Apache Lucene 5

New Features and Improvements for Apache Solr and Elasticsearch

Uwe Schindler
Apache Software Foundation | SD DataSolutions GmbH | PANGAEA

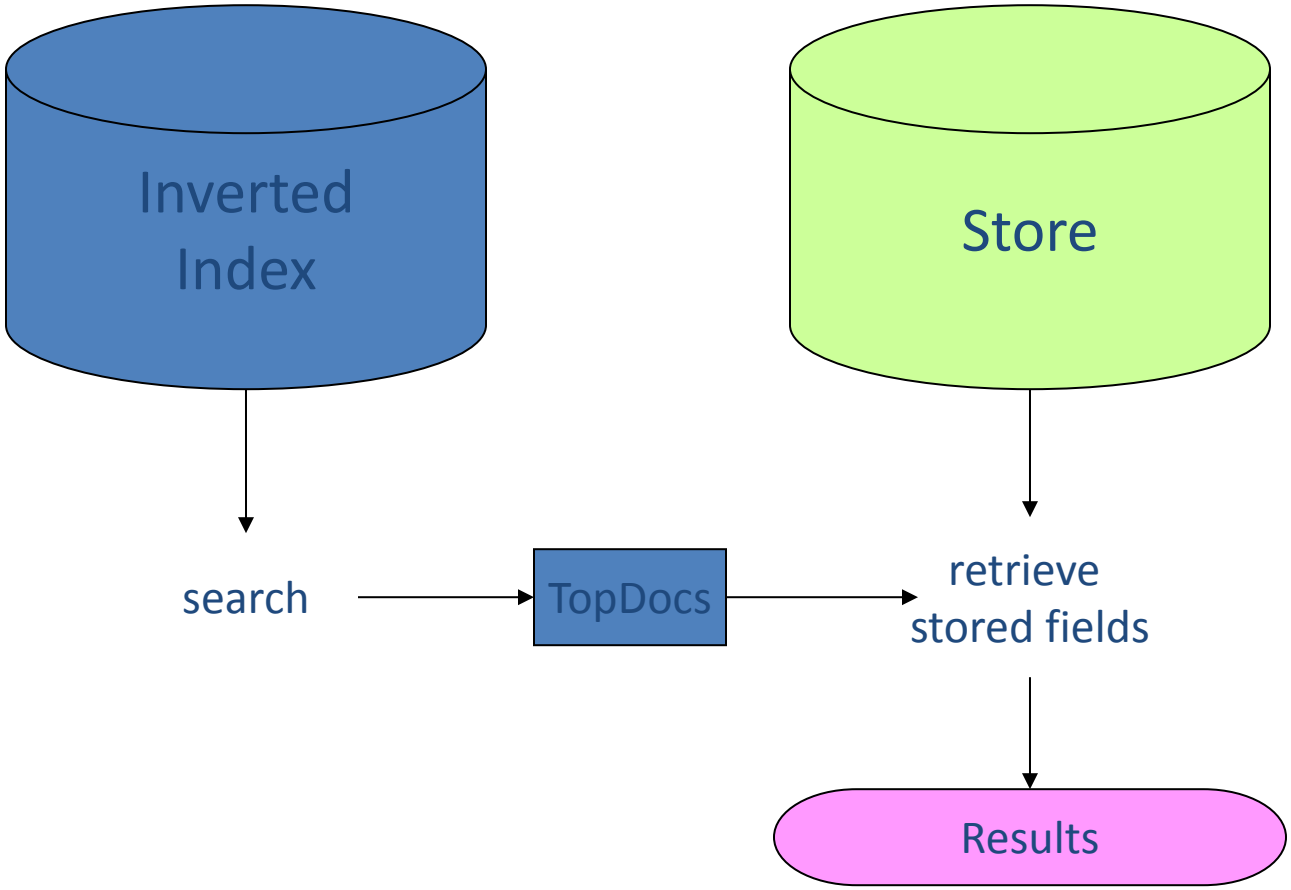
My Background

- 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.

An Overview

APACHE LUCENE ?

Lucene's data structures



c:\docs\einstein.txt:

The important thing is not to stop questioning.

c:\docs\shakespeare.txt:

To be or not to be.

Query: not

c:\docs\einstein.txt:

The important thing is not to stop questioning.

c:\docs\shakespeare.txt:

To be or not to be.

Query: not

c:\docs\einstein.txt:

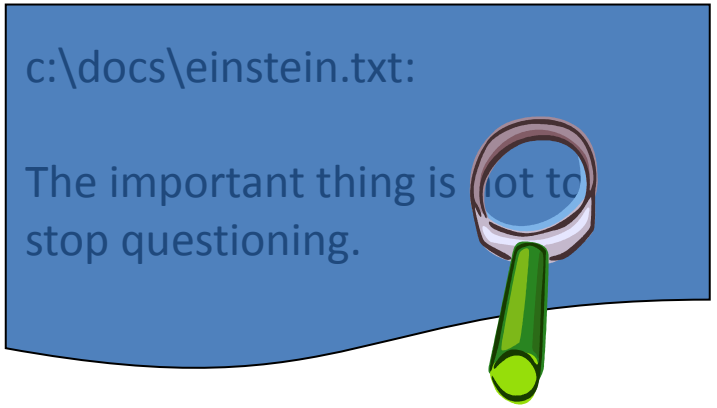
The important thing is not to stop questioning.

