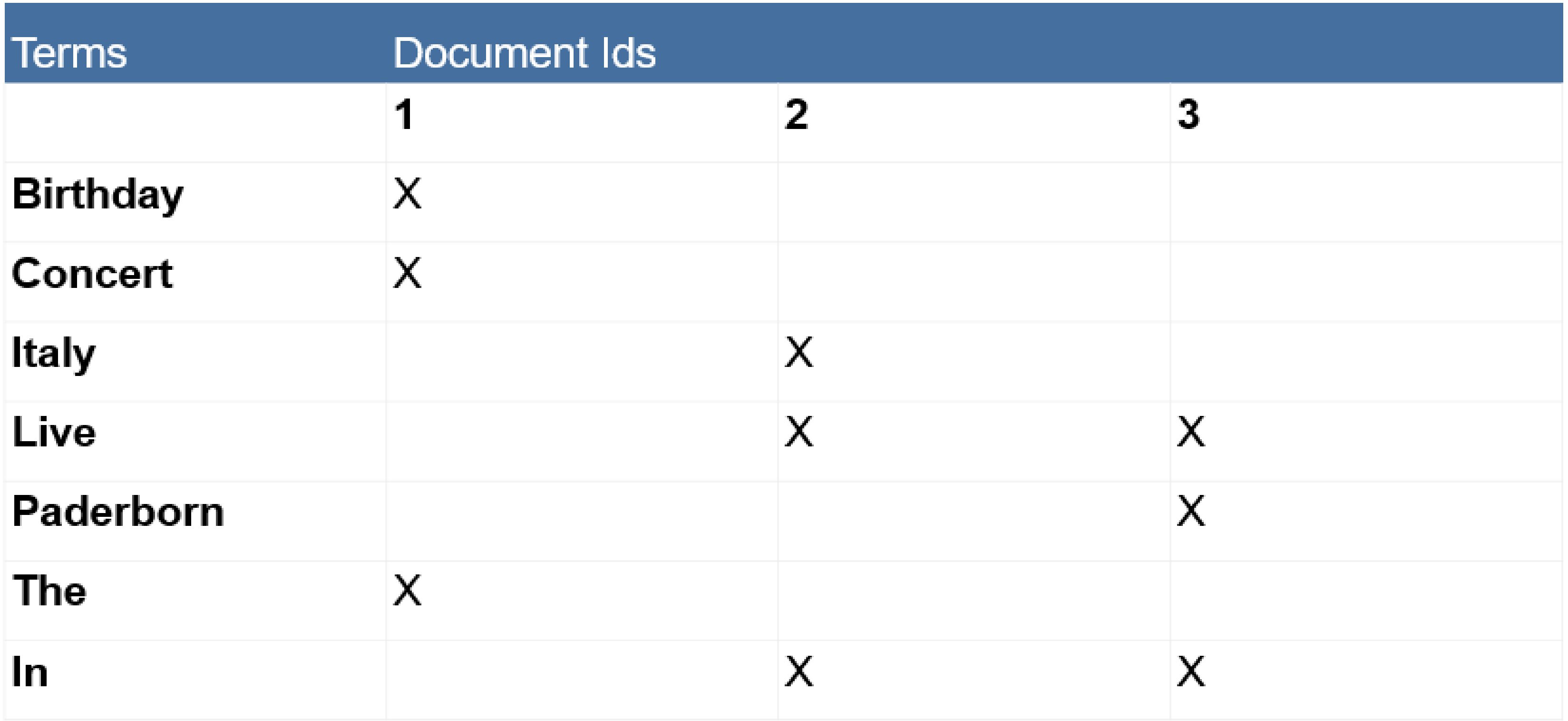
- designed and optimized to allow fast searches at retrieval time
- consists of an ordered list of all the terms that appear in a set of documents

Inverted Index example

Let’s consider 3 documents

```
{
  { "id": 1, "title":"The Birthday Concert" },
  { "id": 2, "title":"Live in Italy" },
  { "id": 3, "title":"Live in Paderborn" }
}
```

Inverted Index example(contd.)



