

# TUTORIAL 3

---

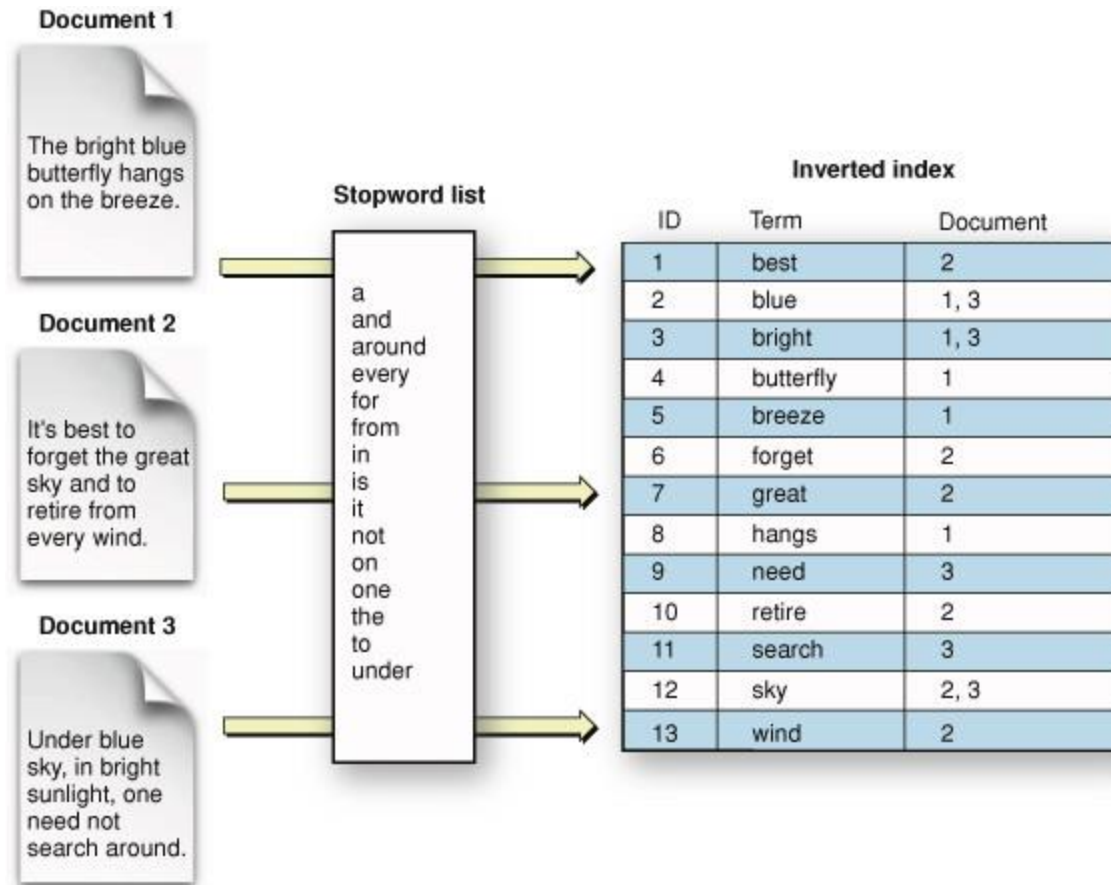
- MapReduce Examples
- Apache Solr

by Suchitra Jayaprakash  
suchitra@cmi.ac.in

# MAPREDUCE

- MapReduce is a programming paradigm with two phases:
  - Map
    - Shuffle & Sort
  - Reduce

# Inverted Index with MapReduce



# Implementation

- `WORD_RE = re.compile(r"[a-zA-Z]{2,}\b")`
- `class MRInvertedIndex(MRJob):`
  - `def mapper(self, _, line):`
    - `## getting input file name`
    - `filepath = os.environ['map_input_file']`
    - `filename=filepath.split('/')[-1]`
    - `for word in WORD_RE.findall(line):`
      - `yield (word.lower(), filename)`
  - `def reducer(self, word, filenames):`
    - `yield (word, ",".join(list(set(filenames))))`

