Apache Lucene 5

New Features and Improvements for Apache Solr and Elasticsearch

Uwe Schindler

Apache Software Foundation | SD DataSolutions GmbH | PANGAEA

@thetaph1 · uschindler@apache.org

My Background

- **Committer** and **PMC member** of **Apache Lucene and Solr** - main focus is on development of Lucene Core.
- Implemented fast numerical search and maintaining the new attribute-based text analysis API. Well known as *Generics and Sophisticated Backwards Compatibility Policeman*.
- **Elasticsearch** lover.
- Working as consultant and software architect at **SD DataSolutions GmbH** in Bremen, Germany.
- Maintaining **PANGAEA** (Publishing Network for Geoscientific & Environmental Data) where I implemented the portal's geo-spatial retrieval functions with Apache Lucene Core and Elasticsearch.

History

ON THE WAY TO Lucene 5…

History: Lucene up to version 3.6

History: Lucene up to version 3.6

Lucene started > 10 years ago

History: Lucene up to version 3.6

Lucene started > 10 years ago

Lucene's VINT format is old and not as friendly as new compression algorithms to CPU's optimizers *(exists since Lucene 1.0)*

History: Lucene up to version 3.6

History: Lucene up to version 3.6

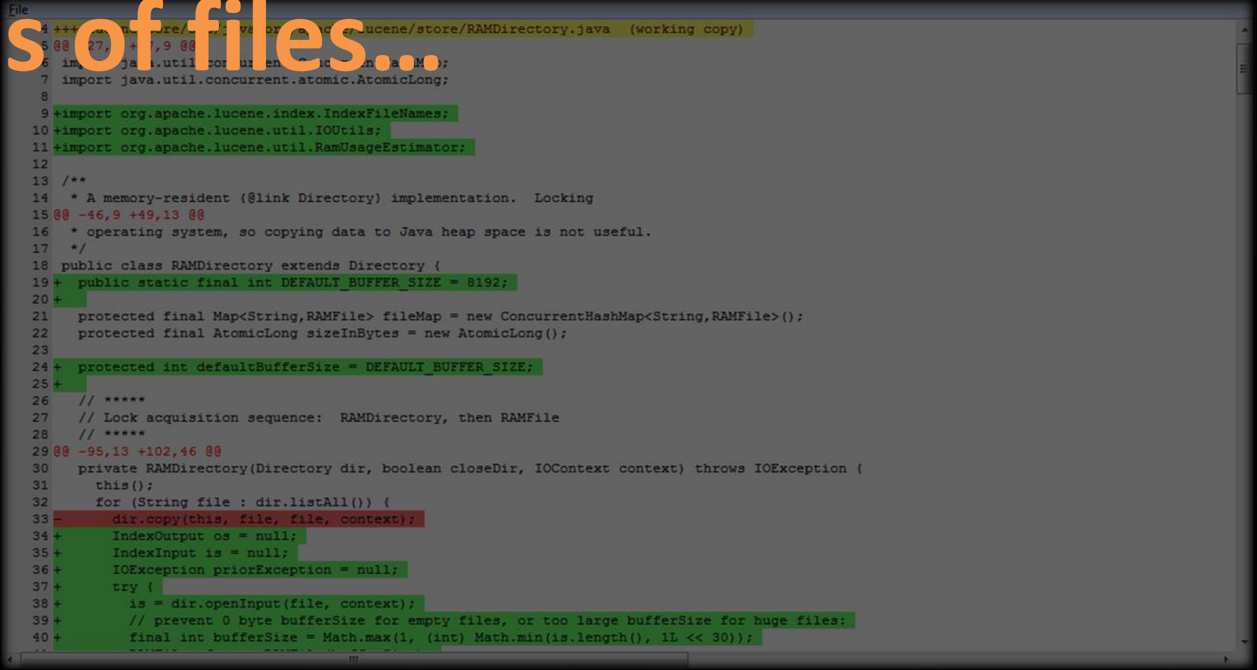
It was hard to add additional **statistics** for scoring to the index

History: Lucene up to version 3.6

It was hard to add additional **statistics** for scoring to the index

IR researchers didn’t use Apache Lucene to try out **new algorithms**

Small changes to index format were often huge patches covering tons of files...



```
+++ [illegible]ore/[illegible]va[illegible]or[illegible]ucene/store/RAMDirectory.java  (working copy)
@@ [illegible]27,[illegible] +[illegible],9 @@
 im[illegible] ja[illegible].util.co[illegible]ur[illegible]en[illegible]a[illegible]M[illegible];
 import java.util.concurrent.atomic.AtomicLong;

+import org.apache.lucene.index.IndexFileNames;
+import org.apache.lucene.util.IOUtils;
+import org.apache.lucene.util.RamUsageEstimator;

 /**
  * A memory-resident {@link Directory} implementation.  Locking
@@ -46,9 +49,13 @@
  * operating system, so copying data to Java heap space is not useful.
  */
 public class RAMDirectory extends Directory {
+  public static final int DEFAULT_BUFFER_SIZE = 8192;
+
   protected final Map<String,RAMFile> fileMap = new ConcurrentHashMap<String,RAMFile>();
   protected final AtomicLong sizeInBytes = new AtomicLong();

+  protected int defaultBufferSize = DEFAULT_BUFFER_SIZE;
+
   // *****
   // Lock acquisition sequence:  RAMDirectory, then RAMFile
   // *****
@@ -95,13 +102,46 @@
   private RAMDirectory(Directory dir, boolean closeDir, IOContext context) throws IOException {
     this();
     for (String file : dir.listAll()) {
-      dir.copy(this, file, file, context);
+      IndexOutput os = null;
+      IndexInput is = null;
+      IOException priorException = null;
+      try {
+        is = dir.openInput(file, context);
+        // prevent 0 byte bufferSize for empty files, or too large bufferSize for huge files:
+        final int bufferSize = Math.max(1, (int) Math.min(is.length(), 1L << 30));
```

History: Apache Lucene 4

- Major release in October 2012

History: Apache Lucene 4

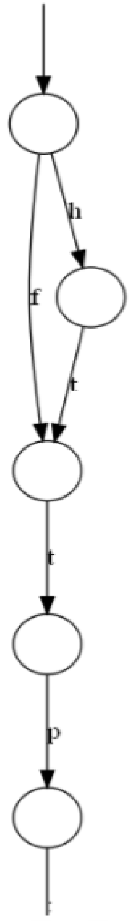
- Major release in October 2012
- New index engine:
  - **Codec** support *(pluggable via SPI)*
  - **DocValues** fields