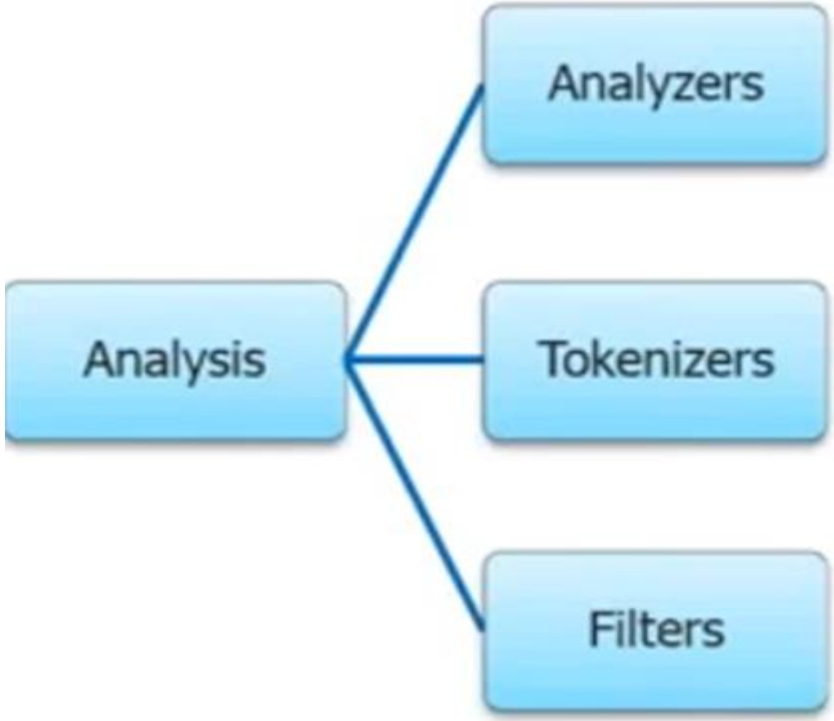
Terms	Document Ids		
	1	**2**	**3**
Birthday	X		
Concert	X		
Italy		X	
Live		X	X
Paderborn			X
The	X		
In		X	X

The Solr Core

- is a container for a specific inverted index
- The index configuration of a given Solr instance resides in a Solr core
- On the disk, Solr cores are directories, each of them with some configuration files that define features and characteristics of the core.
- A Solr application can have 0 or more cores

Text Analysis

- Three main concepts in analysis
  -Analyzers
  -Tokenizers
  -Filters



Analyzers

- Are used both during, when a document is indexed and at query time
- Same analysis process need not be used for both operations
- An analyzer examines the text of fields and generates a token stream
- May be a single class or may be composed of a series of tokenizer and filter class

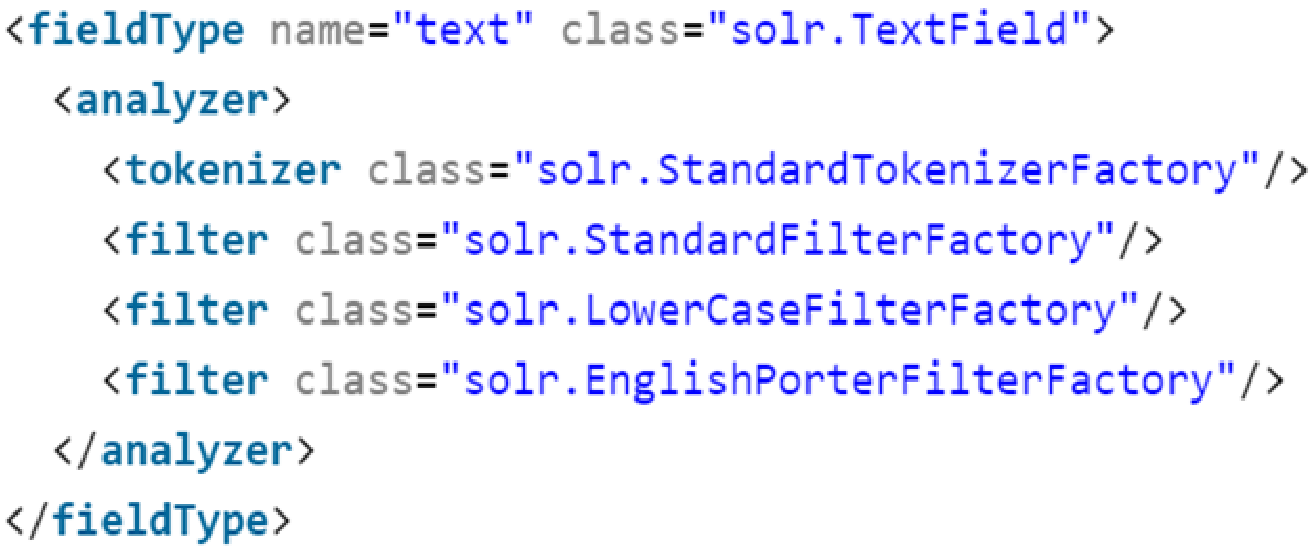
Tokenizer

- The job of a tokenizer is to break up a stream of text into tokens/terms (TokenStream objects)

- Characters in the input stream may be discarded, such as whitespace or other delimiters.

