Full-text Search with NoSQL Technologies

NoSQL Roadshow Berlin

Kai Spichale

25.04.2013

About me

- Kai Spichale
- Software Engineer at adesso AG
- Author in professional journals, conference speaker

- adesso is among Germany‘s top IT service providers
- Consulting and software development focus
- More than 1,000 members of staff
- Some of the most important customers are Allianz, Hannover Rück, Westdeutsche Lotterie, Zurich Versicherung, DEVK, and DAK

adesso

Motivation

NoSQL

- Exponential data growth
- Semi-structured data
- More connections
- 80 percent of business-relevant information is in unstructured form

Search

- Shift in data access:
  - More full-text search
  - Higher user expectations
- Keyword search and link directories become impractical

Agenda

- **Lucene full-text search**
- NoSQL:
  - Architectural drivers
  - MongoDB
  - Neo4j
  - Apache Cassandra
  - Apache Hadoop
- Summary

Full-text search

- Techniques for searching documents in collections

- grep-like naive approach:
  - Serial scanning is slow
  - No negation
  - No distinction between phrase and keyword search

- Build inverted index
  - Term → Document
  - Contains references to documents for each token

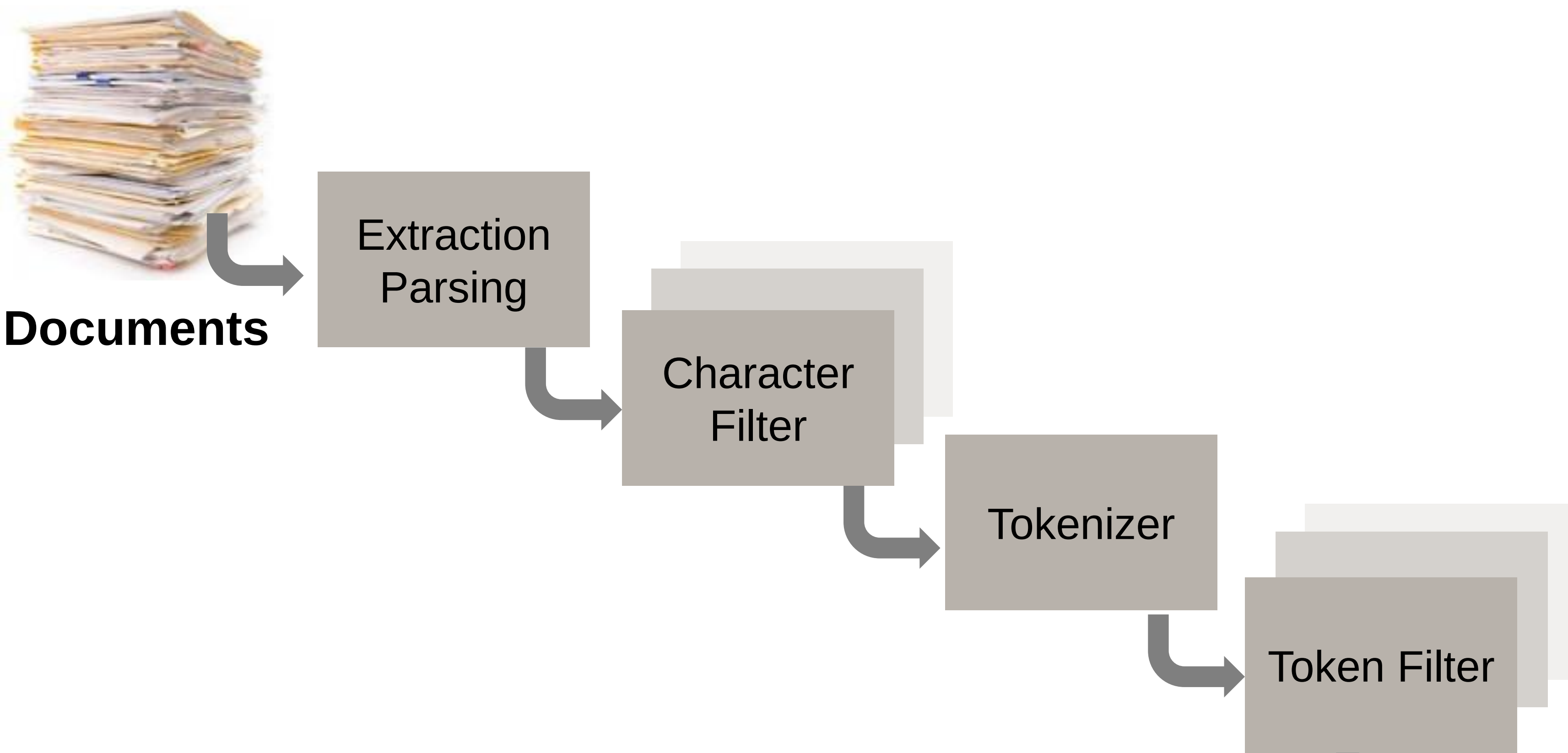
Apache Lucene

- Java lib for full-text searches
- De facto standard for open source software

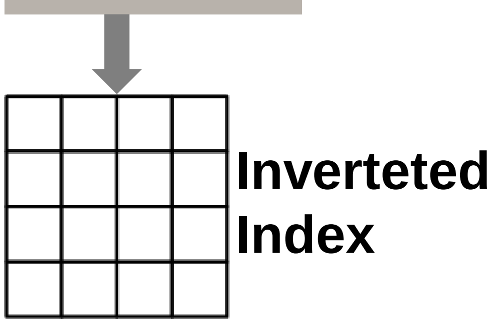
- Attributes:
  - Application-agnostic
  - Scalable, high performance

- Features:
  - Ranked searching
  - Multiple query types, faceting
  - Sorting
  - Multi-Index searching

Text Analysis



de.GermanAnalyzer:
StandardTokenizer > StandardFilter
> LowerCaseFilter > StopFilter > GermanStemFilter



adesso

Text Analysis

Eat your own dog food.

First come, first served.

The exception proves the rule.