History: Apache Lucene 4

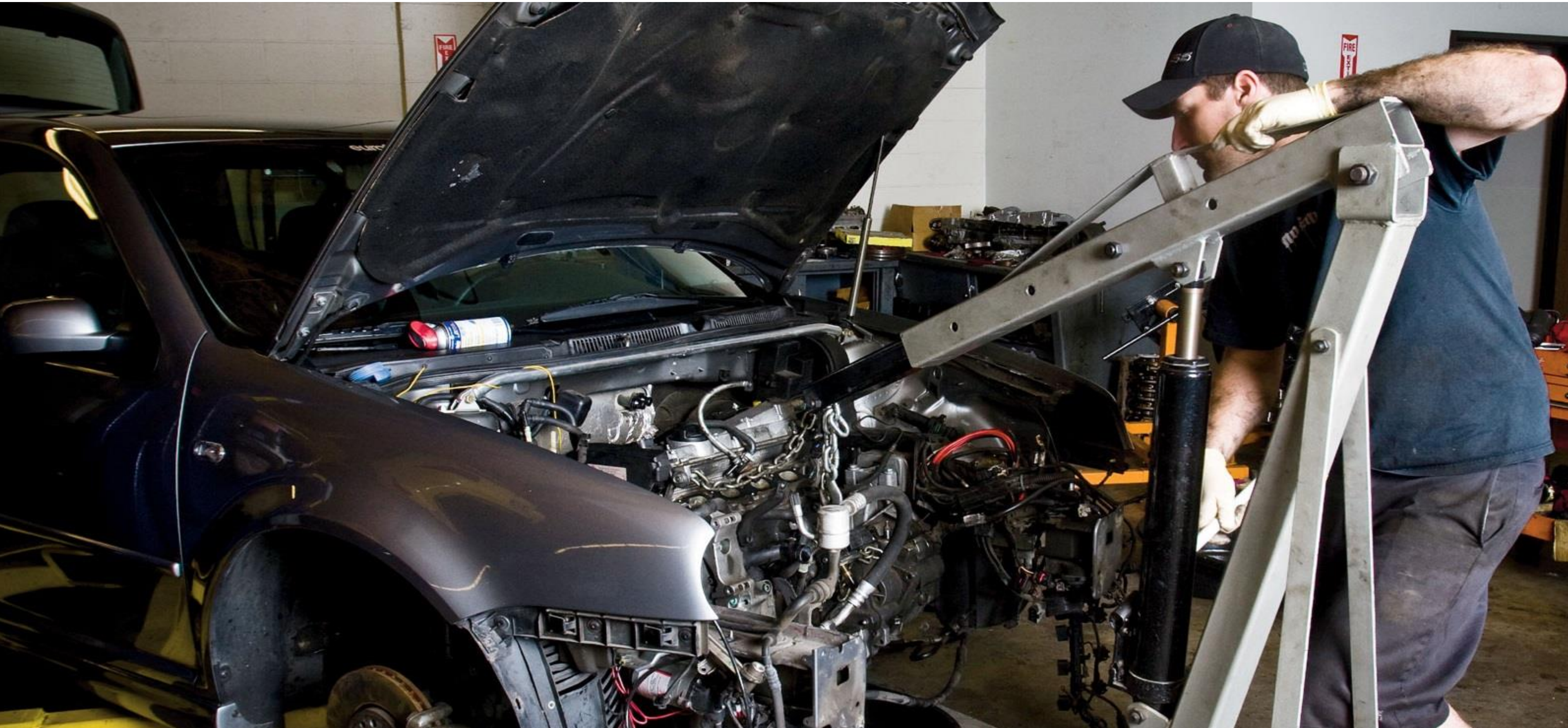
- Major release in October 2012
- New index engine:
  - **Codec** support *(pluggable via SPI)*
  - **DocValues** fields
- New relevancy models: not only TF/IDF!
  - e.g., **BM25**

History: Apache Lucene 4

- Major release in October 2012
- New index engine:
  - **Codec** support *(pluggable via SPI)*
  - **DocValues** fields
- New relevancy models: not only TF/IDF!
  - e.g., **BM25**
- FSAs / FSTs everywhere



History: Apache Lucene 4



History: Apache Lucene 4

Complete overhaul of all APIs

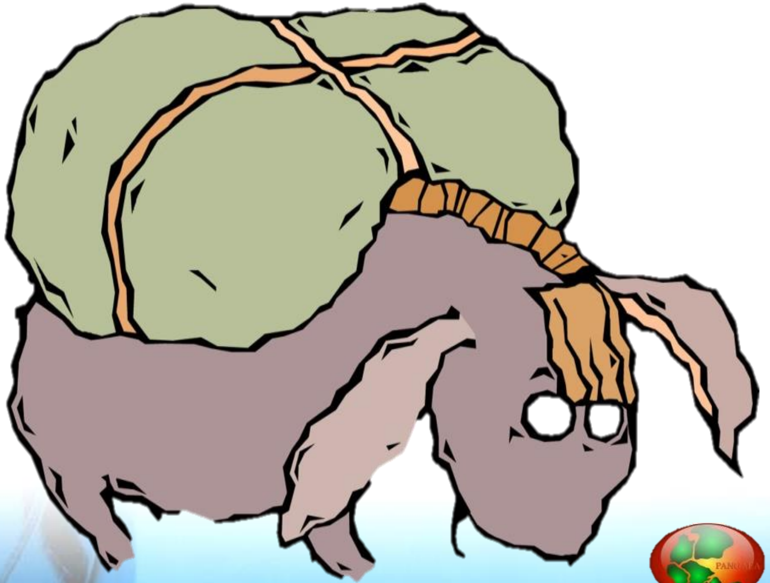
- Terms got `byte[]`
- Low level terms enumerations and postings enumerations refactored
- Query API internals (scorer, weight)
- Analyzers: new module, package structure changed *(pluggable via SPI)*
- IndexReader => AtomicReader, CompositeReader

History: Apache Lucene 4

- Every Lucene 4 release got new features!
  - API glitches!!!