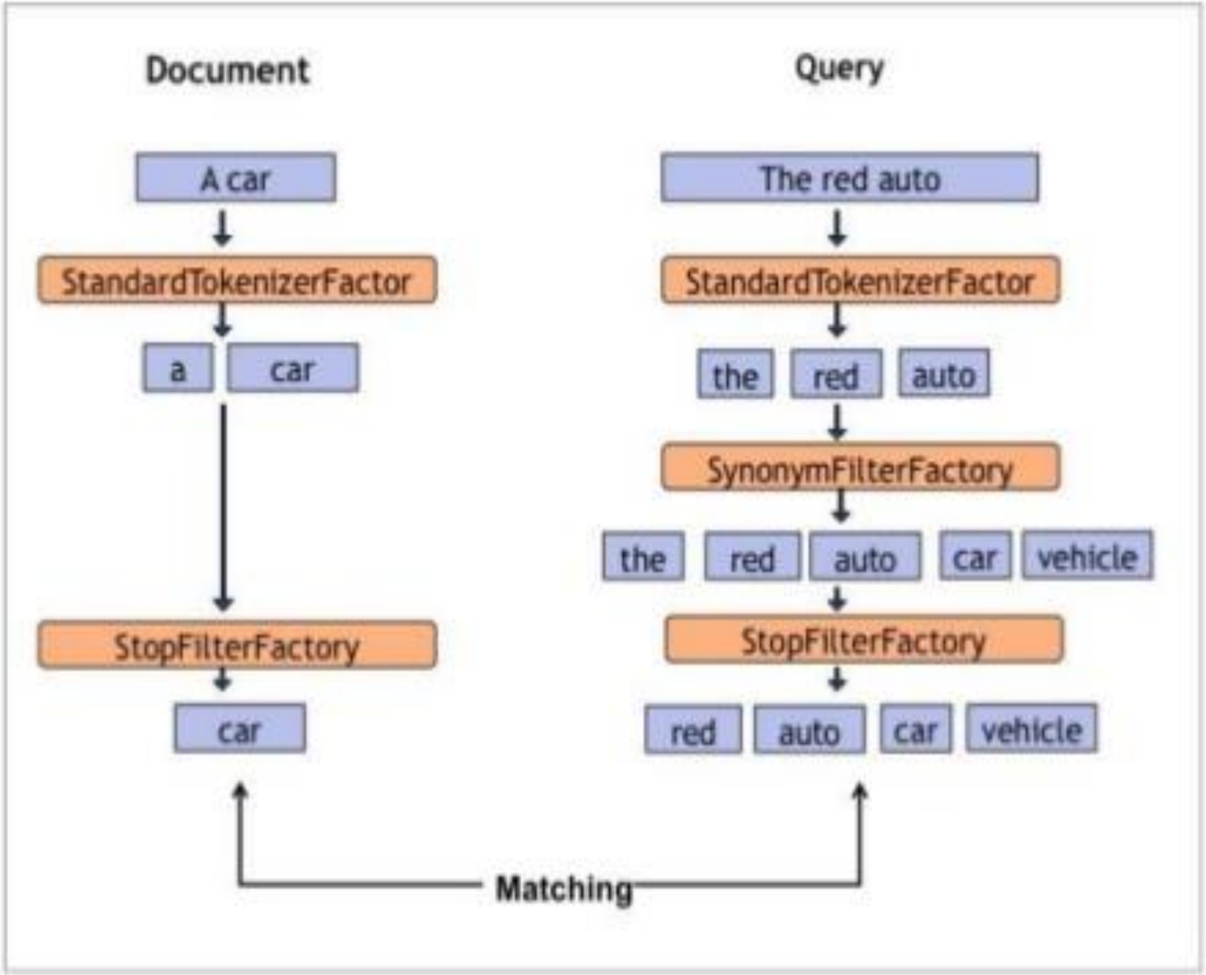
Filters

- Examine a stream of tokens and decides whether to pass it along, replace it or discard it.
- Filters consume one TokenStream and produce a new TokenStream, they can be chained one after another indefinitely



```
<fieldType name="text" class="solr.TextField">
  <analyzer>
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StandardFilterFactory"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.EnglishPorterFilterFactory"/>
  </analyzer>
</fieldType>
```

Solr Query


Document
Query
A car
StandardTokenizerFactor
a
car
StopFilterFactory
car
The red auto
StandardTokenizerFactor
the
red
auto
SynonymFilterFactory
the
red
auto
car
vehicle
StopFilterFactory
red
auto
car
vehicle
Matching