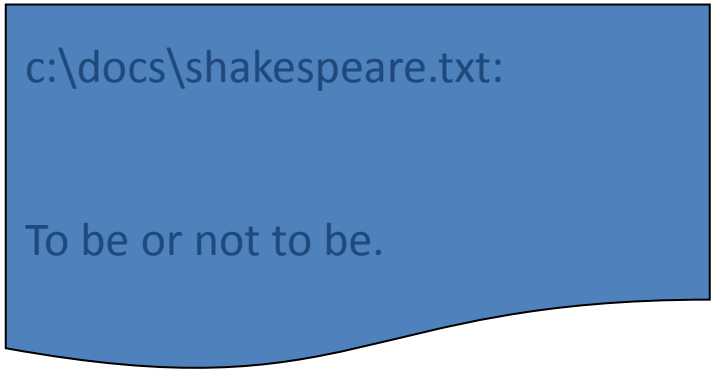
c:\docs\shakespeare.txt:

To be or not to be.

Query: not

String comparison slow!

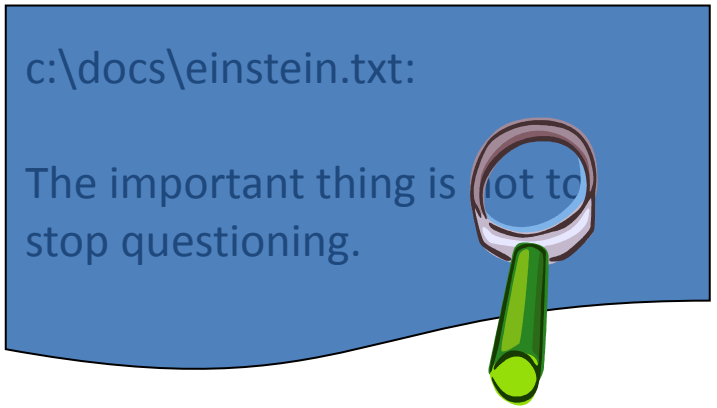


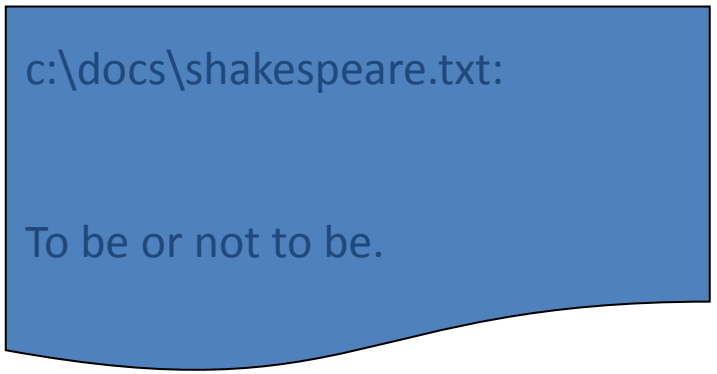


Query: not

String comparison slow!