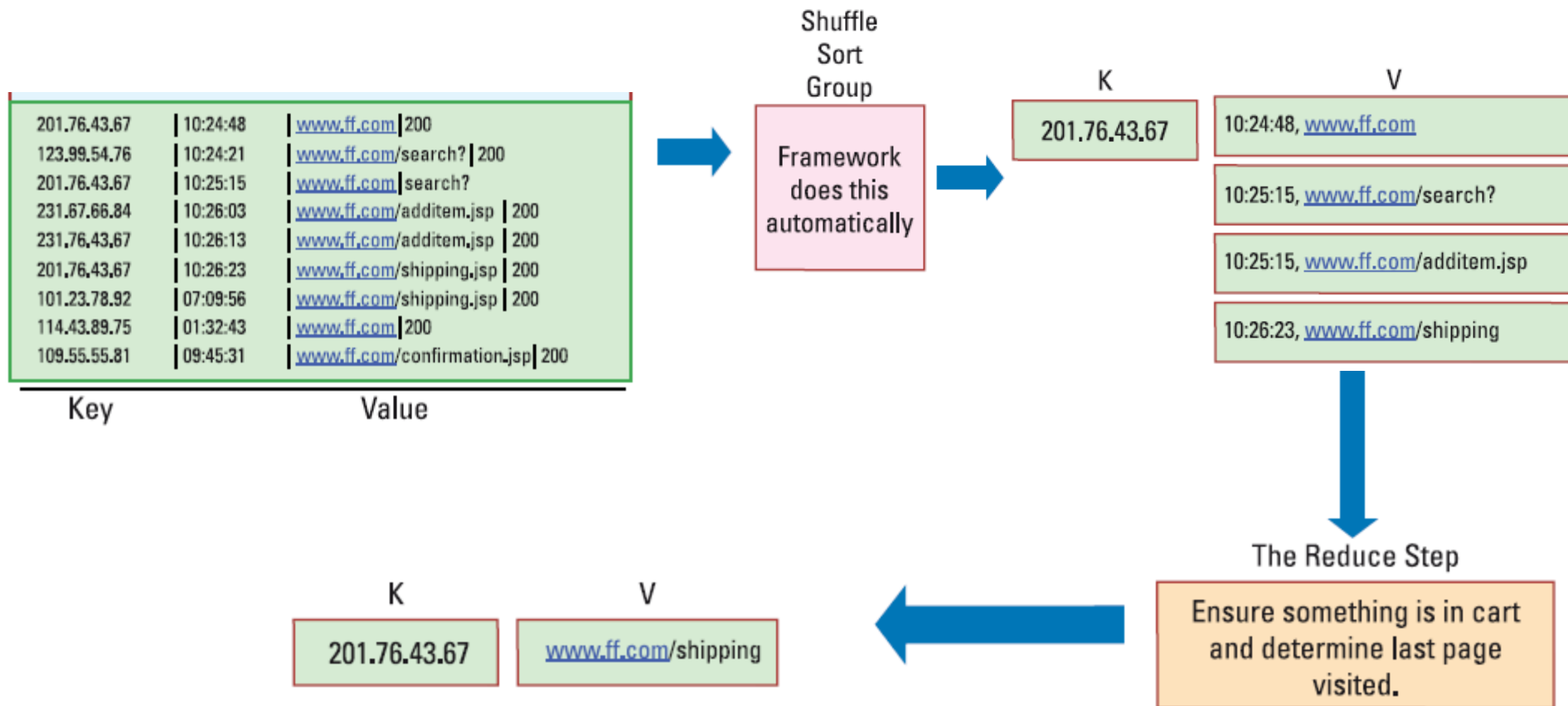
# Capture useful insights from Log data

- An ecommerce web site collects click stream data as log file.

201.76.43.67	10:24:48	<a href="http://www.ff.com">www.ff.com</a>   200
123.99.54.76	10:24:21	<a href="http://www.ff.com/search?">www.ff.com/search?</a>   200
201.76.43.67	10:25:15	<a href="http://www.ff.com">www.ff.com</a>   search?
231.67.66.84	10:26:03	<a href="http://www.ff.com/additem.jsp">www.ff.com/additem.jsp</a>   200
231.76.43.67	10:26:13	<a href="http://www.ff.com/additem.jsp">www.ff.com/additem.jsp</a>   200
201.76.43.67	10:26:23	<a href="http://www.ff.com/shipping.jsp">www.ff.com/shipping.jsp</a>   200
101.23.78.92	07:09:56	<a href="http://www.ff.com/shipping.jsp">www.ff.com/shipping.jsp</a>   200
114.43.89.75	01:32:43	<a href="http://www.ff.com">www.ff.com</a>   200
109.55.55.81	09:45:31	<a href="http://www.ff.com/confirmation.jsp">www.ff.com/confirmation.jsp</a>   200

- Identify key factors behind abandoned shopping carts

# MapReduce Implementation



# Build a search engine

- Over 10 billion products are sold on an ecommerce web site.
- Company wants to build search tool on their site.