Stop word List

a
and
around
every
for
from
in
is
it
not
on
one
the
to
under

ID	Term	Document
1	come	2
2	dog	1
3	eat	1
4	exception	3
5	first	2
5	food	1
6	own	1
7	prove	3
8	rule	3
9	serve	2
10	your	1

Query types

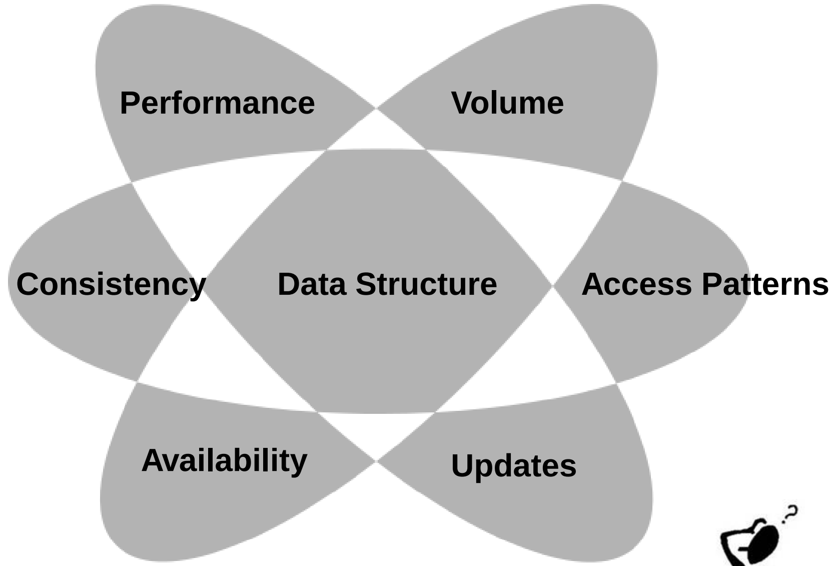
Type	Example
Term (MUST, MUST_NOT, SHOULD)	+adesso –italy
Phrase	„foo bar“
Wildcard	fo*a?
Fuzzy	fobar~
Range	[A TO Z]

Agenda

- Lucene full-text search
- **NoSQL**:
  - Architectural drivers
  - MongoDB
  - Neo4j
  - Apache Cassandra
  - Apache Hadoop
- Summary

~~**One size fits all approach**~~

- Which NoSQL store satisfies our requirements best?
- Is full-text search supported?



NoSQL and Search

Let‘s take a closer look on:

- MongoDB
- Neo4j
- Apache Cassandra
- Apache Hadoop

Document-oriented Databases

- Designed for storing and retrieving documents
- Semi-structured content such as BSON documents

```
{  "_id"         : ObjectId(„42"),
   "firstname" : "John",
   "lastname"  : "Lennon",
   "address"   : { "city"   : "Liverpool",
                   "street" : "251 Menlove Avenue“ }
}
```

MongoDB

- Supports ad-hoc CRUD operations

```
db.things.find({firstname:"John"})
```

- Server-side execution of JavaScript
- Aggregations, MapReduce

- Simple keyword search with multikey indexes:
  - Index array content as separate entries

```
{  article   : “some long text",
  _keywords : [ “some" , “long" , “text“]
}
```

MongoDB

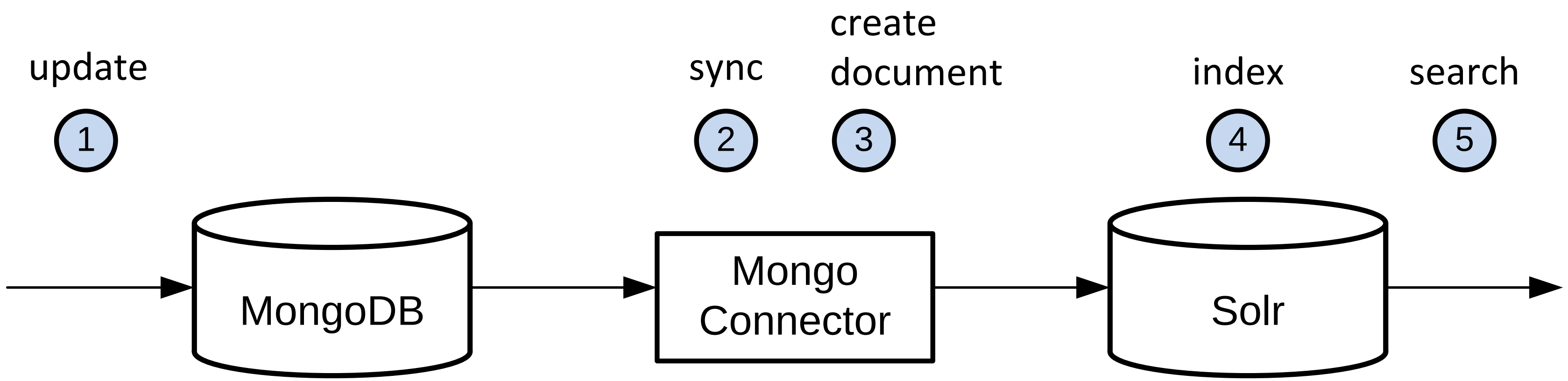
- Version 2.4 supports text indexes
- Language-specific stemming based on Snowball

```
db.foo.runCommand(“text“, {search: “adesso –italy”, language: “english”})
```

- Still a beta feature

MongoDB

- Mongo Connector integrates MongoDB with another system (backup MongoDB cluster, Solr, elasticsearch,)
- System architecture with separate search engine possible