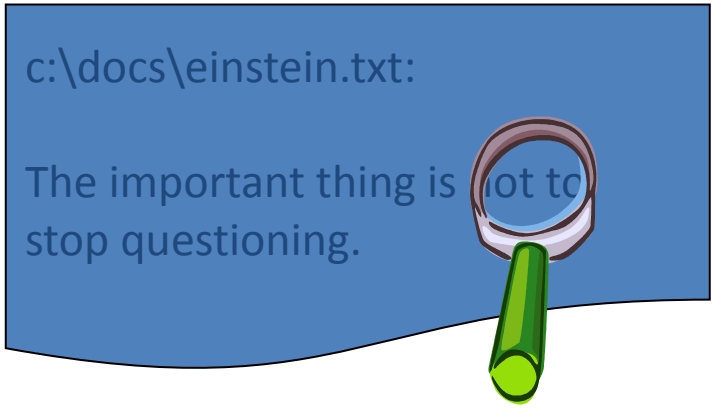
Solution: Inverted index

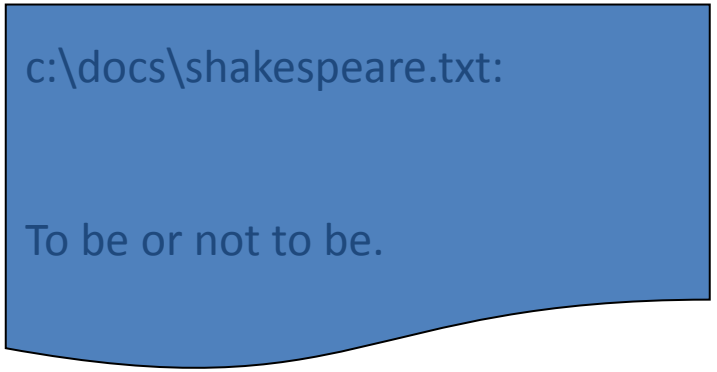




Inverted index

Query: not





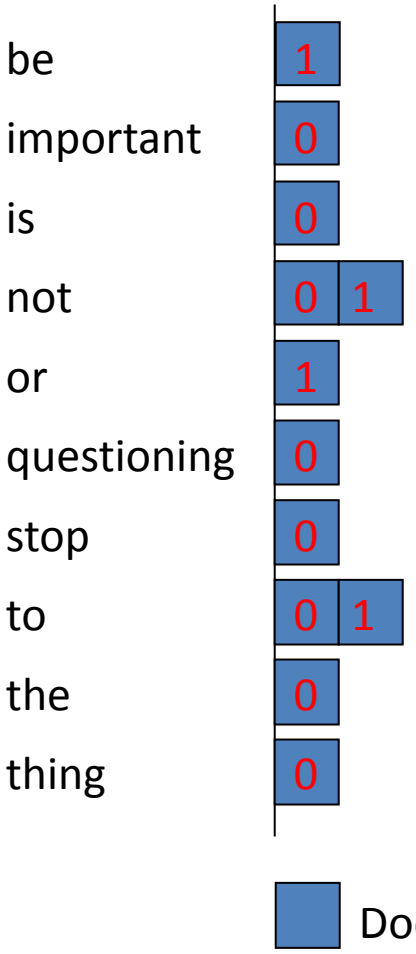
Inverted Index

Inverted Index

Inverted Index

Inverted Index

Inverted index



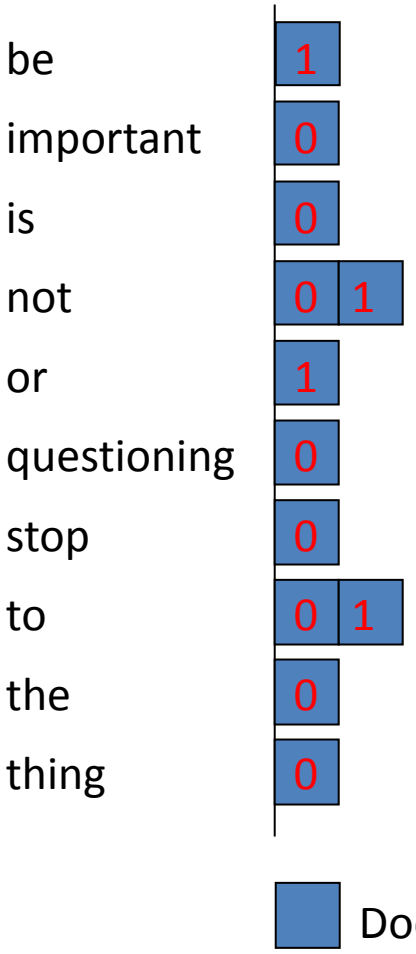
c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

c:\docs\shakespeare.txt: 1

To be or not to be.

Inverted index



Query: not

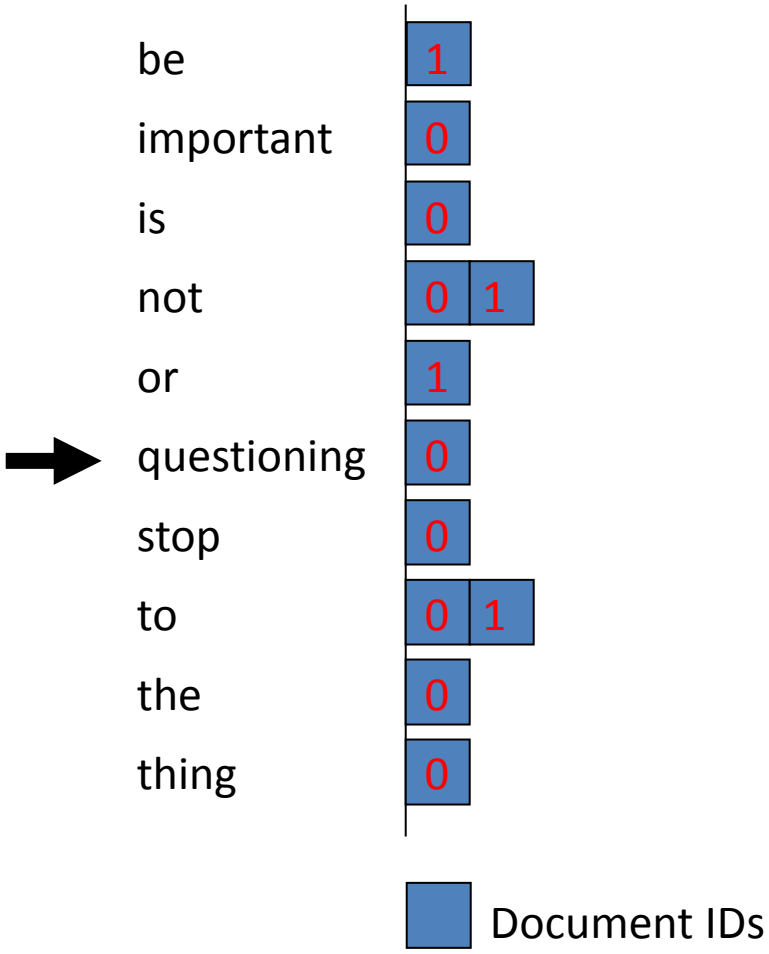
c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

c:\docs\shakespeare.txt: 1

To be or not to be.