History: Apache Lucene 4

- Every Lucene 4 release got new features!
  - API glitches!!!
- Burden of maintaining the old stuff:
  - old index formats
  - especially support for **Lucene 3.x** indexes



On-going Disasters

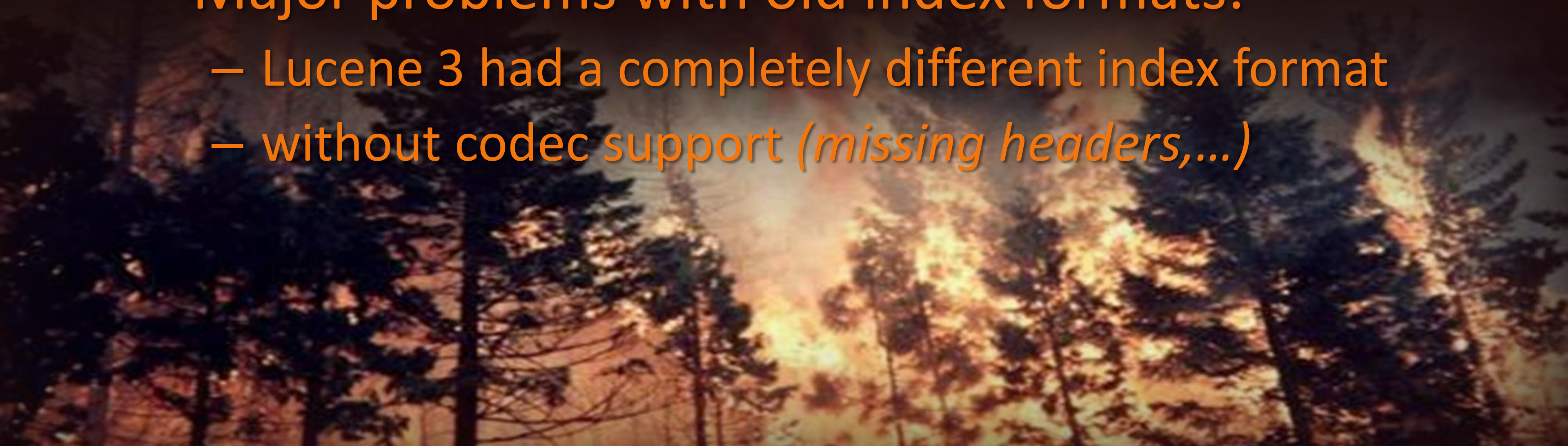
- Not only problems with bugs in Java runtimes

On-going Disasters

- Not only problems with bugs in Java runtimes
  - Story could fill another talk! ☺

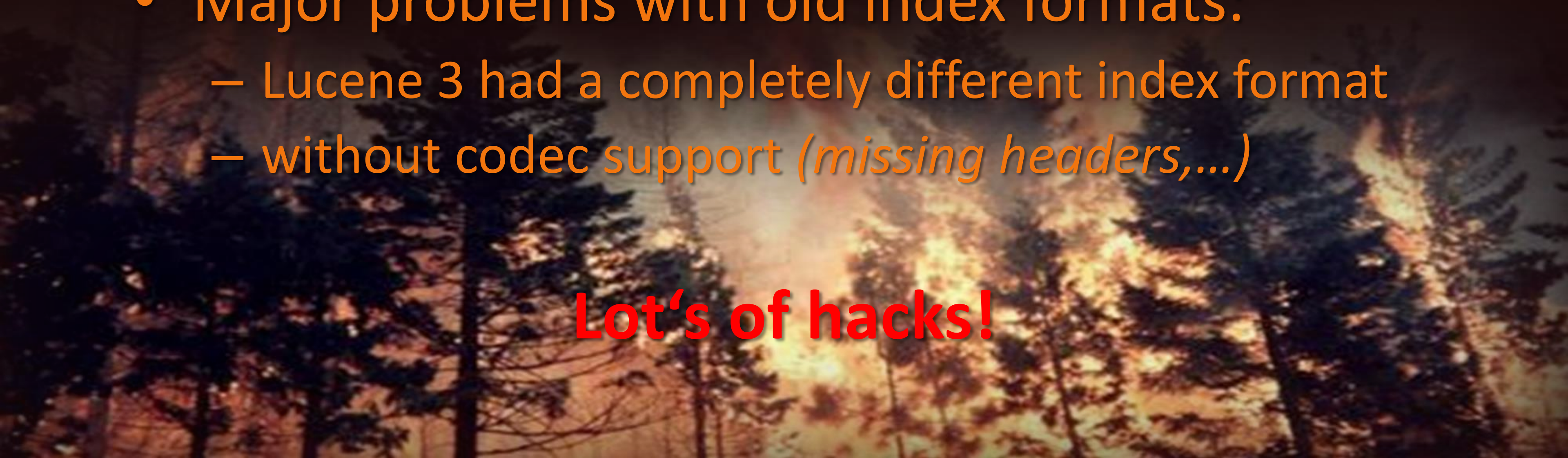
On-going Disasters

- Not only problems with bugs in Java runtimes
  - Story could fill another talk! ☺
- Major problems with old index formats:
  - Lucene 3 had a completely different index format
  - without codec support *(missing headers,…)*



On-going Disasters

- Not only problems with bugs in Java runtimes
  - Story could fill another talk! ☺
- Major problems with old index formats:
  - Lucene 3 had a completely different index format
  - without codec support *(missing headers,…)*



Lot's of hacks!

Chronology

- **Lucene 4.2.0:** Lucene deletes entire index if exception is thrown due do too many open files with `OpenMode.CREATE_OR_APPEND` *(LUCENE-4870)*
- **Lucene 4.9.0:** Closing NRT reader after upgrading from 3.x index can cause index corruption *(LUCENE-5907)*
- **Lucene 4.10.0:** Index version numbers caused `CorruptIndexException` *(LUCENE-5934)*

Lucene 5

Lucene 5

A lot new features!

Lucene 5

A lot new features!

- But not so many as you would expect for major release!

Lucene **5**

A lot new features!