Choosing the Right Approach

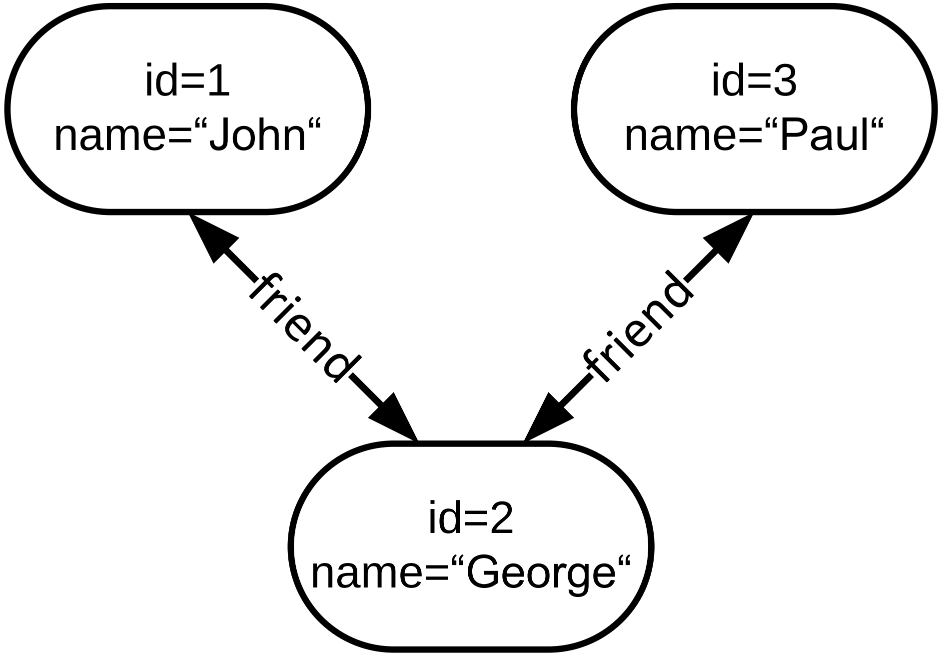
MongoDB	MongoDB + Search Engine	Search Engine
✓ No result set merging	✓ Full-text search with faceting	✓ No result set merging
✓ Complex queries with aggregations	✓ Complex queries with aggregations	✓ Full-text search with faceting
❑ Simple text search (but experimental text index)	❑ Result set merging ❑ Increased complexity (ops, dev)	❑ Maybe no aggregations

Graph Databases

- Stored data is represented as graph structures
  - Nodes
  - Edges (Relationships)
  - Properties

- Universal datamodel
- Traversing

- Example: Neo4j



Neo4j

- Traversing
  - Visiting nodes by following relationships
  - Breadth- and depth-first traversing
  - Gremlin, Cypher

```
START john=node:peoplesearch(name='John')
MATCH john<-[:friend]->afriend RETURN afriend
```

Result = George

Neo4j

- Database itself is a natural index consisting of its edges and nodes
  - Example: „name“, „city“

```
personRepository.findByPropertyValue("name", "John");
```

- Auto indexing keeps track of property changes

Neo4j

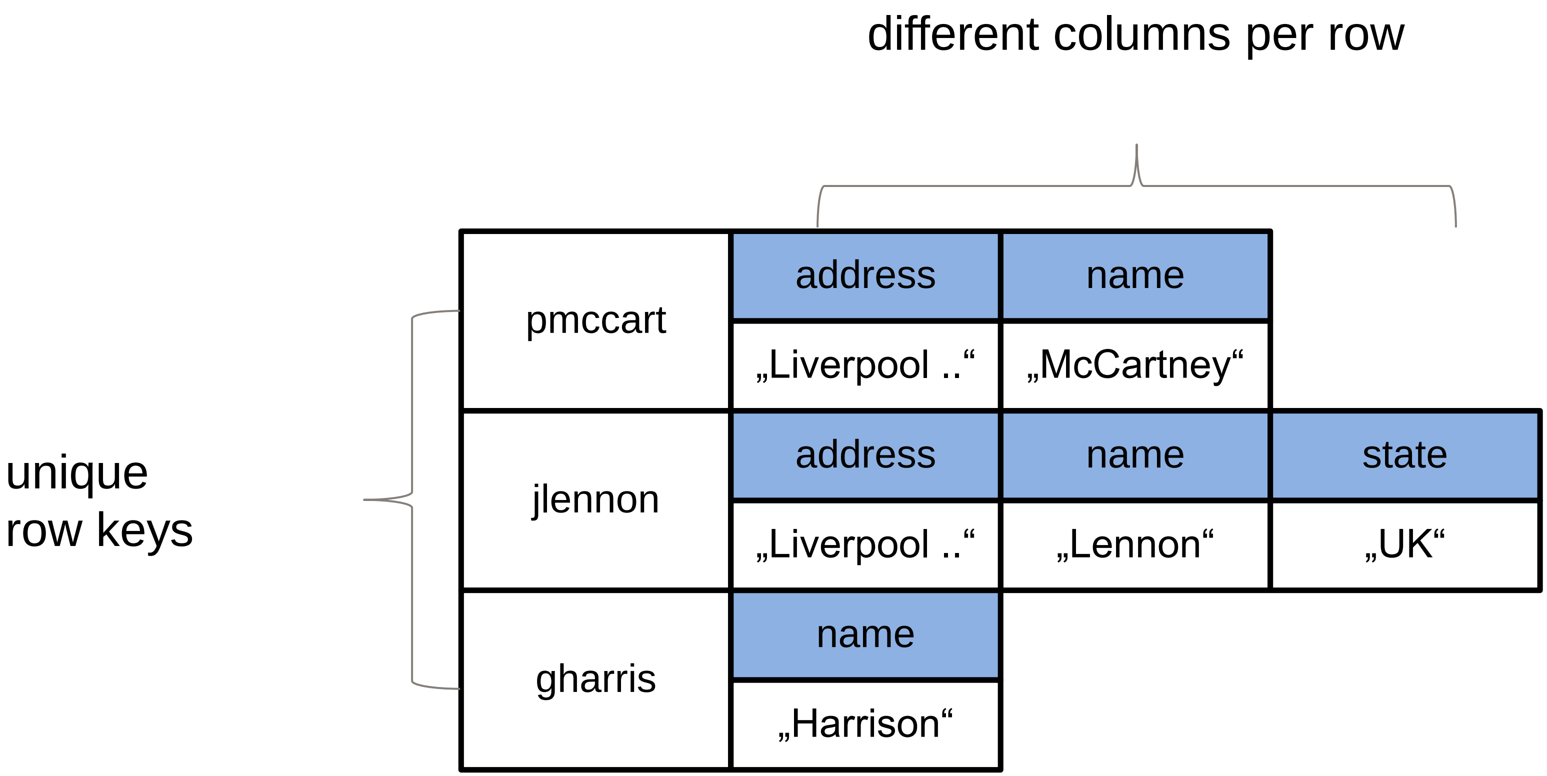
- The default separate index engine used is Apache Lucene

```
@NodeEntity
class Person {
@Indexed(indexName="peoplesearch", indexType=IndexType.FULLTEXT)
private String name;
..
}
```

```
Index<PropertyContainer> index = template.getIndex("peoplesearch");
index.query("name", "Jo*");
```

