OpenSearchCon
NORTH AMERICA | San Francisco 2024

Lucene And Beyond

Core Storage Extension In OpenSearch

OpenSearchCon
NORTH AMERICA | San Francisco 2024

Lucene And Beyond

Core Storage Extension In OpenSearch

Samuel (Sam) Herman

Sr Principal Engineer - Oracle

Why Changing Storage Format?

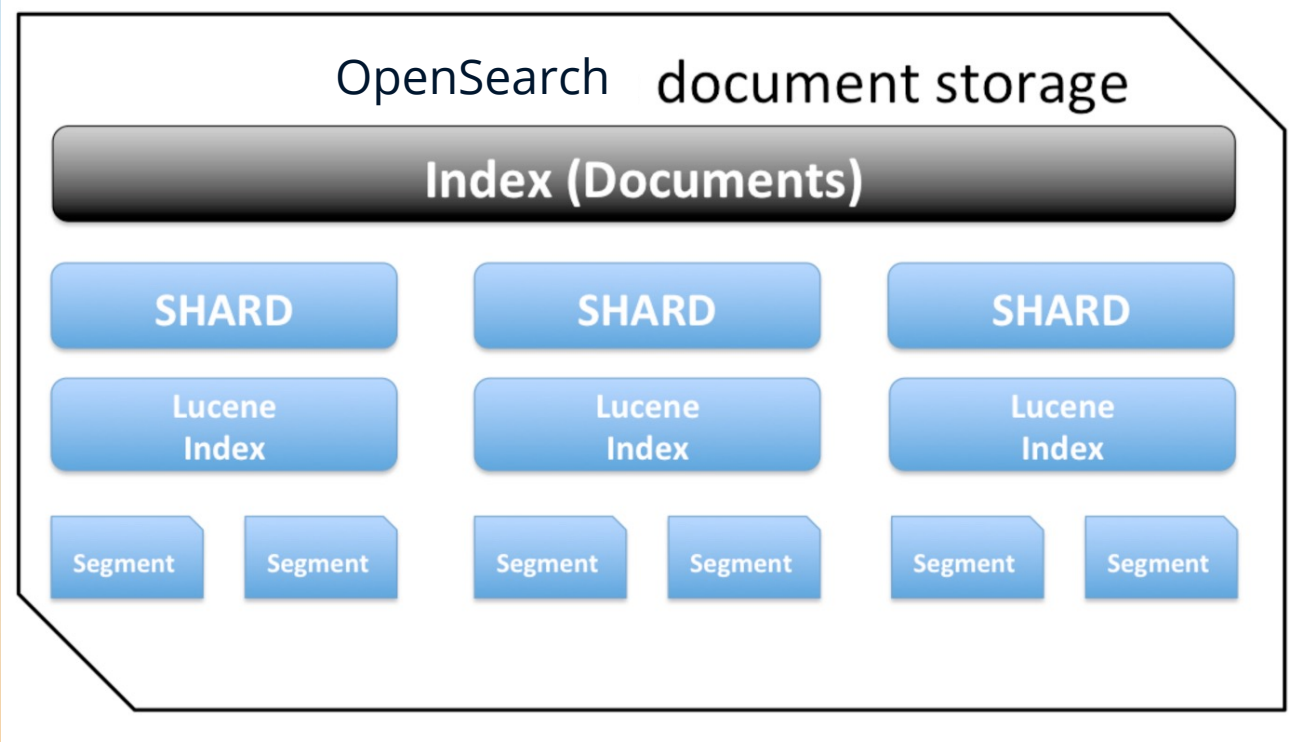
- Easier compatibility with other data warehousing solutions
- Extend functionality
- Better performance

Important Data Abstractions In OpenSearch

Mapper Extension - Adding new field/data type

Engine - Database operations

Lucene Codec - Primitive types properties (e.g. Encoding/Format)