- But not so many as you would expect for major release!
- Some more than in previous minor 4.x releases…

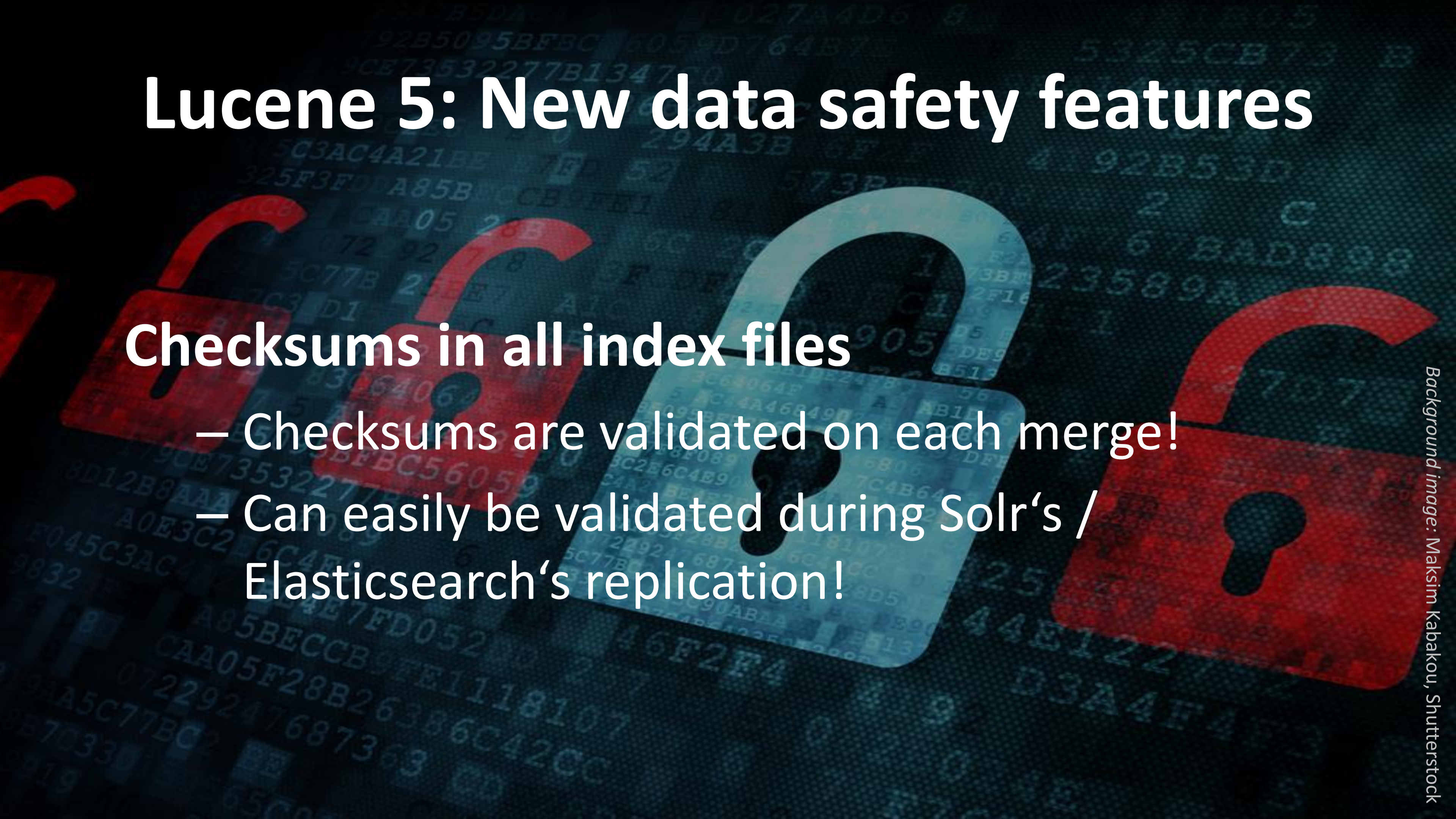
Lucene 5: "Anti-Feature"

Removal of Lucene 3 index support!

Lucene 5: "Anti-Feature"

Removal of Lucene 3 index support!

- **Get rid of old index segments:** `IndexUpgrader` in latest Lucene 4 release helps!
- **Elasticsearch** has automatic index upgrader already implemented / **Solr** users have to manually do this



Lucene 5: New data safety features

Checksums in all index files

- Checksums are validated on each merge!
- Can easily be validated during Solr‘s / Elasticsearch‘s replication!

Background image: Maksim Kabakou, Shutterstock

Lucene 5: New data safety features

Unique per segment ID

- ensures that the reader really sees the segment mentioned in the commit
- prevents bugs caused by failures in replication (e.g., duplicate segment file names)

Java 7 support

- Introduced in **Lucene 4.8**
  - *Could have been "Lucene 5" already ☺*
- **Why?**
  - EOL of Java 6, but still bugs that affected Lucene
  - Java 8 released
  - **use of new features for <u>index safety</u>!**

Java 7 Support

Java 7 Support

Try-With-Resources

- Nice, but we had it already implemented: `IOUtils#closeWhileHandlingExceptions`

Java 7 Support

- Try-With-Resources
  - Nice, but we had it already implemented: `IOUtils#closeWhileHandlingExceptions`
- Some syntactic sugar ☺

Java 7 Support

- Try-With-Resources
  - Nice, but we had it already implemented: `IOUtils#closeWhileHandlingExceptions`
- Some syntactic sugar ☺
- `MethodHandle` / `ClassValue` for Tokenization API‘s internals
  - Huge speedup for dynamic instantiation of token Attributes, especially in Java 8!

Java 7u55+ has
no serious bugs anymore

(still a **no-go** for **G1GC** with Lucene*)