Wide Column Store

- Google BigTable: *„a sparse, distributed multi-dimensional sorted map“*
- Data is organized in rows, column families, and columns
- Ideal for sharding (horizontal partitioning)



Apache Cassandra

- BigTable clone
- Distributed Hash Table (Amazon Dynamo)
- Eventual consistency (configurable levels)

- Cassandra Query Language (CQL) = SQL dialect without joins

```
SELECT name FROM user WHERE firstname=„John“;
```

- Hadoop integration

Apache Cassandra

- Solandra = Solr using Cassandra as backend

- DataStax Enterprise Search
  - One local Solr instance per Cassandra node
  - Integration is based on secondary index API
  - CQL supports Solr Queries

```
SELECT title FROM solr WHERE solr_query=‘name:jo*';
```

  - Cassandra’s ring information is used to construct Solr distributed search queries

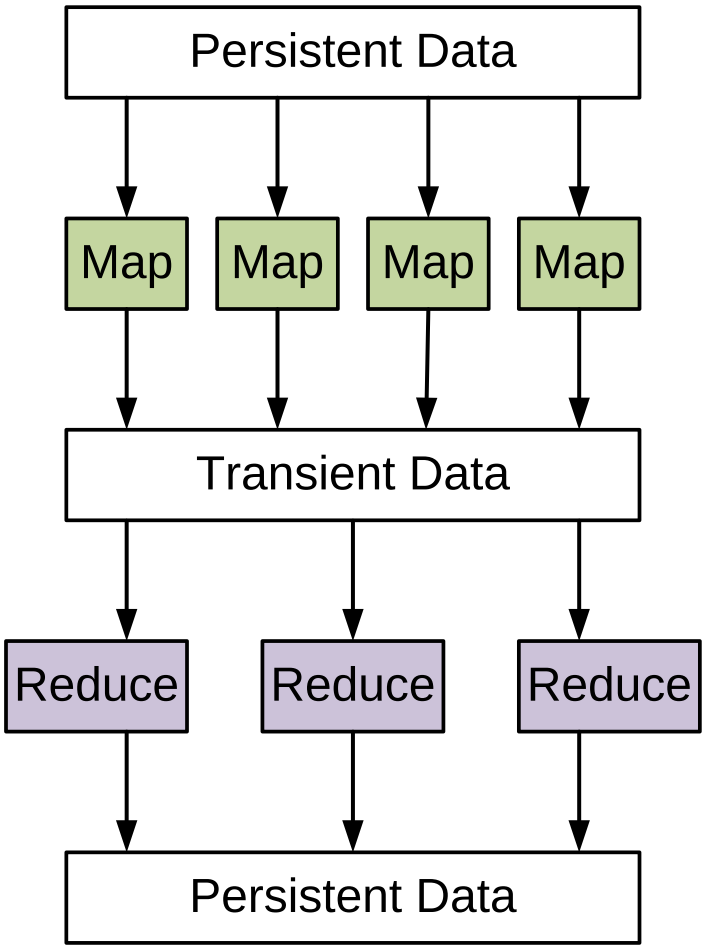
Apache Hadoop

- Hadoop:
  - Framework for distributed processing of large data sets in computer clusters
  - Distributed filesystem + MapReduce implementation

- Scalable and reliable platform of a comprehensive data analysis ecosystem

Hadoop MapReduce

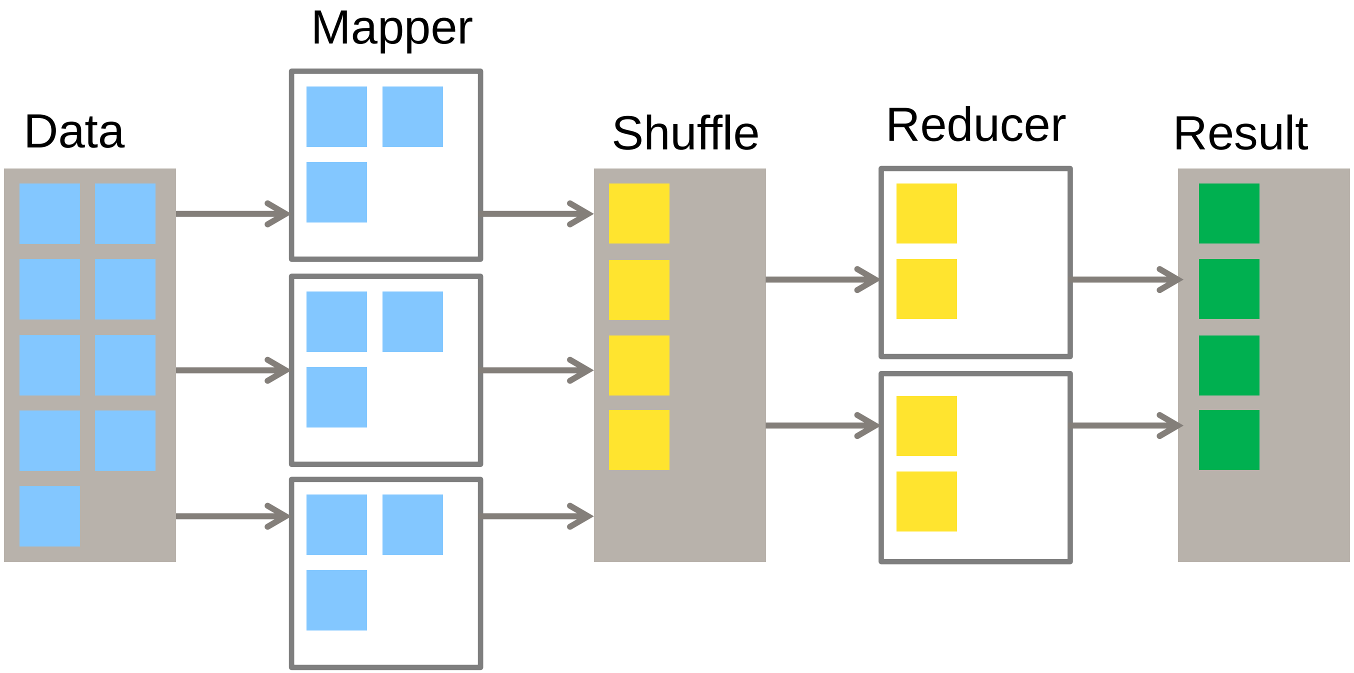
- **Map Phase:**
  - Records are processed by map function
- **Shuffle/Combine Phase:**
  - Distributed sort and grouping
- **Reduce Phase:**
  - Intermediate results are processed by reduce function



Hadoop MapReduce

- Data is processed by mappers and reducers

map(k, v) -> [(K1,V1), (K2,V2), ...]



reduce(Kn, [Vi, Vj, …]) -> (Km, R)

What kind of problems does MapReduce solve?