Inverted index



Query: not

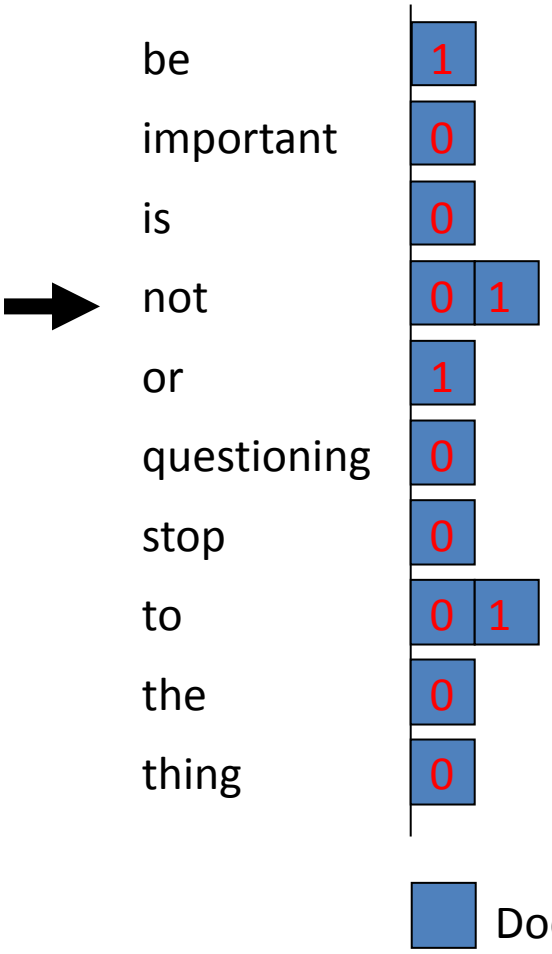
c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

c:\docs\shakespeare.txt: 1

To be or not to be.

Inverted index



Query: not

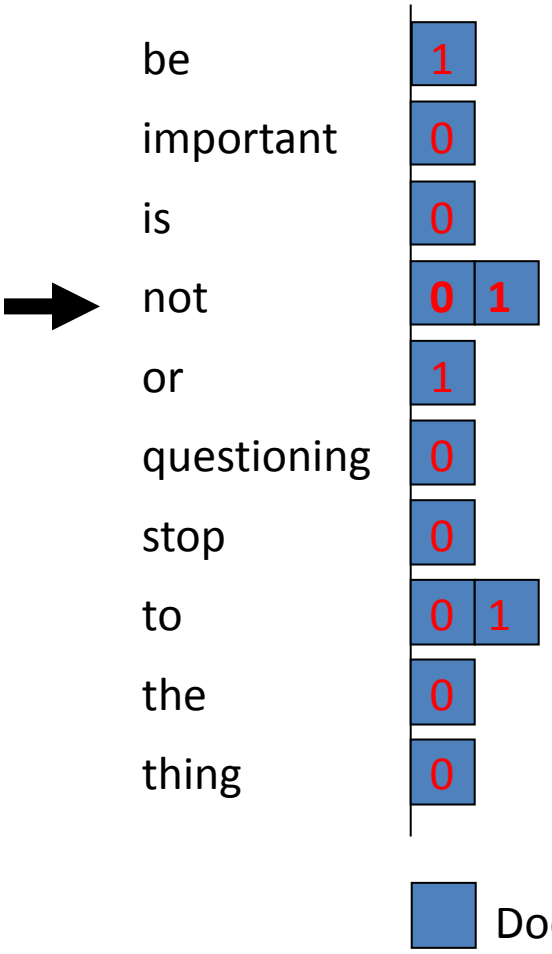
c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

c:\docs\shakespeare.txt: 1

To be or not to be.

Inverted index



Query: not

c:\docs\einstein.txt: 0

The important thing is not to stop questioning.

c:\docs\shakespeare.txt: 1

To be or not to be.

Information Retrieval Model

Lucene is based on a combination of two well known Information Retrieval models:

- Vector Space Model – scoring and relevance
- Boolean Model – narrowing down the documents to score

$$\cos\Theta = \frac{v_1 \cdot v_2}{|v_1| \cdot |v_2|}$$

$$w_{t,d} = tf_t \cdot idf_{t,d}$$

$$idf_{t,d} = \log \frac{|D|}{|t \in d|}$$

Term-Freque document d.

Inverse Document Frequency (idf) → the relation between the number of documents in the corpus and the number of documents containing term t (global parameter).

Indexing with Lucene

- Fast: over 200 GB/hour
- Incremental and “near-realtime”
- Multi-threaded
- Beyond full-text: numbers, dates, binary,...
- Customize what is indexed (“analysis”)
- Customize index format (“codecs”)

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
  - 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

- 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)*
- It's hard to add additional statistics for scoring to the index
  - IR researchers don't use Lucene to try out new algorithms

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)*
- It's hard to add additional statistics for scoring to the index
  - IR researchers don't use Lucene to try out new algorithms
- Small changes to index format are often huge patches covering tons of files