Search Document

- q
- fq
- start
- row
- sort
- fl
- wt

Solr Query Syntax

- Keyword Matching

```
title: foo
title: “foo bar”
title: foo  -title: bar
```

- Wildcard Matching

```
    title: foo*

    title: foo*bar
```

- Range Search

```
Mod_data:[20150101 TO 20160101]
```

- Boosts

```
(title:foo OR title:bar)^1.5 (body:foo OR body:bar)
```

Fuzzy & Proximity Search

- Fuzzy Search
  title: “computer”~0.5

- Proximity Search

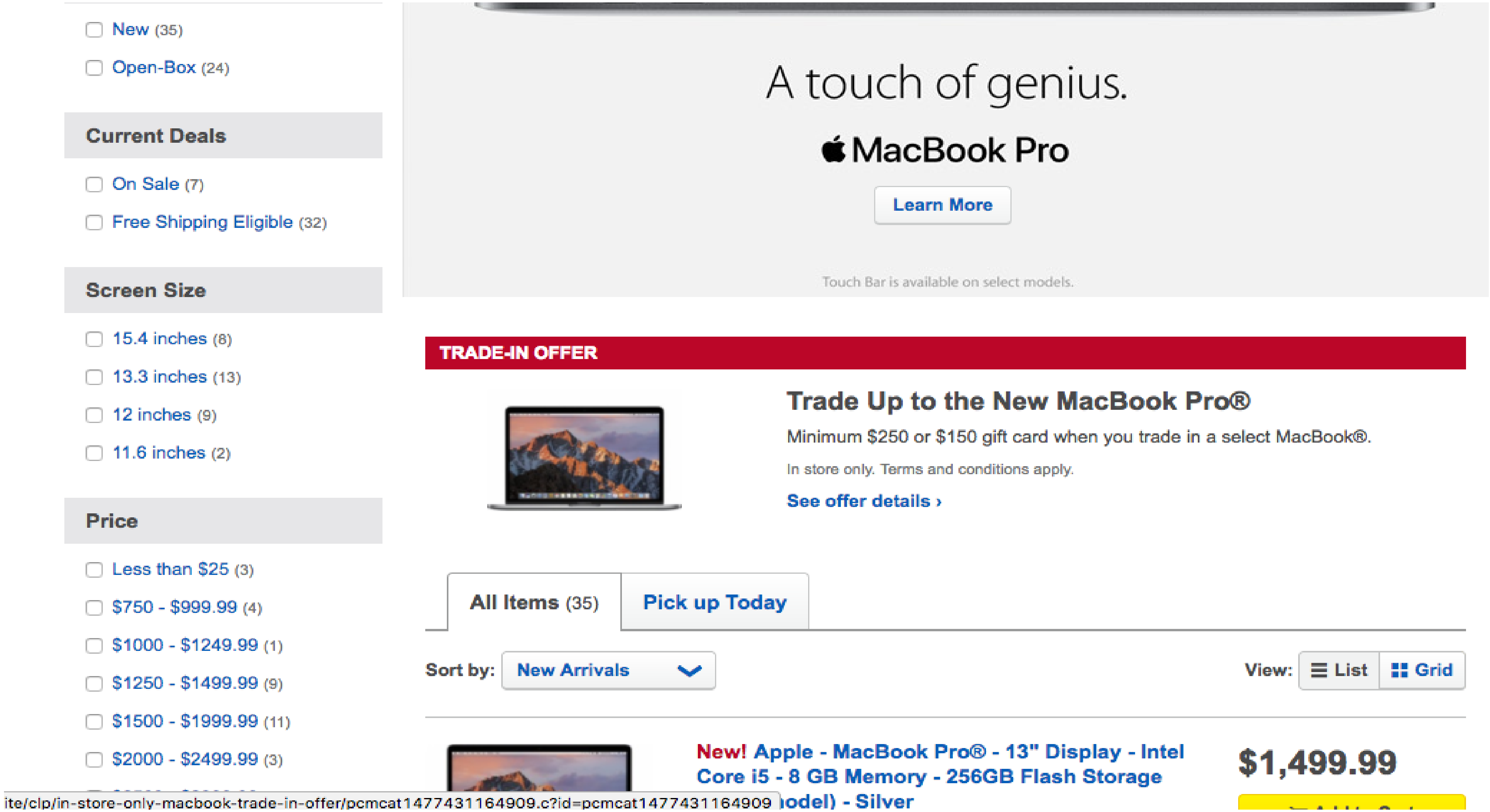
  title: “foo bar”~2

  foo abc def bar

Faceting

- facet.query
- facet.field
- facet.mincount -> f.<field.name>.facet.mincount
- facet.limit -> f.<field.name>.facet.limit
- facet.offset -> f.<field.name>.facet.offset
- facet.sort count, facet.sort index
- tagging & excluding filter
- facet.range
- facet.range.start
- facet.range.finish
- facet.range.gap

Faceting



□ New (35)

□ Open-Box (24)

Current Deals

□ On Sale (7)

□ Free Shipping Eligible (32)

Screen Size

□ 15.4 inches (8)

□ 13.3 inches (13)

□ 12 inches (9)

□ 11.6 inches (2)

Price

□ Less than $25 (3)

□ $750 - $999.99 (4)

□ $1000 - $1249.99 (1)

□ $1250 - $1499.99 (9)

□ $1500 - $1999.99 (11)

□ $2000 - $2499.99 (3)

A touch of genius.

MacBook Pro

Learn More

Touch Bar is available on select models.

TRADE-IN OFFER

Trade Up to the New MacBook Pro®

Minimum $250 or $150 gift card when you trade in a select MacBook®.

In store only. Terms and conditions apply.