Adding New Field/Mapping Type In OpenSearch

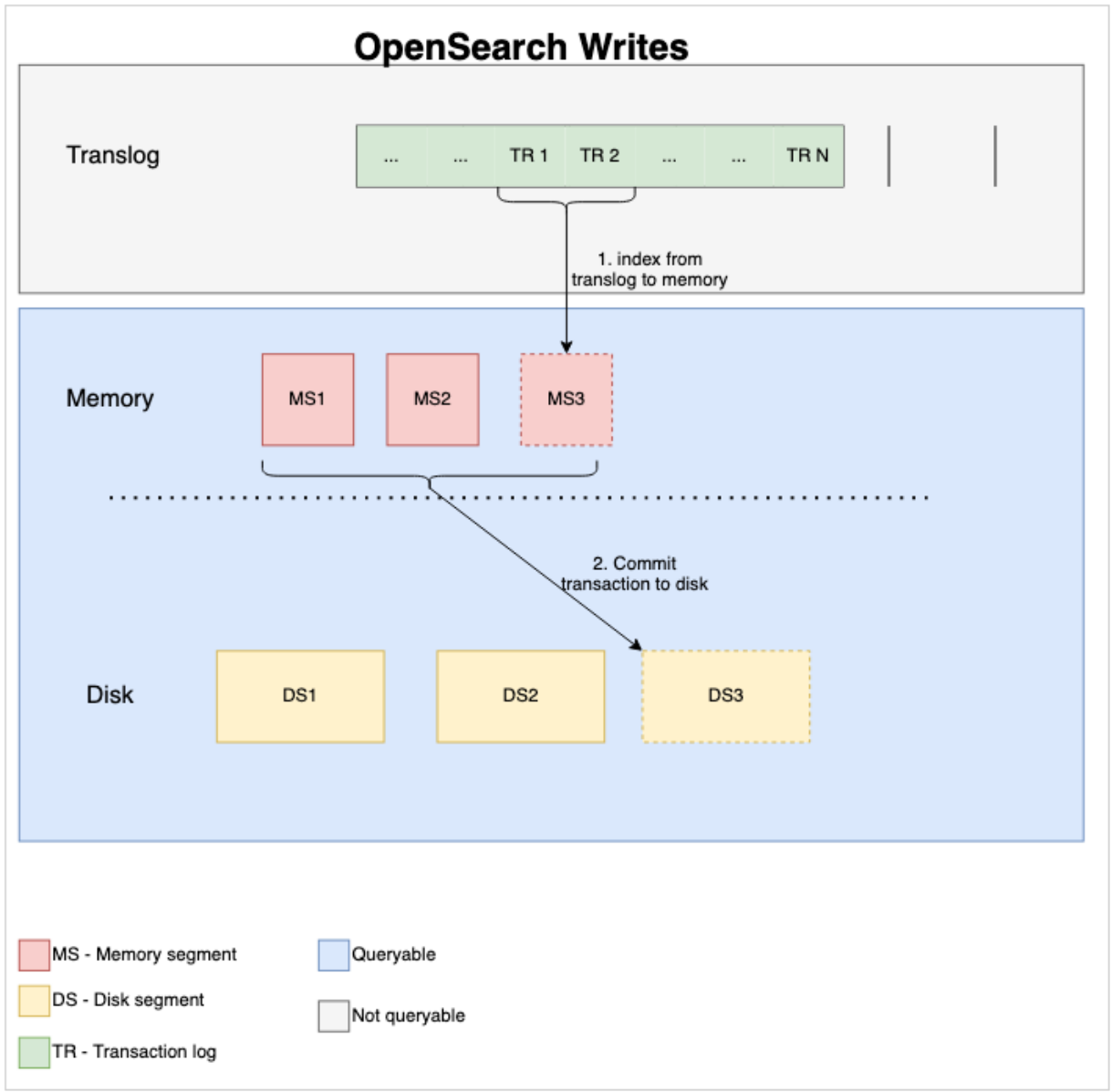
- IP, date, geo point ranges
- New field types
- Mappers and parsers
- Built upon more primitive types (e.g. Lucene)

Engine

Facilitates all index/read/search operations on a SHARD level

Transactions/merges/files tracking/commits etc..

Entirely coupled with Lucene



Lucene As An Extendible Framework

- Directory
- Index/search APIs
- Commits/Transactions/Segments management
- Fields
- Codec

Lucene – Documents and Fields

- Document is a collection of fields and values
- Field types: Text, String, long, int, double, sortedField
- Indexable/Non indexable
- Stored/Not stored
- DocValuesField/StoredField

Components of A Lucene Field

- Posting lists
- Doc Values
- Stored Fields
- Points
- Knn vectors

Lucene Codec

How each of the field formats are implemented:

- KnnVector - implemented as a dense vector
- Point – BKD tree

DocValues vs StoredFields

Row storage

AAPL	2014-01-06	543.93	...	....
AAPL	2014-01-07	540.04	...	...
AAPL	2014-01-07	540.04	...	...
AAPL	2014-01-07	540.04	...	...
AAPL	2014-01-07	540.04	...	...