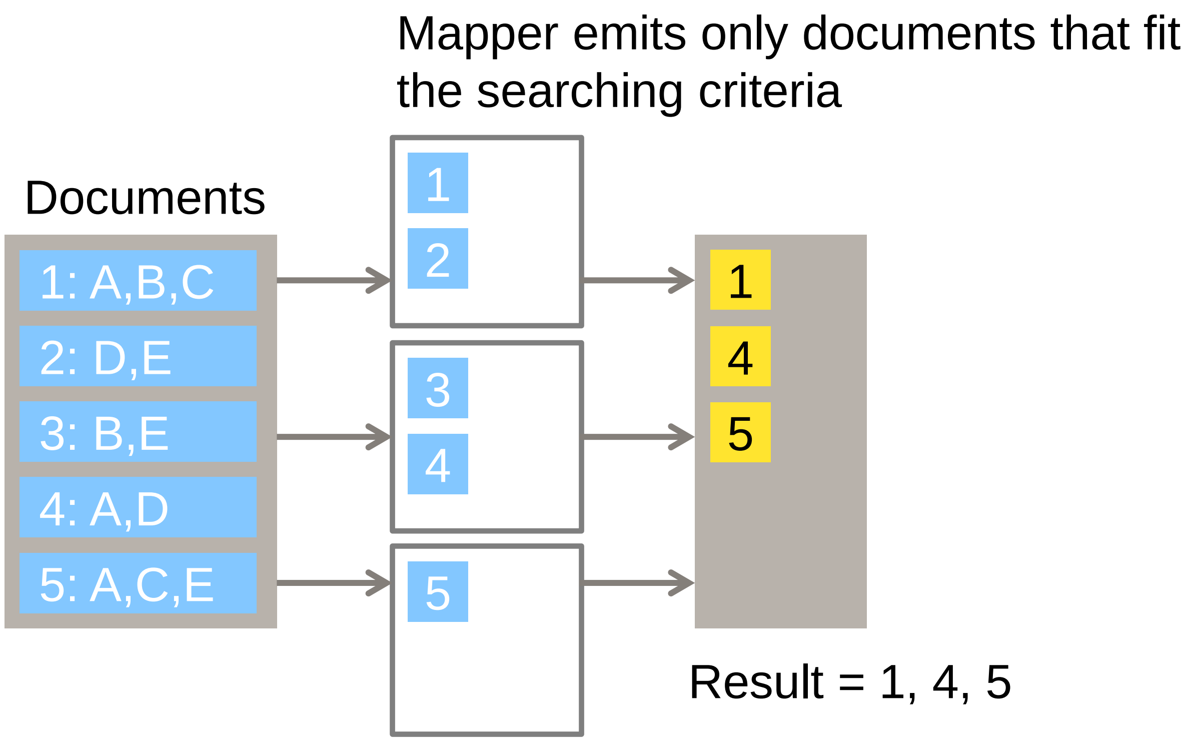
- Problems processed without reducer
  - Searching
  - File converting
  - Sorting
  - Map-side join

- Problem processed with reducer
  - Grouping and aggregation
  - Reduce-side join