See offer details ›

All Items (35) | Pick up Today

Sort by: New Arrivals

View: List | Grid

New! Apple - MacBook Pro® - 13" Display - Intel Core i5 - 8 GB Memory - 256GB Flash Storage odel) - Silver

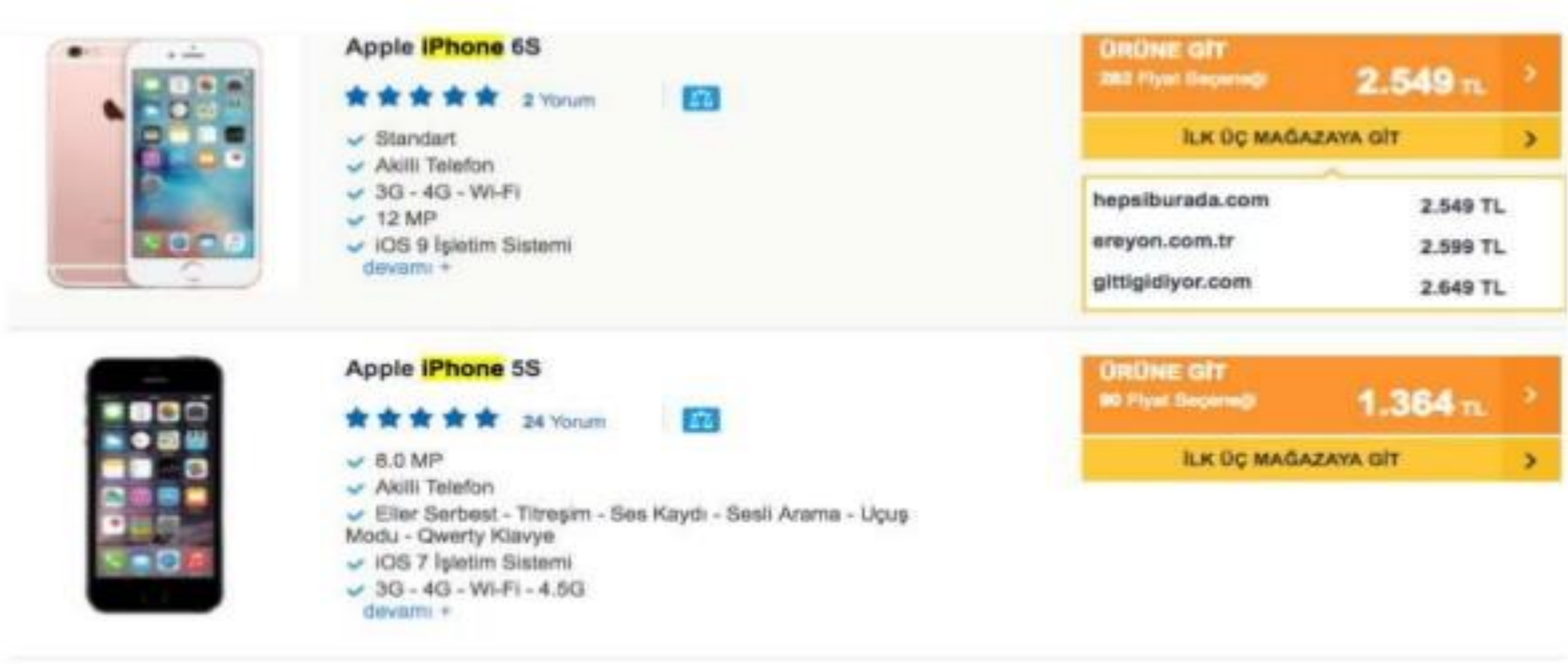
$1,499.99

ite/clp/in-store-only-macbook-trade-in-offer/pcmcat1477431164909.c?id=pcmcat1477431164909

Highlighting

```
hl = true
simple.pre
simple.post
“highlighting” {
        “37477”: {
                “Name”: [“Apple <em>iPhone</em> 6s”]
                }
}
```

Highlighting



Apple iPhone 6S

★★★★★ 2 Yorum

- ✓ Standart
- ✓ Akıllı Telefon
- ✓ 3G - 4G - Wi-Fi
- ✓ 12 MP
- ✓ iOS 9 İşletim Sistemi

devamı +

ÜRÜNE GİT
282 Fiyat Seçeneği 2.549 TL ›

İLK ÜÇ MAĞAZAYA GİT ›

hepsiburada.com	2.549 TL
ereyon.com.tr	2.599 TL
gittigidiyor.com	2.649 TL

Apple iPhone 5S

★★★★★ 24 Yorum

- ✓ 8.0 MP
- ✓ Akıllı Telefon
- ✓ Eller Serbest - Titreşim - Ses Kaydı - Sesli Arama - Uçuş Modu - Qwerty Klavye
- ✓ iOS 7 İşletim Sistemi
- ✓ 3G - 4G - Wi-Fi - 4.5G

devamı +

ÜRÜNE GİT
90 Fiyat Seçeneği 1.364 TL ›

İLK ÜÇ MAĞAZAYA GİT ›

Other Query Features

- spelling check

  ```
  spellcheck.q=Keyword&spellcheck=on
  ```

- grouping

  ```
  group=true&group.field=year
  ```

Application & API

- post command -c coreName -p port
- Rest API
- SolrJ, Spring Data Solr, or other libraries
- DataImportHandler