Column storage

AAPL	2014-01-06	543.93	...	...
AAPL	2014-01-07	540.04	...	...
GOOG	2014-01-06	558.10	...	...
GOOG	2014-01-07	568.86	...	...
GOOG	2014-01-08	570.04	...	...

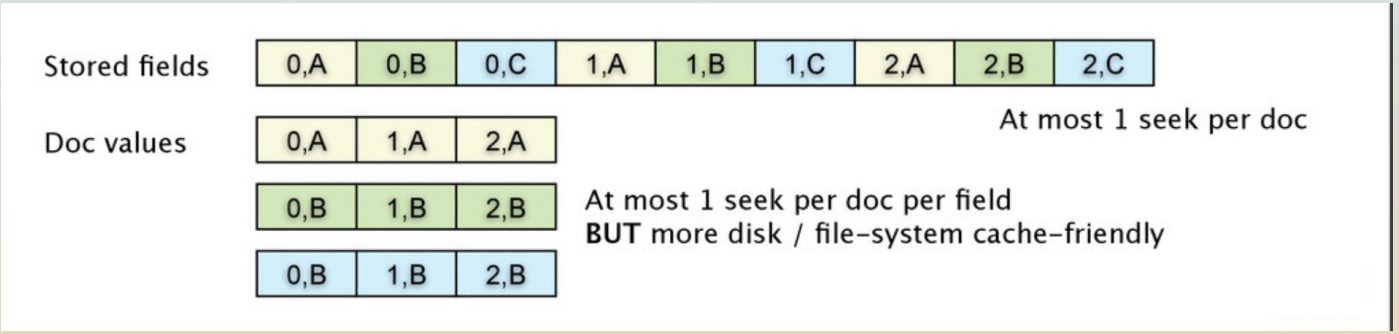
DocValues Vs StoredFields

Stored fields

- get many field values for a single doc
- LZ4 compressed (default)

Doc values

- Get few field values for many docs, good for faceting
- Various compression techniques optimized for each doc value type and also depending on the statistical properties of the data set in each block