# WHAT IS SOLR ?

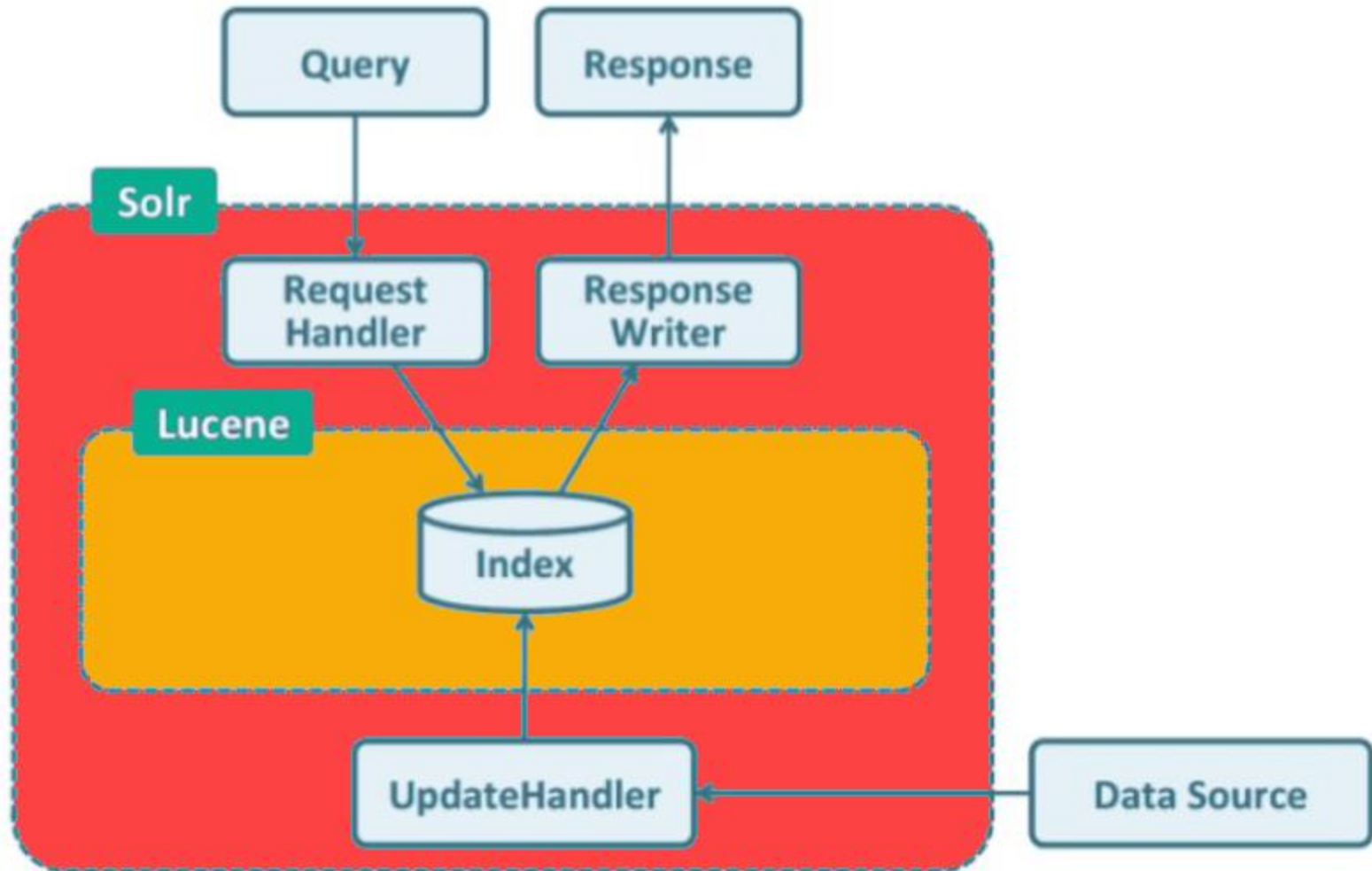
- Open-source REST-API based Enterprise Real-time Search and Analytics Engine Server.
- Highly scalable search applications
- Ready to deploy
- Built using Apache Lucene Framework , Java
- Optimized to search large volumes of text-centric data
- Document-based NoSQL Data Store
- Main Functionality – Indexing & Searching



# SOLR FEATURES

- Full text search
- Faceted Navigation
- Spell check
- Hit highlighting
- Relevant result
- Recommendation
- Geo-Spatial Search
- supports Distributed and Cloud Technology
- Built in Authentication & authorisation
- Learning to rank