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

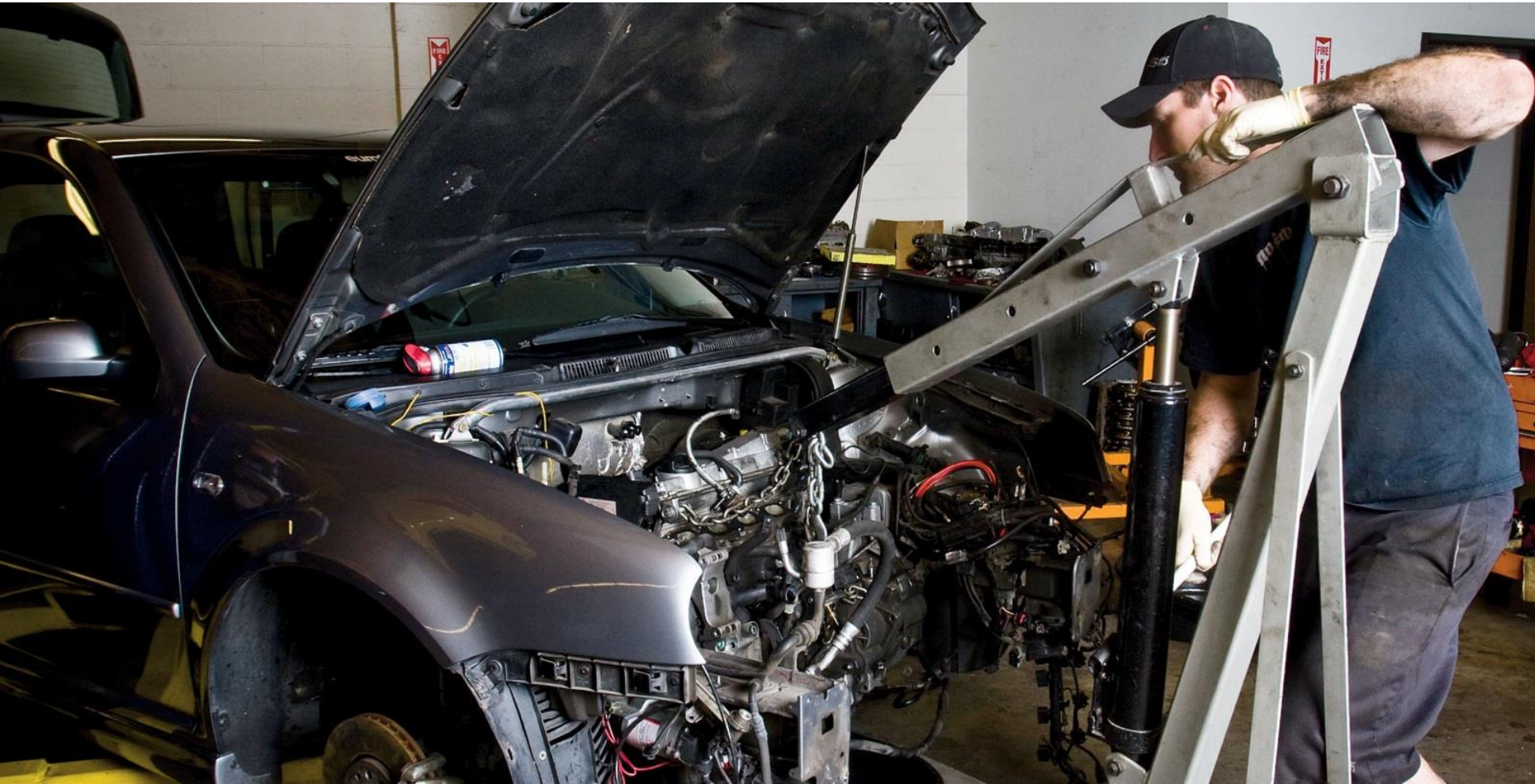
- Major release in October 2012
- New index engine:
  - **Codec** support *(pluggable via SPI)*
  - **DocValues** fields
- New relevancy models: not only TF/IDF !
  - e.g., **BM25**

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

Complete overhaul of all APIs



History: Apache Lucene 4

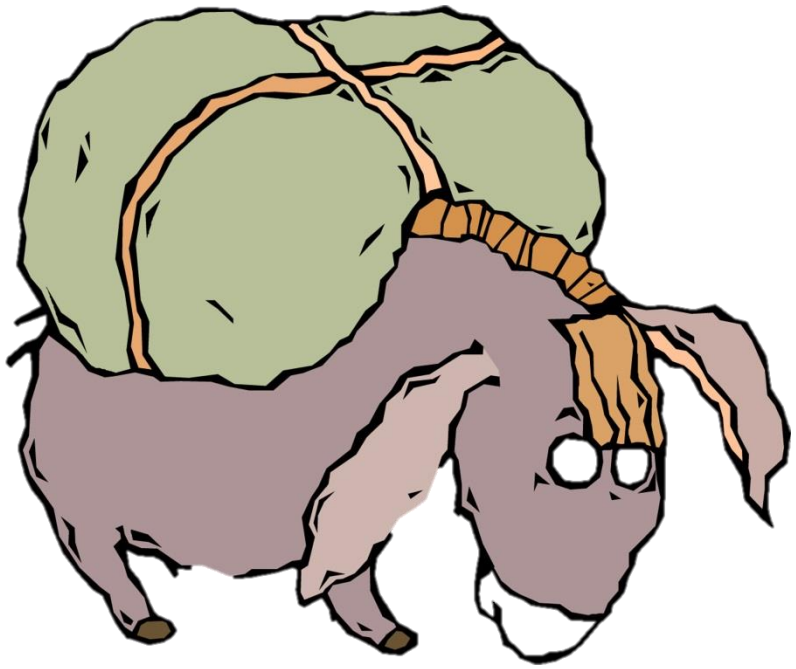
Complete overhaul of all APIs

- 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!!!