Application & API

EMC²
Sears
cisco
macys
shopzilla
arianespace
BOEING
ebay
Ford
Zappos.com
Fender
The Getty
hp
The Motley Fool
Fool.com
FINRA
Information Builders
IRON MOUNTAIN
The Street
TASER
theguardian
BIGLOBE
NEC
intuit
verizon
eHarmony
QUALCOMM
Raytheon
salesforce.com
Advance Auto Parts
Keep the wheels turning.
PTC
STANDARD & POOR'S
Taylor & Francis
Taylor & Francis Group
sensis.com.au
Smithsonian
at&t

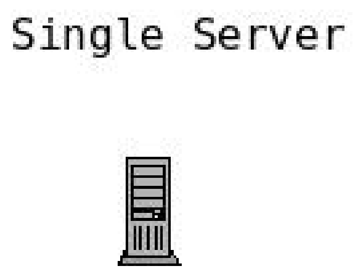
And many many many many more...!

Scalability

- Designed to work under heavy search traffic
- Able to quickly find results with indexed searches
- Is very flexible depending on how many indexes you have
- Can be easily scaled to the user’s needs
- Can use a variety of scaling techniques (horizontal, vertical, replication, sharding, and cloud)
- Able to handle high query volume, and large index size

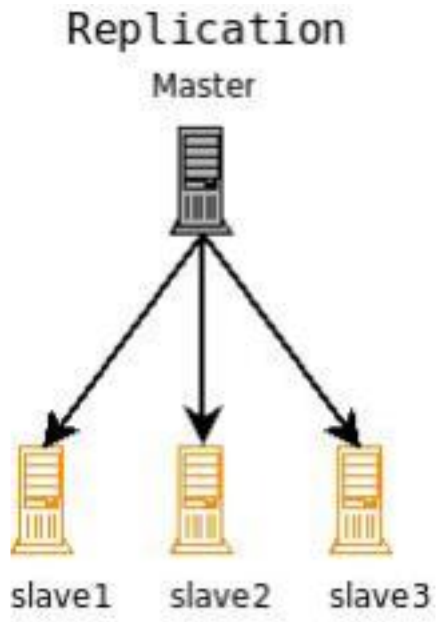
Single Server

- Best to maximize a single server before expanding horizontally or vertically
- Manage index through stop words and term frequencies
- Make use of cache and optimize it



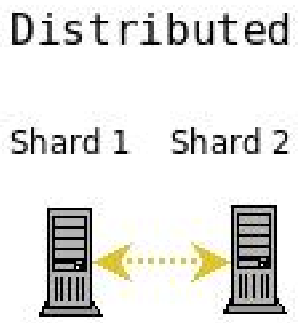
Replication

- Used to handle high query volume
- Uses slaves to help search for indexes
- Used to scale horizontally
- Master takes snapshots and distributes new images



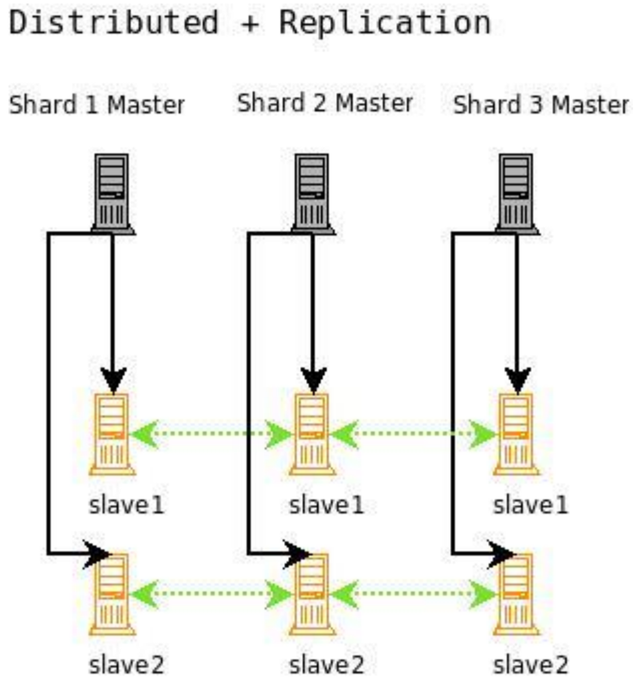
Sharding

- Used to handle a large amount of indexes
- Each system performing a search
- Suffers from excessive chatter
- Not ideal large scale scaling
- Ideal to balance requests per shard



Replication+Sharding

- Used when the index is too large for a machine, as a high query volume.
- Master shards do not communicate with each other
- Allows for fault tolerance using load balancing software