# SOLR ARCHITECTURE



# INSTALL SOLR LOCALLY

- Java Runtime Environment version 8 or greater
- Download Solr from lucene site and unzip for installation.
- Solr Instance is an instance a Solr running in the JVM
- Start/Stop Apache solr
  - bin/solr start
  - bin/solr stop
- Access the Solr Admin User Interface
  - <http://localhost:8983/solr/>

•

# CREATE CORE

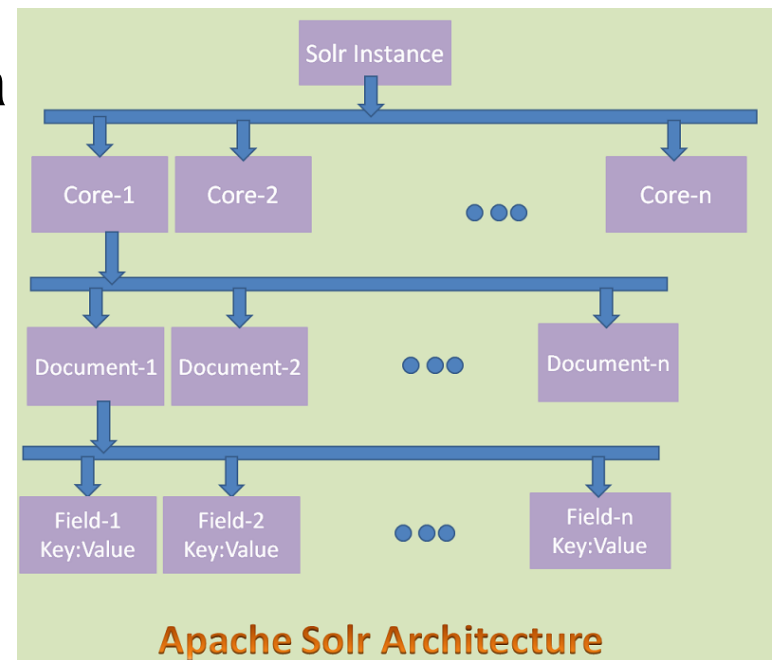
- A Solr Core is a running instance of a Lucene index that contains all the Solr configuration files required to use it.

Core = an instance of Lucene Index + Solr configuration

- We need to create a Solr Core to perform operations like indexing and analyzing
- `solr create_core -c XXXX -p 8983`

# INDEXING DATA

- Solr can ingest many types of files like .csv .json, .xml, .html,.pdf
- We can add data to Solr index via
  - Solr Web Interface.
  - Client APIs like Java, Python, etc.
  - post tool.



- Use Admin console or POST
- command to upload document

- `java -Dc=XXXX -Durl=http://localhost:8983/solr/XXXX/update/ -Dtype=application/xml -jar ../example/exampledocs/post.jar "../example/films/films.xml"`

# CONFIGURATION FILES

- solr.xml - server instance configurations
- core.properties - core configurations such as names, locations and files in the core
- conf/solrconfig.xml - core configurations for field guessing, directories, query settings, spell checking, keyword highlighting and query response formats
- conf/managed-schema - core configurations for field processing

# SEARCH IN ADMIN UI

- Open Solr Admin Console: <http://localhost:8983/solr/>
- Select the core module
- Click on “Query” from the left navigation
- Enter search query in q text box & df (default field)
- Click on “Execute Query” button
- It retrieves matching document from selected core

# SEARCH ....

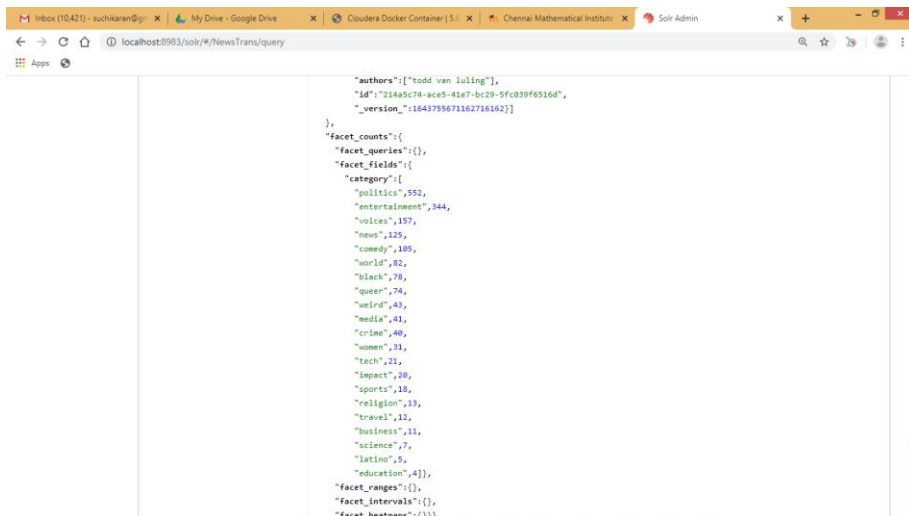
- You can also do search using rest api call
  - [http://localhost:8983/solr/<corename>/select?indent=on&q=\\*&wt=json](http://localhost:8983/solr/<corename>/select?indent=on&q=*&wt=json)
  - <http://localhost:8983/solr/<corename>/select?q=black&mlt=true&mlt.fl=genre&mlt.mindf=1&mlt.mintf=1&fl=id,score&df=genre>
  - <http://localhost:8983/solr/<corename>/spell?spellcheck=true&qt=spellchecker&spellcheck.accuracy=0.5&spellcheck.collate=true&q=Cime>



