What’s new in Lucene 4.0

Simon Willnauer @ BerlinExpertDays 2011
Committer Apache Lucene
simonw@apache.org / simonw@jteam.nl

~ THE CONFERENCE OF HIGH SCALABILITY ~

>BERLINBUZZWORDS<

June 6th / 7th 2011 in Berlin

JTEAM

What’s new in Lucene 4.0

Agenda

- ‣ Flexible Indexing
- - Realtime Search
- - Automaton Query
- - PerDocument Payloads aka. Column-Stride Fields

5 cool things you can do with Lucene 4.0

Agenda

- Unicode

‣ Flexible Indexing

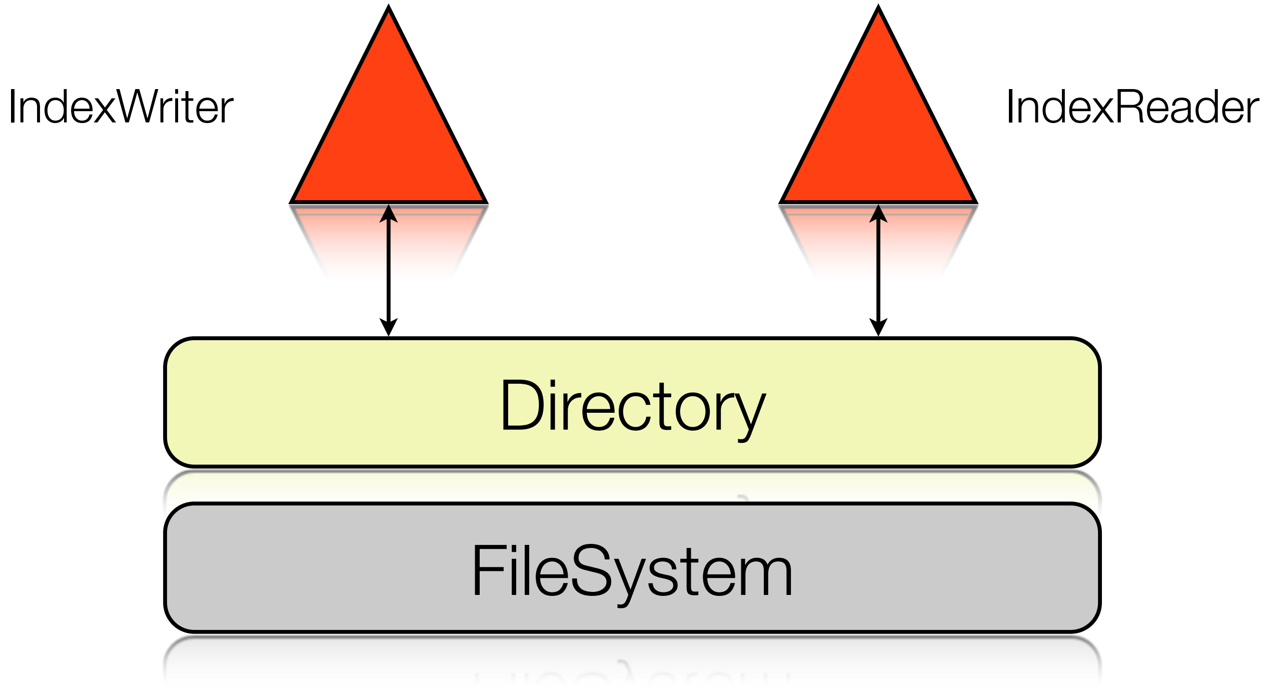
- Realtime Search

- Automaton Query

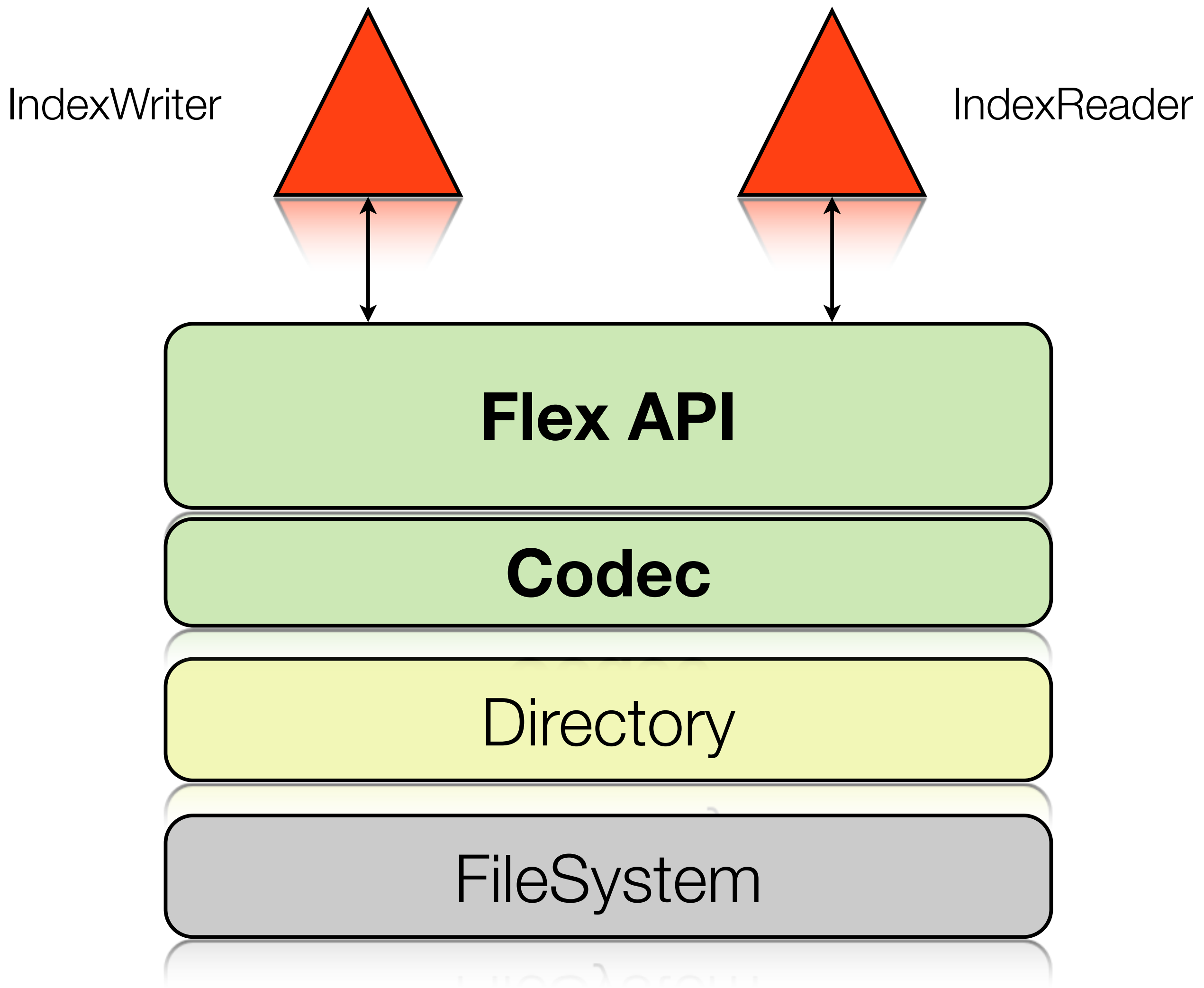
- PerDocument Payloads aka. Column-Stride Fields

JTEAM

The big picture - without Flexible Indexing



Flexible Indexing - The big picture



JTEAM

Flexible Indexing - Motivation

- Extending Lucene on the lowest level was impossible for non-Lucene devs
- Playing with new ideas like different posting list formats
    - Variable Int encoding is slow and shows its age
- Changes like omitTFAP required expert knowledge and changed lots of core code
- Research in the IR community like scoring models or index data structures

Inverted Index 101

1	The old night keeper keeps the keep in the town
2	In the big old house in the big old gown.
3	The house in the town had the big old keep
4	Where the old night keeper never did sleep.
5	The night keeper keeps the keep in the night
6	And keeps in the dark and sleeps in the light.

Table with 6 documents

Example from:
Justin Zobel , Alistair Moffat,
Inverted files for text search engines,
ACM Computing Surveys (CSUR)
v.38 n.2, p.6-es, 2006

JTEAM

Inverted Index 101

1	The old night keeper keeps the keep in the town
2	In the big old house in the big old gown.
3	The house in the town had the big old keep
4	Where the old night keeper never did sleep.
5	The night keeper keeps the keep in the night
6	And keeps in the dark and sleeps in the light.