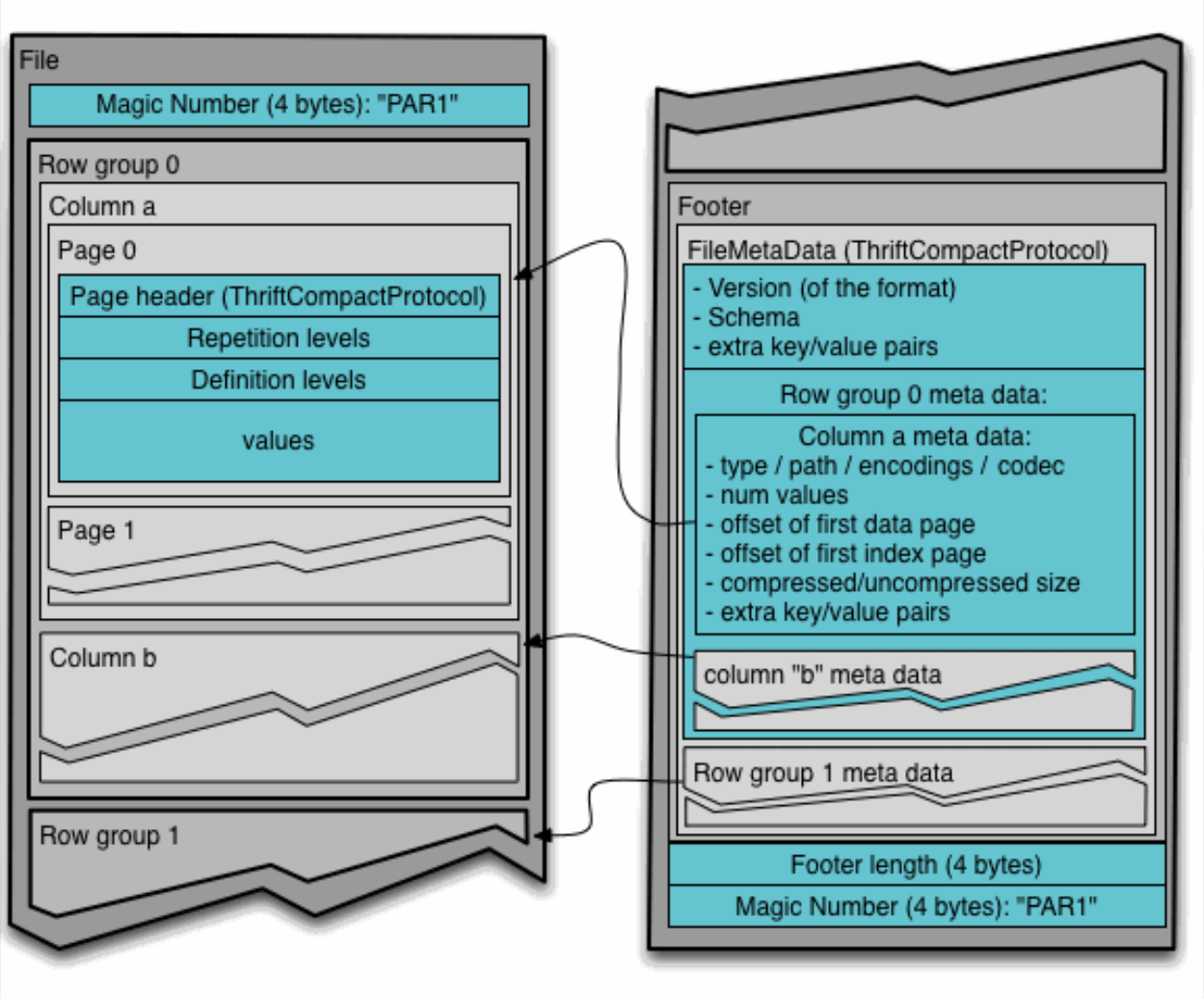


What About complex formats?

Where should we fit more complex formats? Parquet, Avro etc..

Parquet

```
4-byte magic number "PAR1"
<Column 1 Chunk 1 + Column Metadata>
<Column 2 Chunk 1 + Column Metadata>
...
<Column N Chunk 1 + Column Metadata>
<Column 1 Chunk 2 + Column Metadata>
<Column 2 Chunk 2 + Column Metadata>
...
<Column N Chunk 2 + Column Metadata>
...
<Column 1 Chunk M + Column Metadata>
<Column 2 Chunk M + Column Metadata>
...
<Column N Chunk M + Column Metadata>
File Metadata
4-byte length in bytes of file metadata (little endian)
4-byte magic number "PAR1"
```



Implement As DocValues?

- Consumer / Producer interface
- Different consumer for every field
- Optimized for throughput, writes all the doc values of a field in one go
- Limiting us in preserving row group structure

Implement As DocValues With Binary Fields?

- Treat as a single stored binary doc value that passes a record
- Client behavior change – awareness of data format
- OpenSearch with Lucene can be used to enhance the interpretation of the field

Implement As StoredFields?

- Preserves row group structure
- Easier to read outside of the cluster
- Double buffering
- Misleading the abstraction implementation

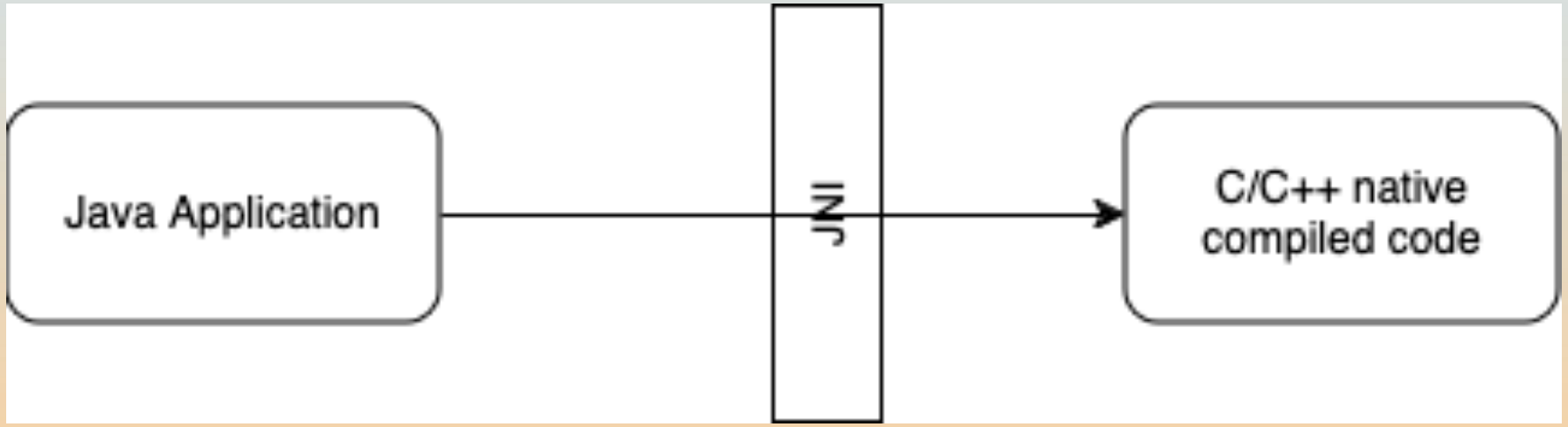
Extend the Lucene IndexingChain or other Interfaces