- Burden of maintaining the old stuff:
  - old index formats
  - especially support for **Lucene 3.x** indexes



On-going Disasters

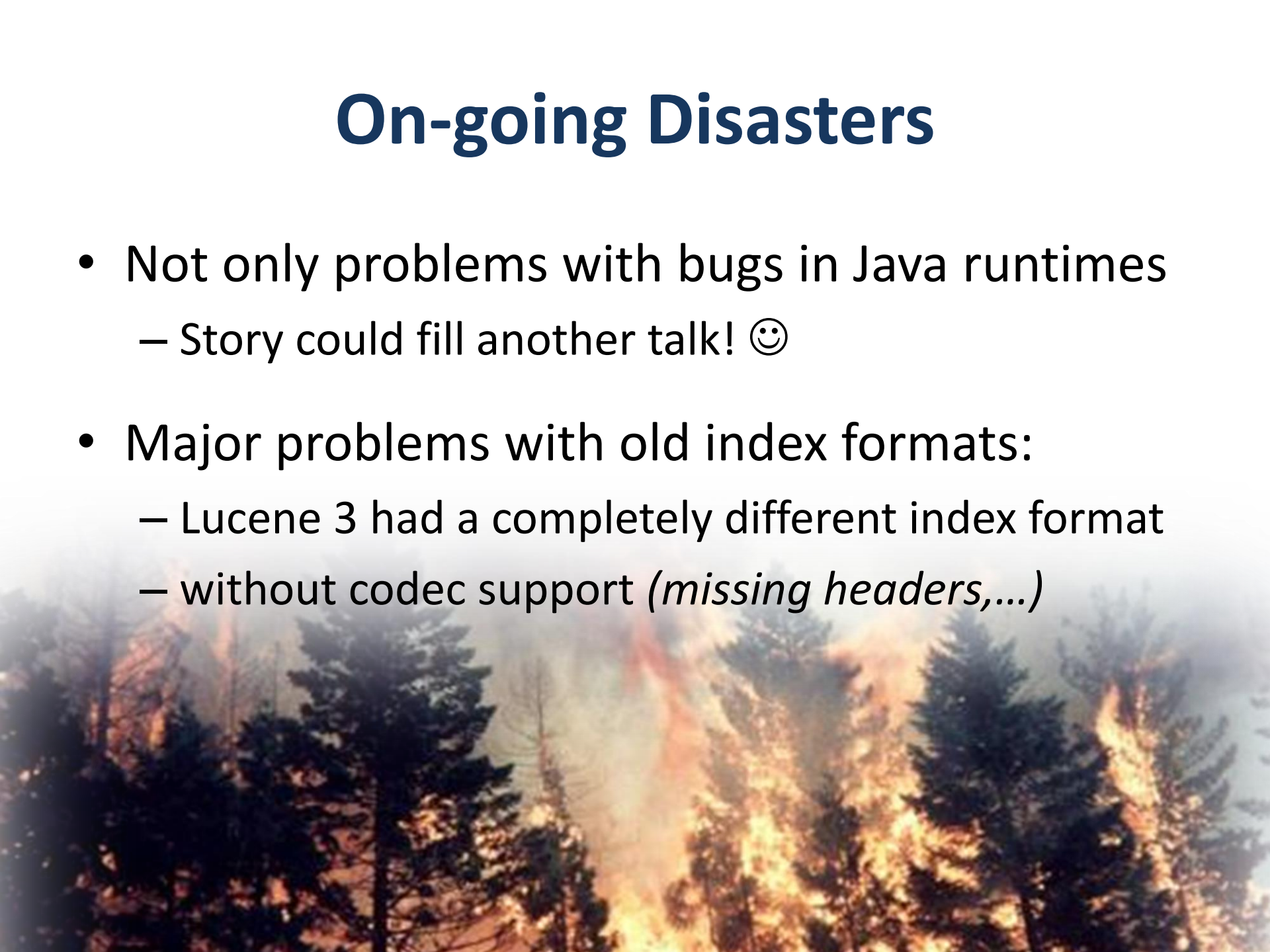
- Not only problems with bugs in Java runtimes