Java 7u55+ has **no serious bugs anymore**

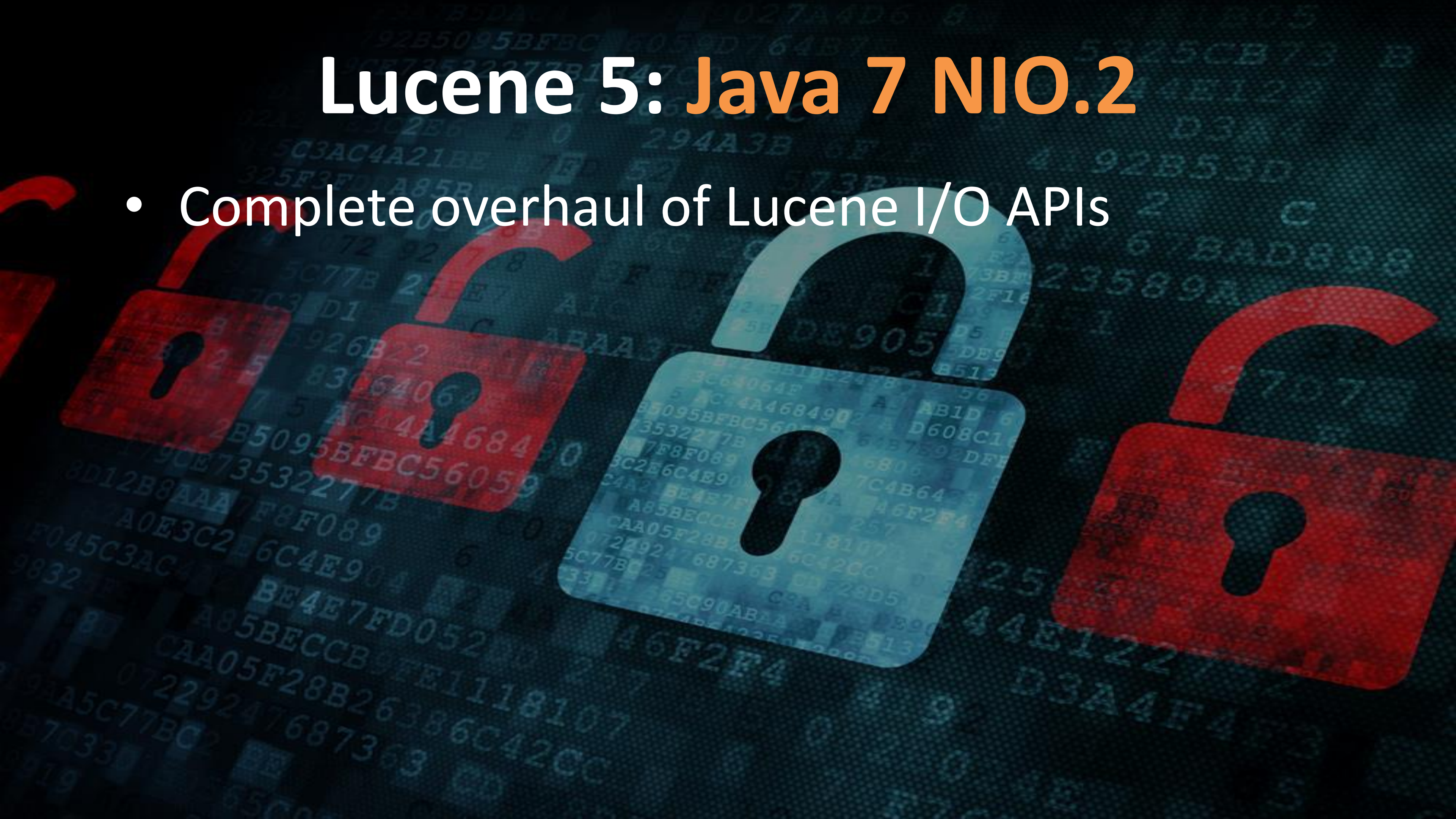
(still a **no-go** for **G1GC** with Lucene*)

**) we‘re investigating Java 8u40+ !!!*

Lucene 5: New index safety features

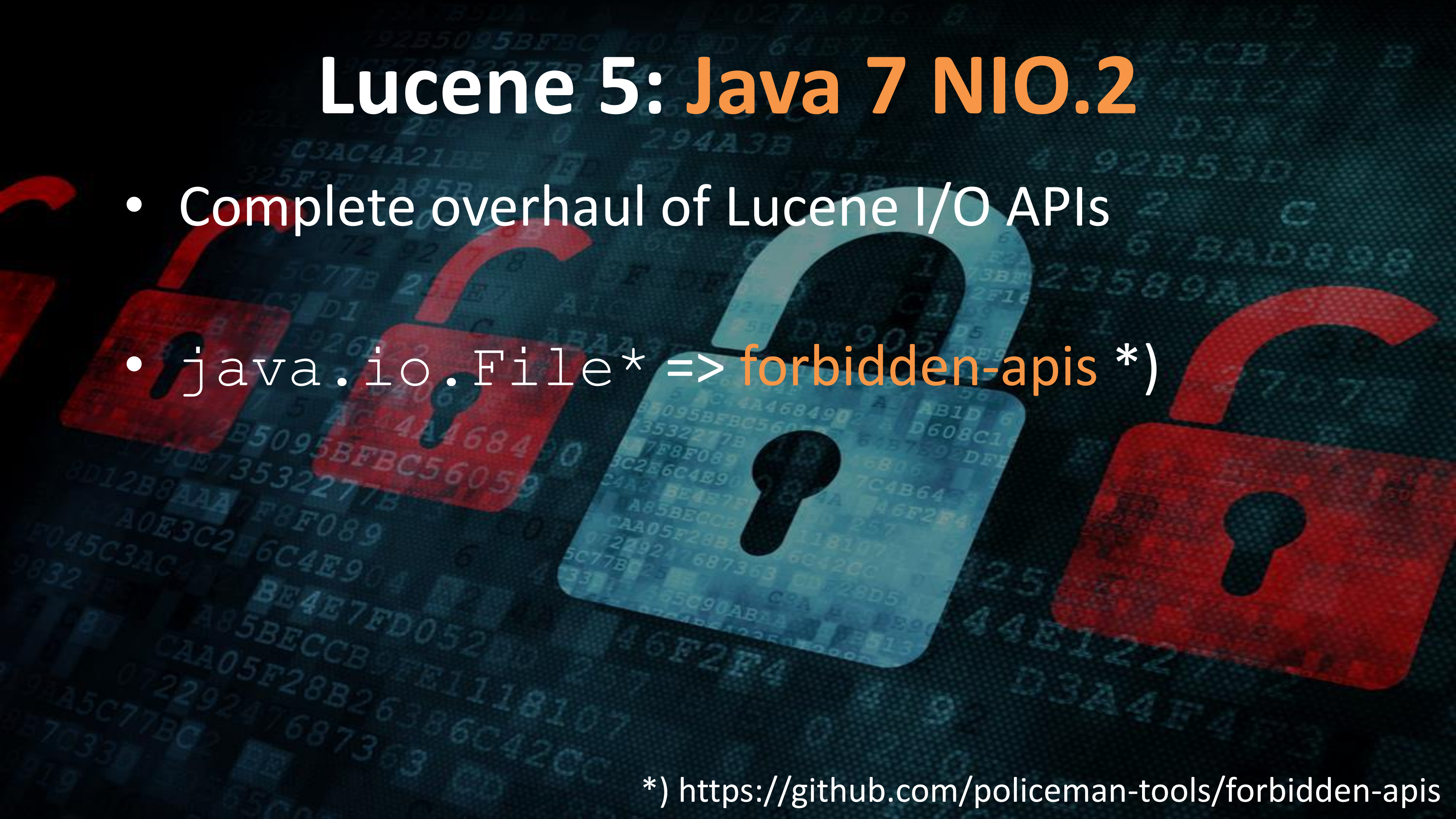
Cutover to NIO.2

(Java 7, JSR 203)