- More complex problems:
  - Solved by combinations of multiple MapReduce jobs

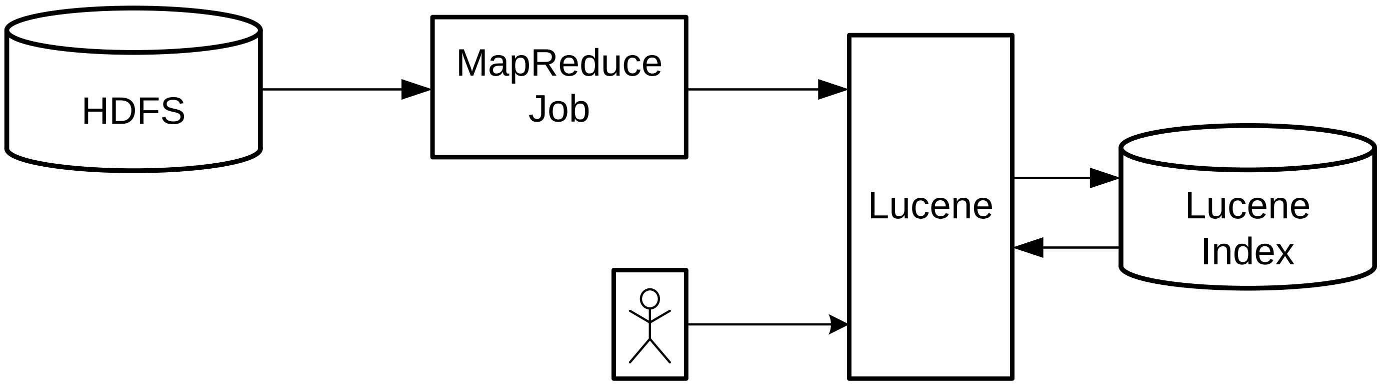
Hadoop MapReduce: Searching

- Search document including „A“

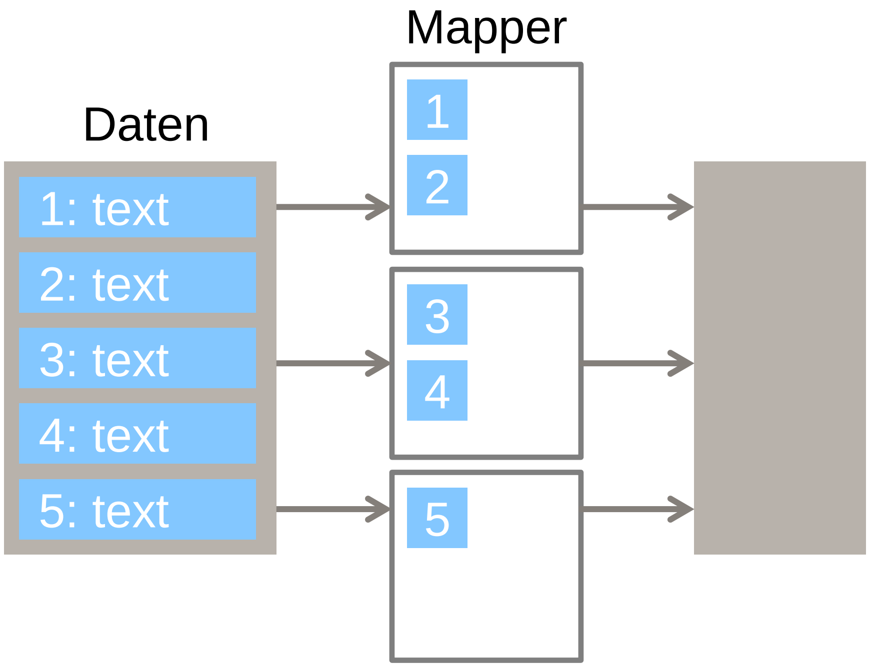


Hadoop MapReduce: Indexing

- HDFS:
  - > Stores raw data
- Mapper:
  - > Extracts text (creates e.g. SolrInputDocument)
  - > Calls Lucene for indexing (calls e.g. StreamingUpdateSolrServer)



Hadoop MapReduce: Indexing

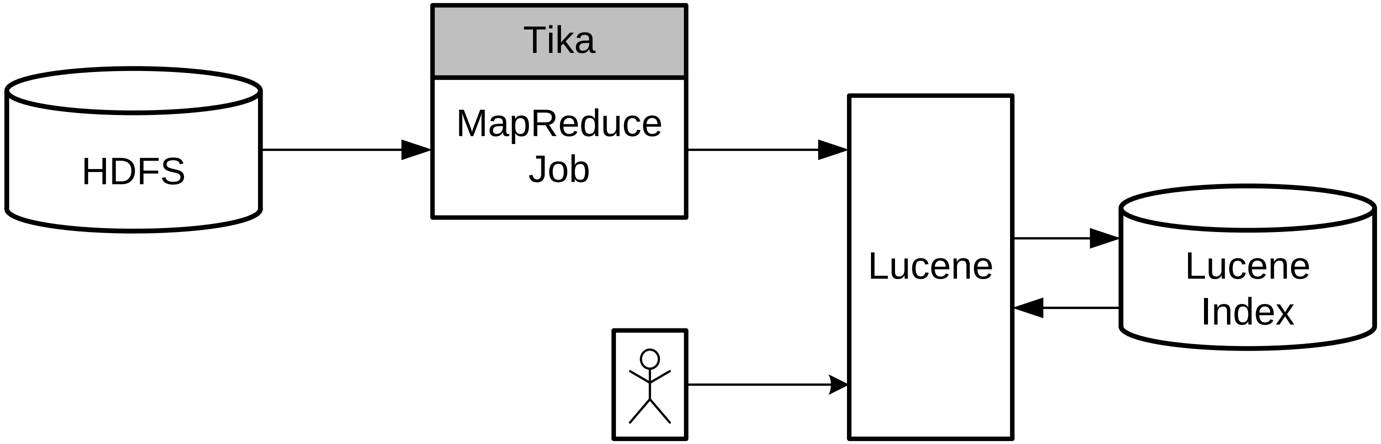



```
@Override
public void map(
         LongWritable key, Text val,
         OutputCollector<NullWritable,
         NullWritable> output,
         Reporter reporter)
         throws IOException {

 st = new StringTokenizer(val.toString());
 lineCounter = 0;

 while (st.hasMoreTokens()) {

  doc= new SolrInputDocument();

  doc.addField("id", fileName +
    key.toString() + lineCounter++);

  doc.addField("txt", st.nextToken());

  try {
   server.add(doc);
  } catch (Exception exp) {
   …
  }
}}
```


adesso

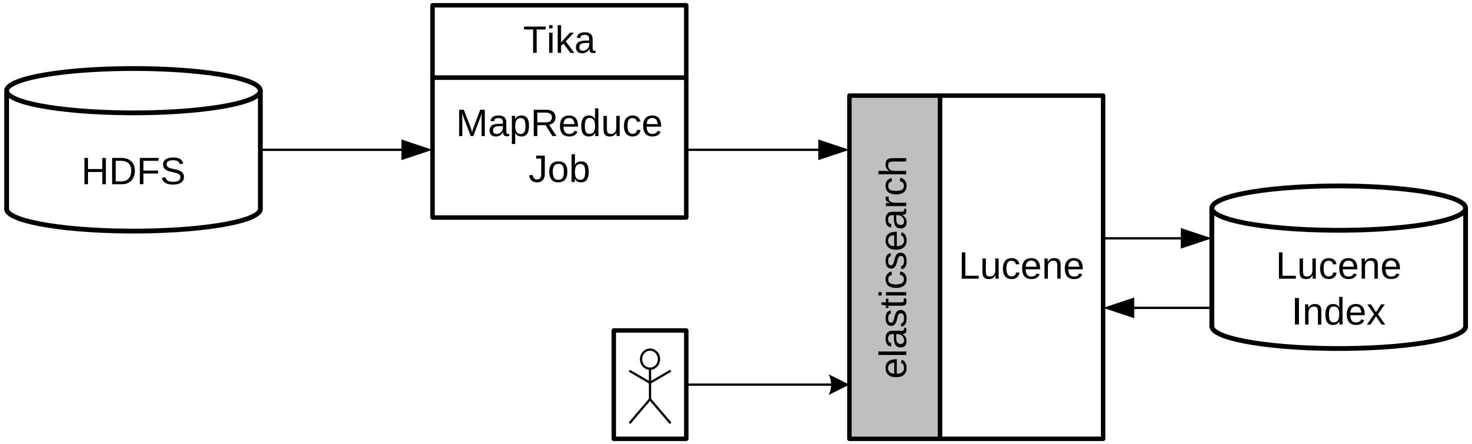
Apache Tika

- Extracts metadata and structured text content
  - HTML, MS Office documents, PDF, etc.
- Stream parser can process large files