On-going Disasters

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

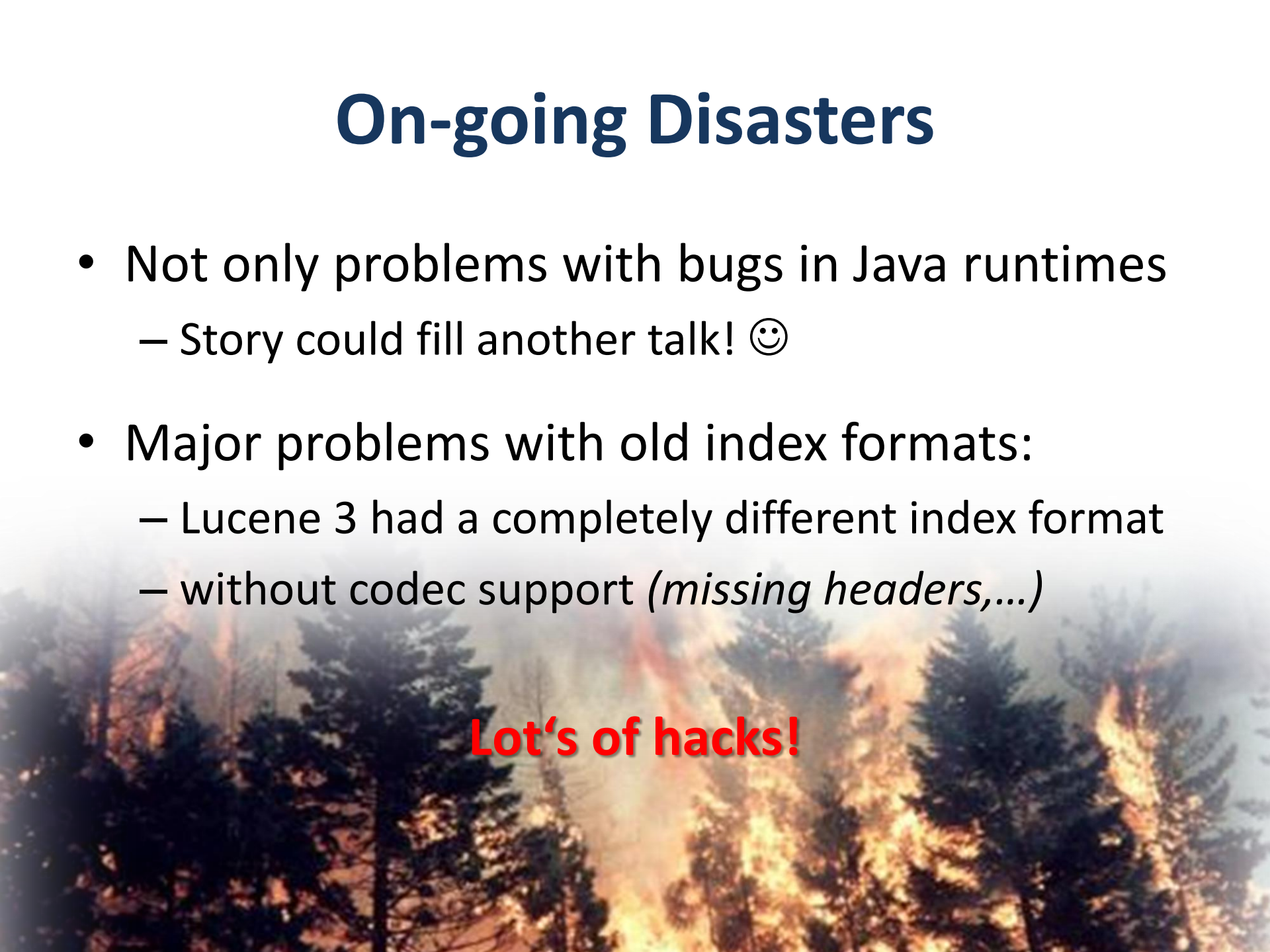
On-going Disasters

- 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)*

Apache Lucene 5

A lot new features!

Apache Lucene 5

A lot new features!

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

Apache 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

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!

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!
- **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)

Lucene 5: New index safety features

Cutover to NIO.2 *(Java 7, JSR 203)*

atomic rename to publish commit

`fsync()` on index directory

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 index safety!**

Java 7 support (Lucene 4.8+)

Java 7 support (Lucene 4.8+)

- Try-With-Resources
  - Nice, but we had it already implemented: IOUtils.closeWhileHandlingExceptions()

Java 7 support (Lucene 4.8+)

- Try-With-Resources
  - Nice, but we had it already implemented: IOUtils.closeWhileHandlingExceptions()
- Some syntactic sugar ☺

Java 7 support (Lucene 4.8+)

- Try-With-Resources
  - Nice, but we had it already implemented: IOUtils.closeWhileHandlingExceptions()
- Some syntactic sugar ☺
- Partial implementation of NIO.2 for FSDirectory
  - *(allows to delete open files on Windows!)*

Java 7 support (Lucene 4.8+)

- Try-With-Resources
  - Nice, but we had it already implemented: IOUtils.closeWhileHandlingExceptions()
- Some syntactic sugar ☺
- Partial implementation of NIO.2 for FSDirectory
  - *(allows to delete open files on Windows!)*
- `MethodHandle` / `ClassValue` for Tokenization API's internals
  - Huge speedup for dynamic instantiation of token Attributes, especially in Java 8!

Java 7 support (Lucene 4.8+)

Java 7u55+ has

no serious bugs anymore

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

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://code.google.com/p/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://code.google.com/p/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

- Don‘t 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)

Java 7 NIO.2 - Consequences

- Don‘t 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)
- All other file I/O is now channel based (or MMap)
  - If cancelled throws `ClosedByInterruptException`
  - also `SimpleFSDirectory` !

Java 7 NIO.2 - Consequences

- Don‘t 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)
- All other file I/O is now channel based (or MMap)
  - If cancelled throws `ClosedByInterruptException`
  - also `SimpleFSDirectory` !
- Use `Paths.get()` while opening `DirectoryReader` / `IndexWriter`
  - Alternative: use `File.toPath()`

Lucene 5.0: Overhaul of Codec API

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

Lucene 5.0: Index merging

Lucene 5.0: Index merging

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

Lucene 5.0: 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 5.0: 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 5.0: 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 5.0: 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 5.0: 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 5.0: 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
- 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!