Lucene 5: Java 7 NIO.2

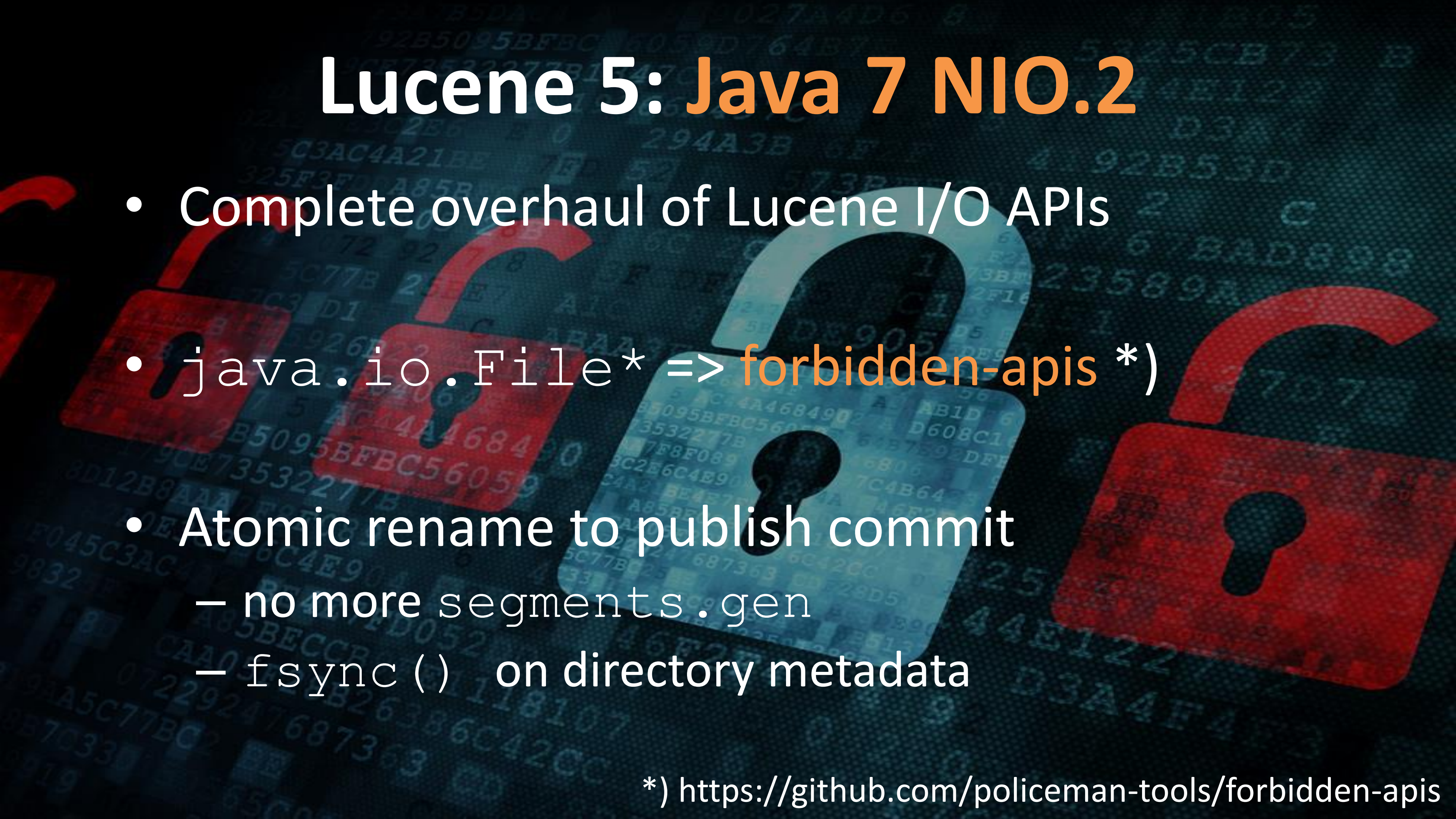
- Complete overhaul of Lucene I/O APIs



Lucene 5: Java 7 NIO.2

- Complete overhaul of Lucene I/O APIs
- `java.io.File*` => forbidden-apis *)