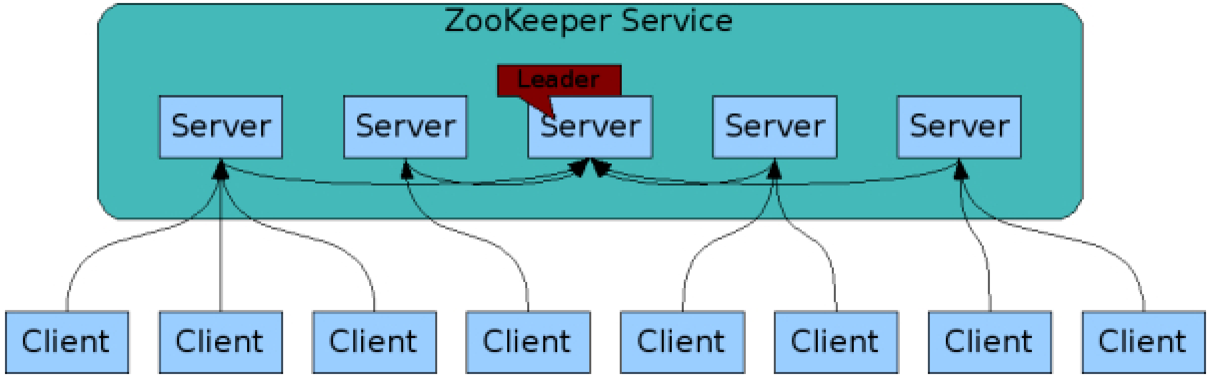


Solr Cloud

- Contains high fault tolerance
- High availability
- Central configuration for the entire cluster
- Automatic load balancing and fail-over for queries
- ZooKeeper integration for cluster coordination and configuration
- Flexible distributed search and indexing

Solr Cloud ZooKeeper

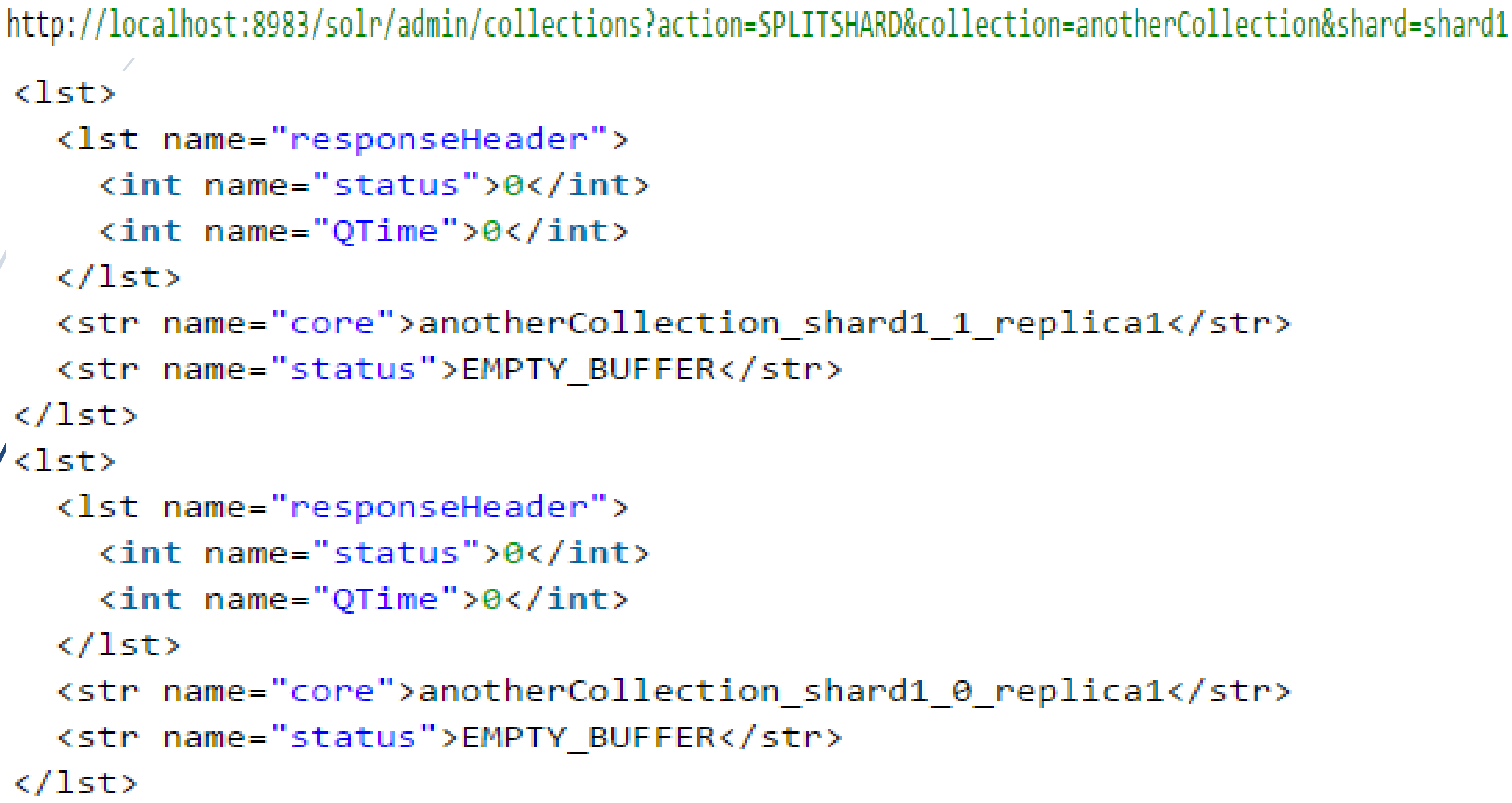
- Used to manage nodes for SolrCloud
- Keeps track of changes made
- Needs 2xF+1 machines, to ensure requests can be served even on failure