What‘s new

Solr

Apache Solr 5.0

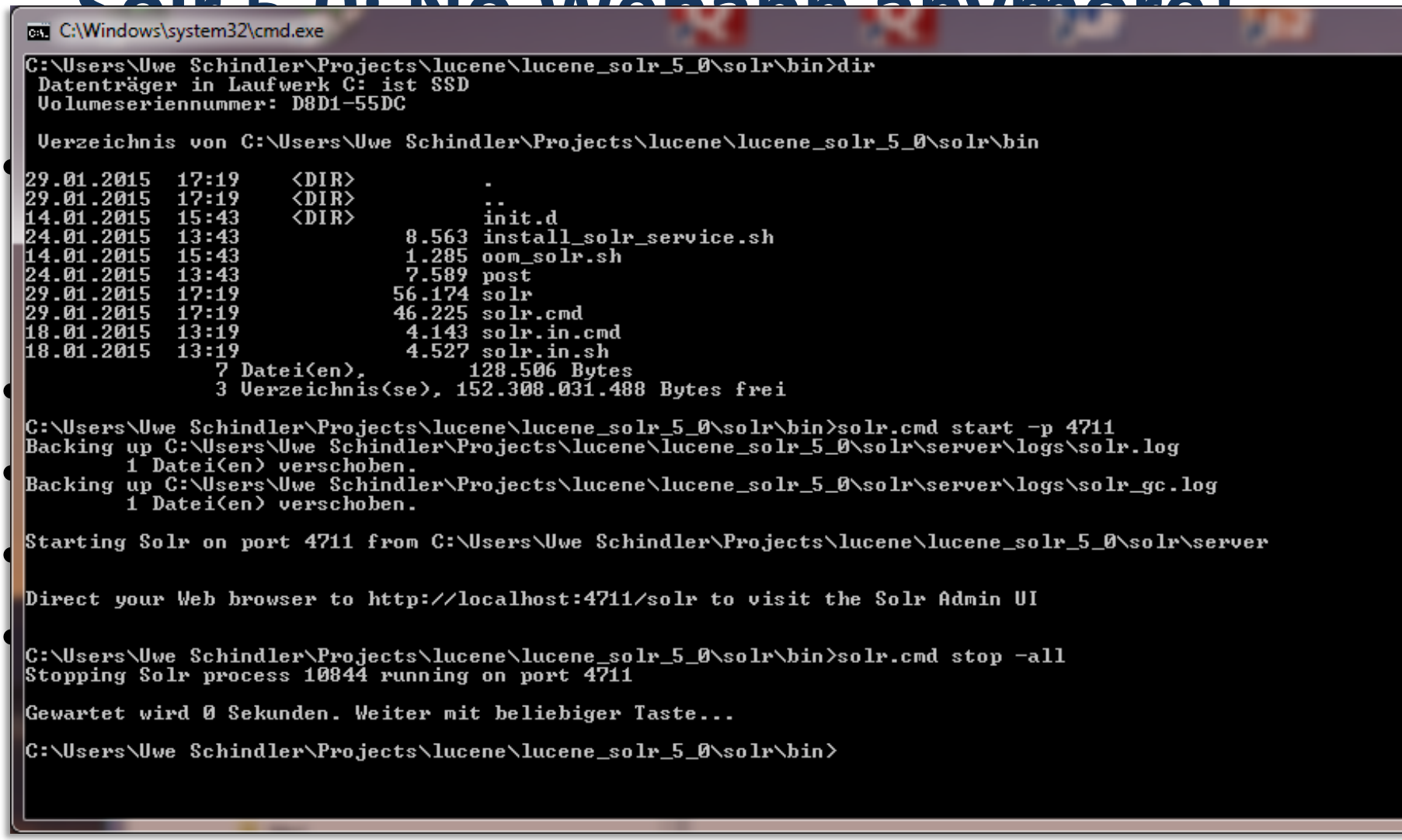
New release bundled
with Lucene 5.0 release

Improved fault tolerance

Solr 5.0: No Webapp anymore!

- Solr ships as server software
  - like MySQL, PostgreSQL,…
  - or Elasticsesarch ☺
- Start/Stop scripts for SysVinit
- JVM tuning by default
- Scripts to create collections
- No "official" WAR file anymore
  - Maven
  - Download distribution

Solr 5.0: No Webapp anymore!



Solr 5.0: Distributed IDF

Support for distributed *Inverse Document Frequency:*

- Makes use of caching of IDF from other nodes
- Several caching implementations

Solr 5.0: Distributed IDF

Support for distributed *Inverse Document Frequency:*

- Makes use of caching of IDF from other nodes
- Several caching implementations

Should only be used if exact scoring is really needed

- If documents are not well (randomly) distributed

Solr 5.0: Config API

- Makes parameters of RequestHandlers configurable
- Allows to change RequestHandlers
- Upload of plugin JARs

Solr 5.0: Other features

- Bandwidth control for index replication
- BLOBs API
- SolrJ improvements:
  - Rename SolrServer to SolrClient
  - Support of Collections API
- Split Clusterstate
  - Scales better for hundreds of nodes

THANK YOU!

Questions?

Contact

Uwe Schindler
uschindler@apache.org
http://www.thetaphi.de
@thetaph1

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