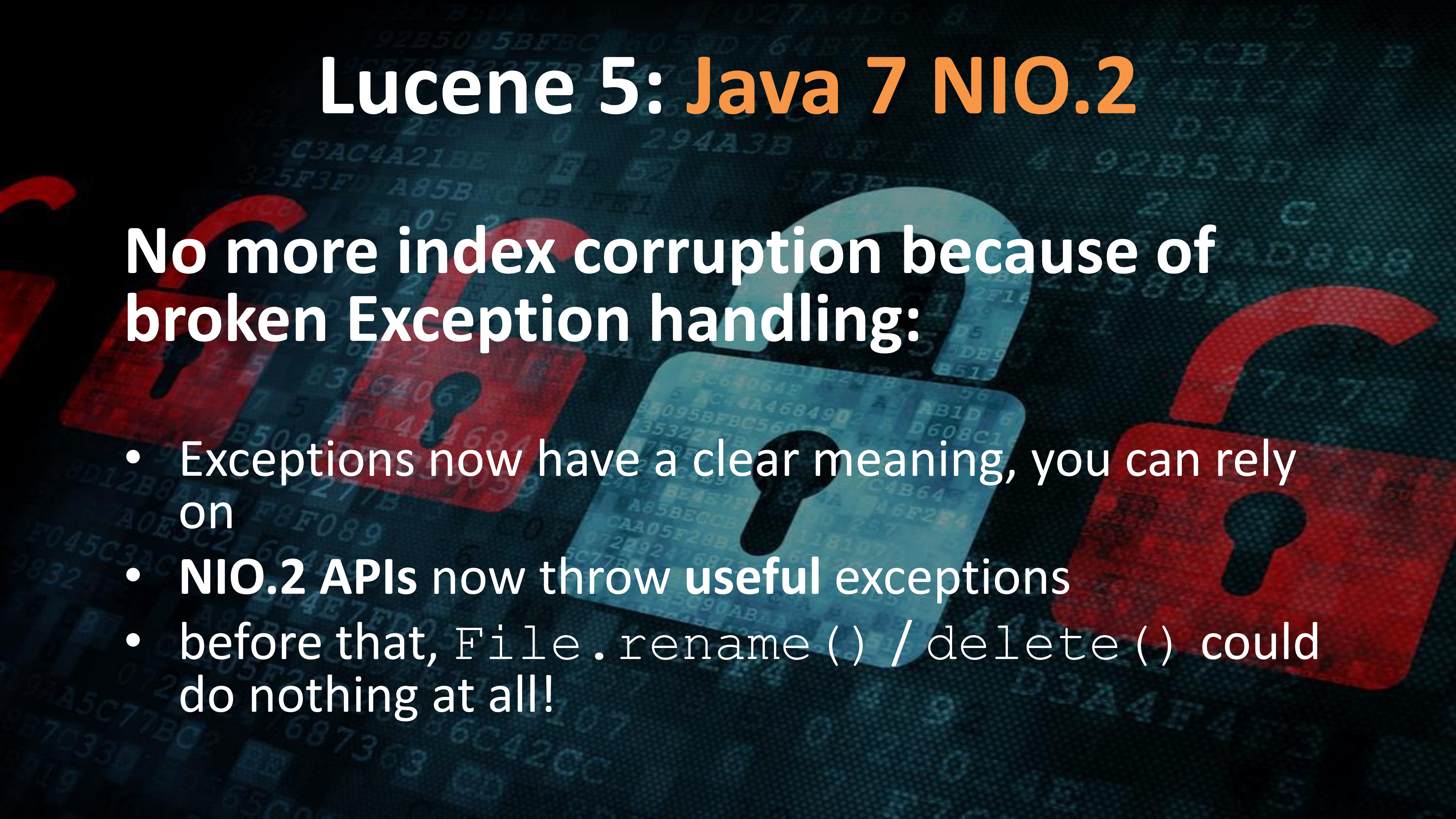
*) https://github.com/policeman-tools/forbidden-apis



Lucene 5: Java 7 NIO.2

- Complete overhaul of Lucene I/O APIs
- `java.io.File*` => forbidden-apis *)
- Atomic rename to publish commit
  - no more `segments.gen`
  - `fsync()` on directory metadata

*) https://github.com/policeman-tools/forbidden-apis



Lucene 5: Java 7 NIO.2

No more index corruption because of broken Exception handling:

- Exceptions now have a clear meaning, you can rely on
- **NIO.2 APIs** now throw **useful** exceptions
- before that, `File.rename()` / `delete()` could do nothing at all!

Java 7 NIO.2 - Consequences

Java 7 NIO.2 - Consequences

- Use Java 7 APIs to open indexes:
  `Paths.get()`

Java 7 NIO.2 - Consequences

- Use Java 7 APIs to open indexes: `Paths.get()`
- All file I/O is now channel based *(or mmap)*
  - if interrupted throws `ClosedByInterruptException`
  - also `SimpleFSDirectory`!

Java 7 NIO.2 - Consequences

Java 7 NIO.2 - Consequences

- Never use `Future.cancel(true)` !!!
  - Never interrupt searching threads, it kills your `IndexReader`!
  - Alternative: `org.apache.lucene.store.RAFDirectory` (RAF = RandomAccessFile, only available in “misc” module)

Lucene 5: Overhaul of Codec API

- Pull APIs throughout Codec components
  - E.g., PostingsFormat
- Norms are now handled by separate codec component

Lucene 5: Index merging

Lucene 5: Index merging

- Linux: Detection if index is on SSD
  - Better default merging settings
  - Other operating systems assume spinning disks (no change)

Lucene 5: Index merging

- Linux: Detection if index is on SSD
  - Better default merging settings
  - Other operating systems assume spinning disks (no change)
- Merge Scheduler: Auto Throttling
  - Automatically controls I/O rates based on indexing/merging rate
  - Stalling under high load is more unlikely!

Lucene 5: Reduced Heap Usage

- Query Filters uses new bit set types
- `CachingWrapperFilter` replacement:
  - New, highly configureable filter cache
  - Tracks filter‘s frequency of use
  - Simplifies code in Apache Solr and Elasticsearch
- Merging uses much less heap

Lucene 5: Reduced Heap Usage

- Query Filters uses new bit set types
- `CachingWrapperFilter` replacement:
  - New, highly configureable filter cache
  - Tracks filter‘s frequency of use
  - Simplifies code in Apache Solr and Elasticsearch
- Merging uses much less heap

- Most classes now implement Accountable
  - Allows to query heap usage
  - Nice "tree view" on heap usage of index components

Lucene 5: Reduced Heap Usage

- Query Filters uses new bit set types
- `CachingWrapperFilter` replacement:
  - New, highly configureable filter cache
  - Tracks filter‘s frequency of use
  - Simplifies code in Apache Solr and Elasticsearch
- Merging uses much less heap

```
_cz(5.0.0):C8330469: 28MB
 postings [...]: 5.2MB
 ...
 field 'latitude' [...]: 678.5KB
  term index [FST(nodes=6679, ...)]: 678.3KB
```

Lucene 5: CustomAnalyzer

- Freely configurable Analyzer
- Based on SPI framework for Tokenizers, TokenFilters and CharFilters
- Similar to Apache Solr‘s schema.xml:
  - Generic names of components (like Elasticsearch)
  - Same config options like Apache Solr
- Builder API