Shards and Indexing Data in SolrCloud

- Automatic document distribution and indexing
- Can use the router to hash documents to shards, such as “q=solr&_route_=IBM!”
- Able to split shards even after the initial declaration of shards using CollectionAPI

Collection API Shard Splitting



```
http://localhost:8983/solr/admin/collections?action=SPLITSHARD&collection=anotherCollection&shard=shard1
```

```
<lst>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">0</int>
  </lst>
  <str name="core">anotherCollection_shard1_1_replica1</str>
  <str name="status">EMPTY_BUFFER</str>
</lst>
<lst>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">0</int>
  </lst>
  <str name="core">anotherCollection_shard1_0_replica1</str>
  <str name="status">EMPTY_BUFFER</str>
</lst>
```

Fault Tolerance

Write Tolerance

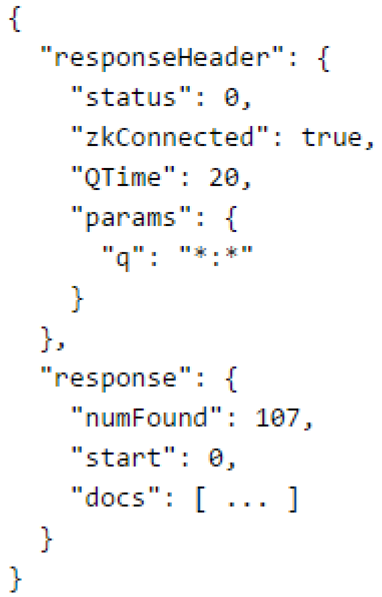
- Node uses leader to update shards
- Nodes keep track of updates with Transaction Log

Read Tolerance

- Only needs one available replica
- Can read partial results

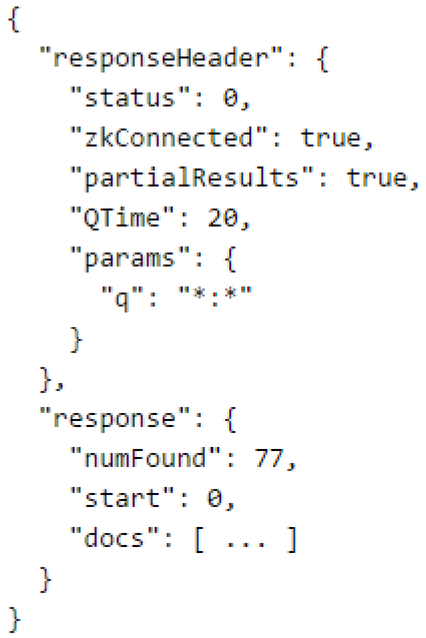
Read Fault Tolerance

Fault Tolerance



```
{
  "responseHeader": {
    "status": 0,
    "zkConnected": true,
    "QTime": 20,
    "params": {
      "q": "*:*"
    }
  },
  "response": {
    "numFound": 107,
    "start": 0,
    "docs": [ ... ]
  }
}
```

Partial Results



```
{
  "responseHeader": {
    "status": 0,
    "zkConnected": true,
    "partialResults": true,
    "QTime": 20,
    "params": {
      "q": "*:*"
    }
  },
  "response": {
    "numFound": 77,
    "start": 0,
    "docs": [ ... ]
  }
}
```

References

- https://wiki.apache.org/solr/
- https://www.packtpub.com/mapt/book/Big-Data-and-Business-Intelligence/
- https://lucidworks.com/blog/2009/09/02/scaling-lucene-and-solr/
- http://zookeeper.apache.org/
- https://cwiki.apache.org/confluence/display/solr/Apache+Solr+Reference+Guide
- http://www.solrtutorial.com/solrj-tutorial.html
- http://www.slideshare.net/erikhatcher/solr-application-development-tutorial
- http://www.edureka.co/apache-solr-self-paced

Thank You