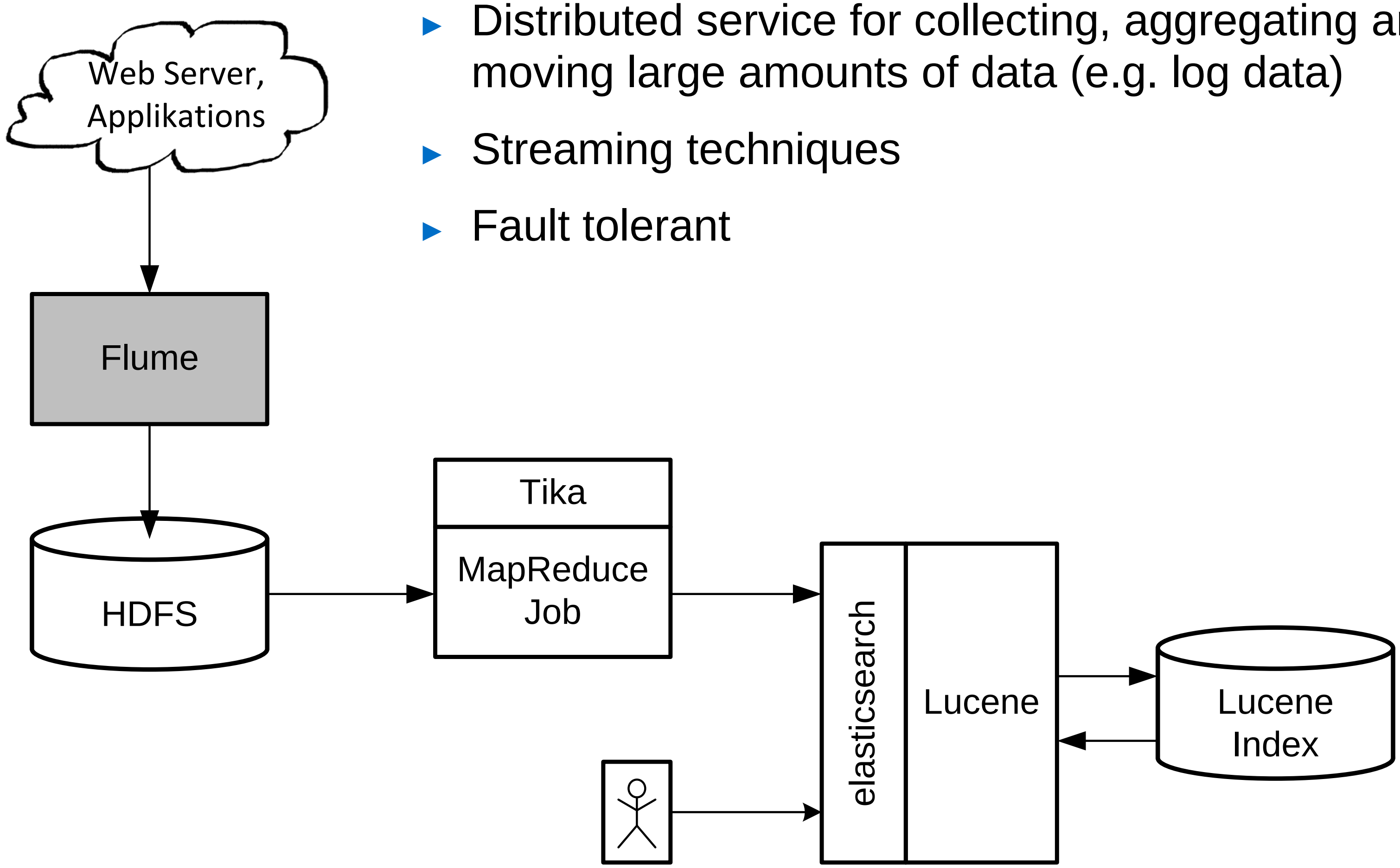
Apache Solr / elasticsearch

- Lucene is only a libary, not a standalone search engine
- Complete search engines:
  - Solr
  - ElasticSearch