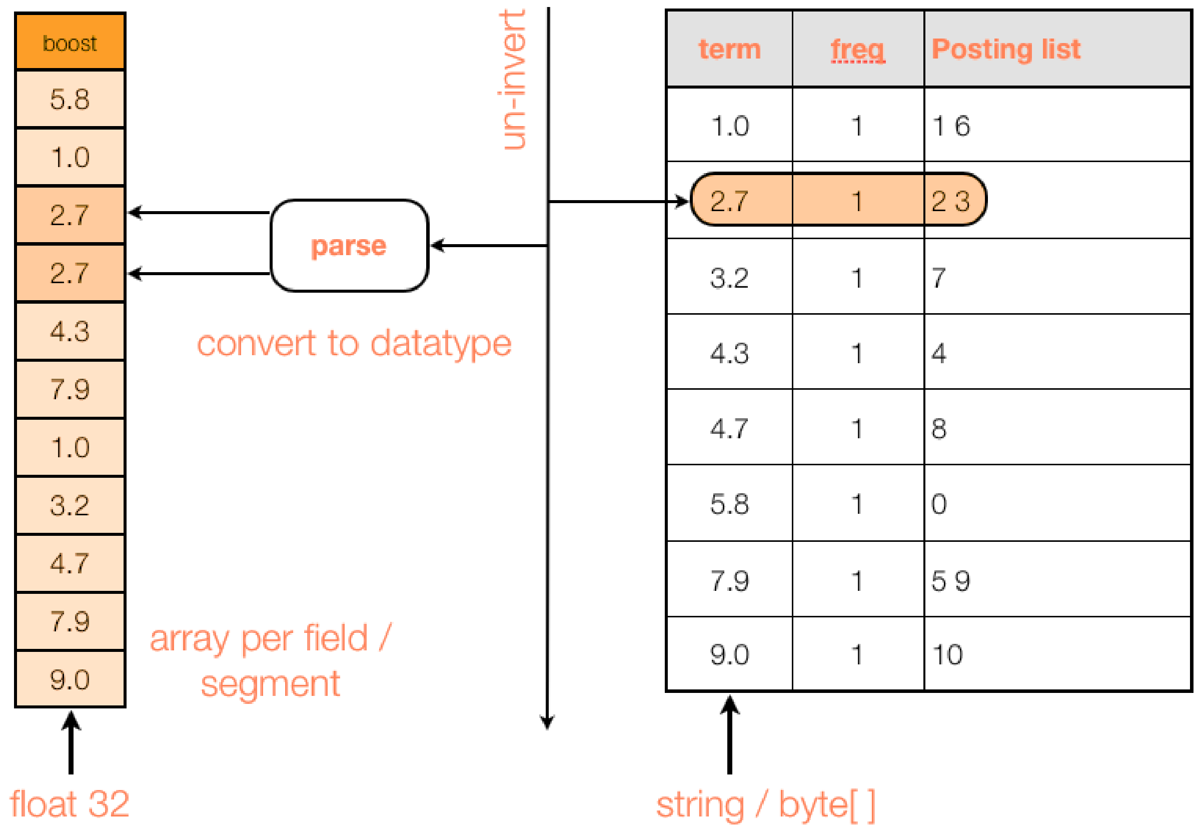
Lucene 5: CustomAnalyzer

- Freely configurable Analyzer

```
Analyzer ana =
CustomAnalyzer.builder(Paths.get("/path/to/config"))
  .withTokenizer("standard")
  .addTokenFilter("standard")
  .addTokenFilter("lowercase")
  .addTokenFilter("stop",
    "ignoreCase", "false",
    "words", "stopwords.txt",
    "format", "wordset")
  .build();
```

Die, FieldCache,… die, die, die!

- FieldCache is gone from Lucene Core



term	freq	Posting list
1.0	1	1 6
2.7	1	2 3
3.2	1	7
4.3	1	4
4.7	1	8
5.8	1	0
7.9	1	5 9
9.0	1	10

Image credits: Simon Willnauer / Trifork

Die, FieldCache,… die, die, die!

- FieldCache is gone from Lucene Core
- Use DocValues fields and APIs!

Die, FieldCache,… die, die, die!

- FieldCache is gone from Lucene Core
- Use DocValues fields and APIs!

- Not completely gone:
  - UninvertingReader in `misc/` module emulates DocValues by uninverting index
  - UninvertingReader allows to merge to a new index, automatically adding DocValues!

Future

ON THE WAY TO LUCENE 6…

Lucene 5.1: Filter => Query

Lucene 5.1: Filter => Query

- *(planned)* Removal of Filters
  - new `Occur.FILTER` in `BooleanQuery`
  - Removed some duplicate classes already: `BooleanFilter, Term(s)Filter, NumericRangeFilter`…

Lucene 5.1: Filter => Query

- *(planned)* Removal of Filters
  - new `Occur.FILTER` in `BooleanQuery`
  - Removed some duplicate classes already: `BooleanFilter, Term(s)Filter, NumericRangeFilter`…
- Backwards compatibility:
  - `Filter` extends `Query`
  - query API calls `getDocIdSet`
  - returns 0 as score (boost ignored)

Lucene 5.1: Two Phase Iterators

- Split iterators into *cheap* and *expensive* part

Lucene 5.1: Two Phase Iterators

- Split iterators into *cheap* and *expensive* part
- Used by `PhraseQuery`:
  - *Cheap* part is the „matching“ of terms (conjunction)
  - *Expensive* part is loading & checking positions

Lucene 5.1: Two Phase Iterators

- Split iterators into *cheap* and *expensive* part
- Used by `PhraseQuery`:
  - *Cheap* part is the „matching“ of terms (conjunction)
  - *Expensive* part is loading & checking positions
- Allows to share common code

Lucene 5.2: Span Queries

- Complete rewrite

Lucene 5.2: Span Queries

- **Complete rewrite**

- Uses Lucene 5.1 "two phase iterators"
- Shares code with `BooleanQuery` (conjunction / disjunction)

Lucene 5.2: Auto-Prefix Codec

- Moves `NumericRangeQuery` logic into codec
- More flexible „precisionStep“ (completely automatic based on terms distribution)

Lucene 5.2: Auto-Prefix Codec

- Works also with `TermRangeQuery`
- Will replace NRQ in Lucene 6…
  - Requires reindexing of numeric fields
  - no migration (at the moment)

Lucene 5.3+: NIO.2 again

More NIO.2:

- `LockFactory` was already refactored for 5.0

Lucene 5.3+: NIO.2 again

More NIO.2:

- `LockFactory` was already refactored for 5.0

- Take #2: bring file locking to next phase!
- Better remote file system support:
  - **CIFS/Samba** safety: `Lock.ensureValid()`
  - **NFS** ? Maybe – but it‘s still broken for commits…

THANK YOU!

Questions?

SD DataSolutions GmbH
Wätjenstr. 49
28213 Bremen, Germany
+49 421 40889785-0
http://www.sd-datasolutions.de