- Gap in Lucene interfaces
- IndexingChain can be extended
- Field can be extended

Limitations

- Lucene - Java
- OpenSearch – Java + JarHell

Possible Solutions



Extensions

- Experimental
- Not working with local storage
- Launch new process and make sure it's local
- Route all requests based on locality

https://github.com/opensearch-project/OpenSearch/issues/2447

Local External Writer

- Can be extended for non Java formats
- Solves the issues of locality
- Solves the issues of dependency collision

https://github.com/opensearch-project/OpenSearch/issues/13668

lucene-custom-format | master | App run

Project

- _0.kdm
- _0.nvd
- _0.nvm
- _0.prqd
- _0.si
- _0_Lucene90_0.dvd
- _0_Lucene90_0.dvm
- _0_Lucene99_0.doc
- _0_Lucene99_0.pos
- _0_Lucene99_0.tim
- _0_Lucene99_0.tip
- _0_Lucene99_0.tmd
- segments_1
- write.lock

IndexFiles.java | CustomCodec.java | App.java | build.gradle (:app) | ParquetCodec.java

```
45  public class IndexFiles implements AutoCloseable {  Samuel Herman *
48    public static void main(String[] args) throws Exception {  Samuel Herman *
100     Analyzer analyzer = new StandardAnalyzer();
101     IndexWriterConfig iwc = new IndexWriterConfig(analyzer);
102     iwc.setUseCompoundFile(false); // This makes it easier to inspect the files we are creating
103     //iwc.setCodec(new CustomCodec());
104     //iwc.setCodec(new ParquetCodec());
105     if (create) {
106       // Create a new index in the directory, removing any
107       // previously indexed documents:
108       iwc.setOpenMode(OpenMode.CREATE);
109     } else {
110       // Add new documents to an existing index:
111       iwc.setOpenMode(OpenMode.CREATE_OR_APPEND);
112     }
```

Run | App run

App run: successful At 9/22/24, 5:11 PM — 6 sec, 978 ms

```
resources/test-files/docs/gpl3.0.txt

g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/mozilla_eula_thunderbird2.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/freebsd.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/mozilla1.1.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/cpl1.0.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/lgpl2.1.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/epl1.0.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/apache1.1.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/apache1.0.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/gpl2.0.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/mit.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/lpgl2.0.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/lgpl3.txt"}
g a parquet record and caching it: {"path": "src/test/resources/test-files/docs/apache2.0.txt"}
```

lucene-custom-format > app > src > main > java > lucene > custom > format > IndexFiles > main — 104:9 LF UTF-8 2 spaces*

Lucene Custom Codec Demo (and Parquet)

opensearch-parquet-storage-plugin — -zsh — 168×55

...sr/local/bin/yarn start | ...rapperMain clean run | ...rapperMain clean run | ...torage-plugin — -zsh | ...torage-plugin — -zsh | ...bin-hadoop3 — -zsh

```
saherman@saherman-mac opensearch-parquet-storage-plugin % curl -XGET "http://localhost:9200/_cat/indices"

green open my-normal_index  -2296pI3Q4yRUqaay1yCYQ 1 0 2 0  3.4kb  3.4kb
green open my-parquet_index Ynjld6YsS-WGb29Df_-Nww 1 0 2 0  3.7kb  3.7kb
green open .kibana_1        9o7zIHZsRoqNCw2KqUGFQA 1 0 3 1 21.7kb 21.7kb
saherman@saherman-mac opensearch-parquet-storage-plugin %
```

OpenSearch On Parquet!

Questions?