Introduction to The Solr Enterprise Search Server

Table of contents

1 Solr in a Nutshell

Solr is a standalone enterprise search server with a REST-like API. You put documents in it (called "indexing") via XML, JSON or binary over HTTP. You query it via HTTP GET and receive XML, JSON, or binary results.

- Advanced Full-Text Search Capabilities
- Optimized for High Volume Web Traffic
- Standards Based Open Interfaces - XML,JSON and HTTP
- Comprehensive HTML Administration Interfaces
- Server statistics exposed over JMX for monitoring
- Scalability - Efficient Replication to other Solr Search Servers
- Flexible and Adaptable with XML configuration
- Extensible Plugin Architecture

2 Solr Uses the Lucene Search Library and Extends it!

- A Real Data Schema, with Numeric Types, Dynamic Fields, Unique Keys
- Powerful Extensions to the Lucene Query Language
- Faceted Search and Filtering
- Geospatial Search
- Advanced, Configurable Text Analysis
- Highly Configurable and User Extensible Caching
- Performance Optimizations
- External Configuration via XML
- An Administration Interface
- Monitorable Logging
- Fast Incremental Updates and Index Replication
- Highly Scalable Distributed search with sharded index across multiple hosts
- JSON, XML, CSV/delimited-text, and binary update formats
- Easy ways to pull in data from databases and XML files from local disk and HTTP sources
- Rich Document Parsing and Indexing (PDF, Word, HTML, etc) using Apache Tika
- Apache UIMA integration for configurable metadata extraction
- Multiple search indices

3 Detailed Features

3.1 Schema

- Defines the field types and fields of documents

- Can drive more intelligent processing
- Declarative Lucene Analyzer specification
- Dynamic Fields enables on-the-fly addition of new fields
- CopyField functionality allows indexing a single field multiple ways, or combining multiple fields into a single searchable field
- Explicit types eliminates the need for guessing types of fields
- External file-based configuration of stopword lists, synonym lists, and protected word lists
- Many additional text analysis components including word splitting, regex and sounds-like filters

3.2 Query

- HTTP interface with configurable response formats (XML/XSLT, JSON, Python, Ruby, PHP, Velocity, binary)
- Sort by any number of fields, and by complex functions of numeric fields
- Advanced DisMax query parser for high relevancy results from user-entered queries
- Highlighted context snippets
- Faceted Searching based on unique field values, explicit queries, date ranges, and numeric ranges
- Multi-Select Faceting by tagging and selectively excluding filters
- Spelling suggestions for user queries
- More Like This suggestions for given document
- Function Query - influence the score by user specified complex functions of numeric fields or query relevancy scores.
- Range filter over Function Query results
- Date Math - specify dates relative to "NOW" in queries and updates
- Dynamic search results clustering using Carrot2
- Numeric field statistics such as min, max, average, standard deviation
- Combine queries derived from different syntaxes
- Auto-suggest functionality for completing user queries
- Allow configuration of top results for a query, overriding normal scoring and sorting
- Performance Optimizations

3.3 Core

- Dynamically create and delete document collections without restarting
- Pluggable query handlers and extensible XML data format
- Pluggable user functions for Function Query
- Customizable component based request handler with distributed search support

- Document uniqueness enforcement based on unique key field
- Duplicate document detection, including fuzzy near duplicates
- Custom index processing chains, allowing document manipulation before indexing
- User configurable commands triggered on index changes
- Ability to control where docs with the sort field missing will be placed
- "Luke" request handler for corpus information

3.4 Caching

- Configurable Query Result, Filter, and Document cache instances
- Pluggable Cache implementations, including a lock free, high concurrency implementation
- Cache warming in background
  - When a new searcher is opened, configurable searches are run against it in order to warm it up to avoid slow first hits. During warming, the current searcher handles live requests.
- Autowarming in background
  - The most recently accessed items in the caches of the current searcher are re-populated in the new searcher, enabling high cache hit rates across index/searcher changes.
- Fast/small filter implementation
- User level caching with autowarming support

3.5 Replication

- Efficient distribution of index parts that have changed
- Pull strategy allows for easy addition of searchers
- Configurable distribution interval allows tradeoff between timeliness and cache utilization
- Replication and automatic reloading of configuration files

3.6 Admin Interface

- Comprehensive statistics on cache utilization, updates, and queries
- Interactive schema browser that includes index statistics
- Replication monitoring
- Full logging control
- Text analysis debugger, showing result of every stage in an analyzer
- Web Query Interface w/ debugging output
  - parsed query output
  - Lucene explain() document score detailing

- explain score for documents outside of the requested range to debug why a given document wasn't ranked higher.