# Sample Output

## Faceting :

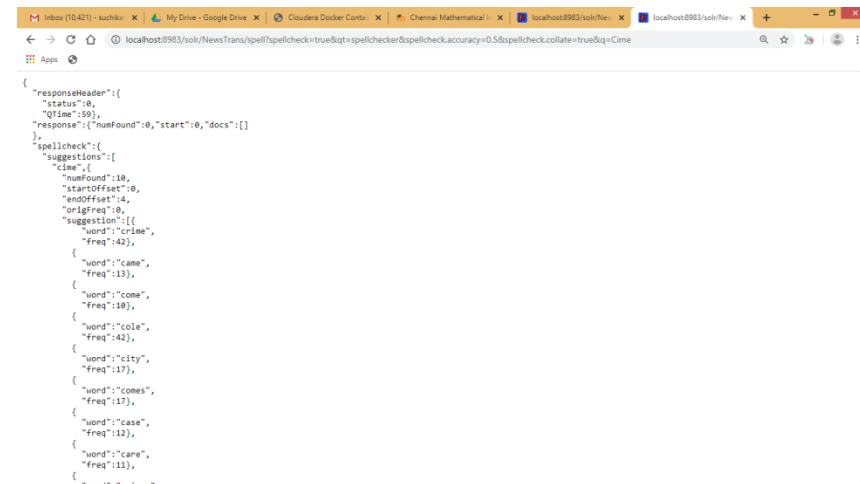
<http://localhost:8983/solr/NewsTrans/select?facet.field=category&facet=on&q=%22crime%22>



```
{
  "authors":["todd van luling"],
  "id":"72445c74-ace5-41e7-bc29-5f0839f8516d",
  "_version_":1643755671162716162]
},
"facet_counts":{
  "facet_queries":{},
  "facet_fields":{
    "category":{
      "politics",552,
      "entertainment",344,
      "voices",157,
      "news",125,
      "comedy",105,
      "world",82,
      "black",78,
      "queer",74,
      "weird",43,
      "media",41,
      "crime",40,
      "women",31,
      "tech",21,
      "impact",20,
      "sports",18,
      "religion",13,
      "travel",12,
      "business",11,
      "science",7,
      "latino",5,
      "education",4}},
  "facet_ranges":{},
  "facet_intervals":{},
  "facet_heatmaps":{}}
```

## Spellcheck :

<http://localhost:8983/solr/NewsTrans/spell?spellcheck=true&qt=spellchecker&spellcheck.accuracy=0.5&spellcheck.collate=true&q=Cime>



```
{
  "responseHeader":{
    "status":0,
    "QTime":99,
    "response":{"numFound":0,"start":0,"docs":[]
  }},
  "spellcheck":{
    "suggestions":{
      "crime":{
        "numFound":10,
        "startOffset":0,
        "endOffset":4,
        "origFreq":0,
        "suggestion":[{"
          "word":"crime",
          "freq":42},
          {
            "word":"came",
            "freq":13},
          {
            "word":"come",
            "freq":10},
          {
            "word":"cola",
            "freq":42},
          {
            "word":"city",
            "freq":17},
          {
            "word":"comes",
            "freq":17},
          {
            "word":"case",
            "freq":12},
          {
            "word":"care",
            "freq":11},
          {
            "word":"cime",
            "freq":11}
        ]}
      }
    }
  }
}
```

# Quiz 3

- Identity mapper is used to do "nothing". It converts the input to output "as is". Assume we have a long list of words. Can you suggest where identity mapper on this word list could be used?
- A) Sorting
- B) Searching
- C) Aggregating
- D) All of the above
- E) None of the above

THANK YOU