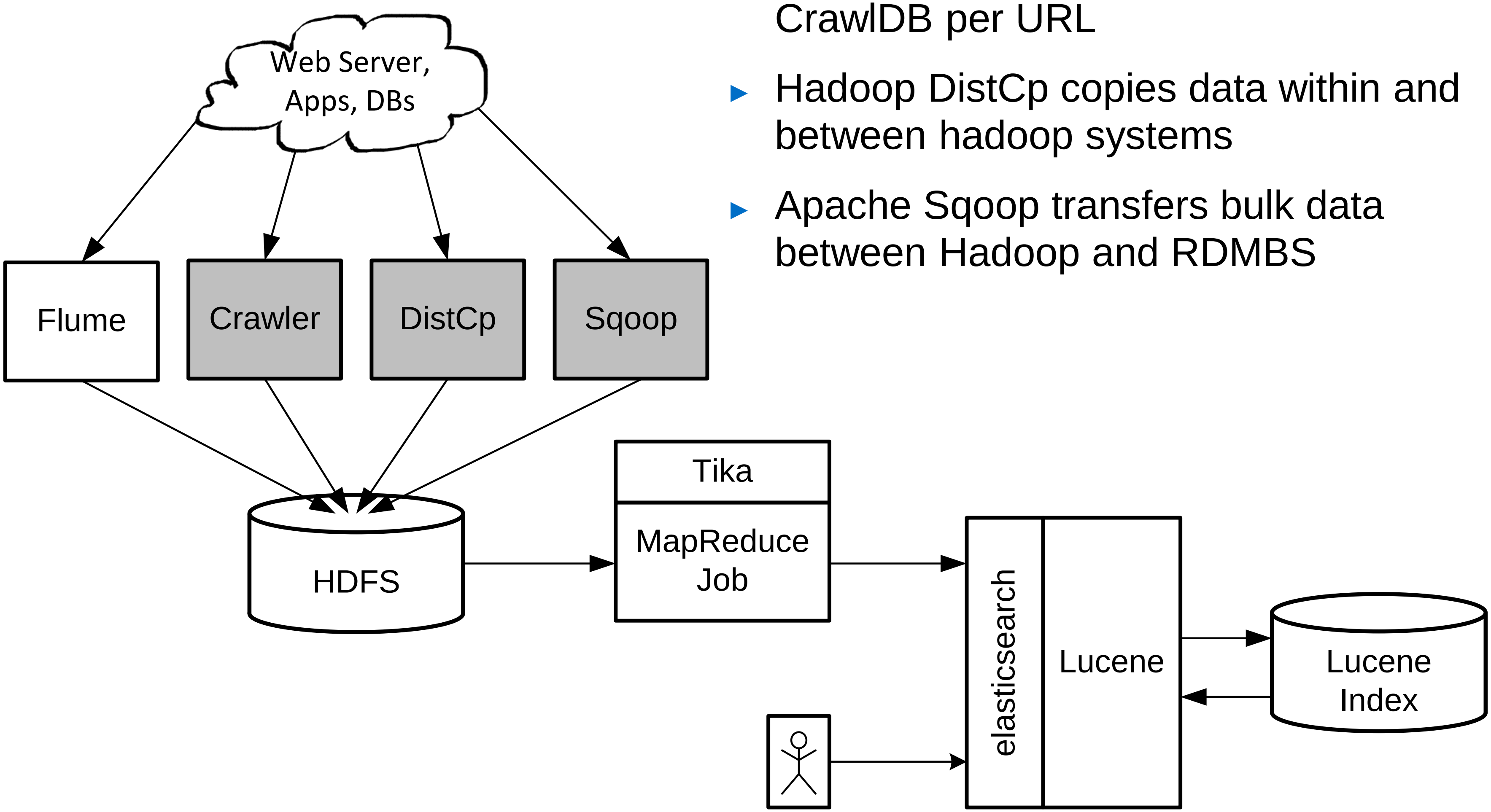


Apache Flume

- Distributed service for collecting, aggregating and moving large amounts of data (e.g. log data)
- Streaming techniques
- Fault tolerant

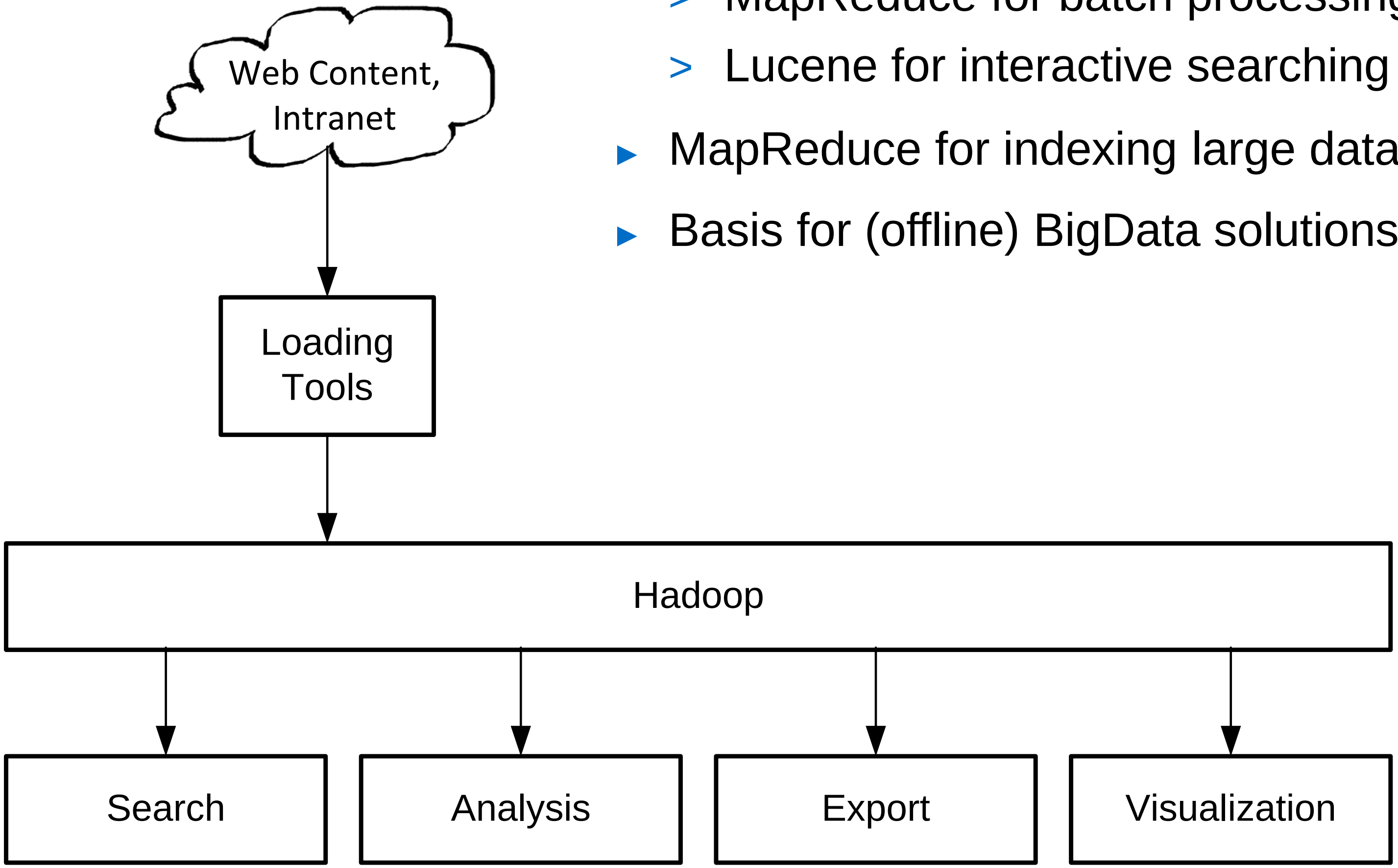


Alternatives



- Nutch Crawler creates one entry in CrawlDB per URL
- Hadoop DistCp copies data within and between hadoop systems
- Apache Sqoop transfers bulk data between Hadoop and RDMBS

Apache Hadoop



- Fundamental mismatch:
  - MapReduce for batch processing
  - Lucene for interactive searching
- MapReduce for indexing large datasets
- Basis for (offline) BigData solutions

Summary

- More semi-structured data
- Increasing relevance of full-text searching

- Combination of NoSQL and Lucene:
  - MongoDB: integration via MongoDB Connector
  - Neo4j: native Lucne integration
  - Cassandra: Datastax‘s Solr integration
  - Hadoop: indexing large datasets with MapReduce

- Alternative: search engine as document-oriented database

Thank you for your attention!