Table with 6 documents

Dictionary and posting lists for a single field

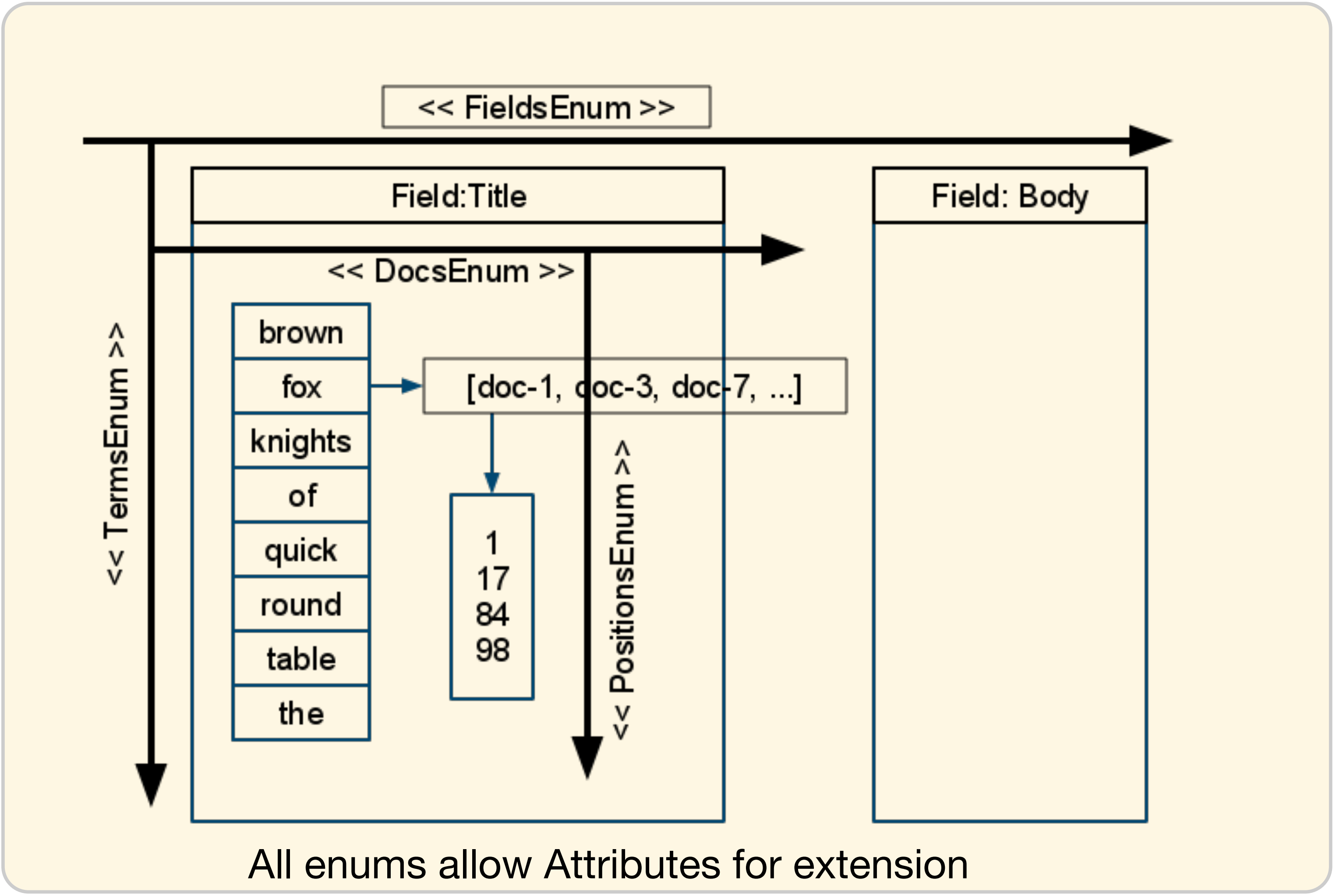
term	freq	Posting list
and	1	6
big	2	2 3
dark	1	6
did	1	4
gown	1	2
had	1	3
house	2	2 3
in	5	<1> <2> <3> <5> <6>
keep	3	1 3 5
keeper	3	1 4 5
keeps	3	1 5 6
light	1	6
never	1	4
night	3	1 4 5
old	4	1 2 3 4
sleep	1	4
sleeps	1	6
the	6	<1> <2> <3> <4> <5> <6>
town	2	1 3
where	1	4

TermsEnum $\downarrow$

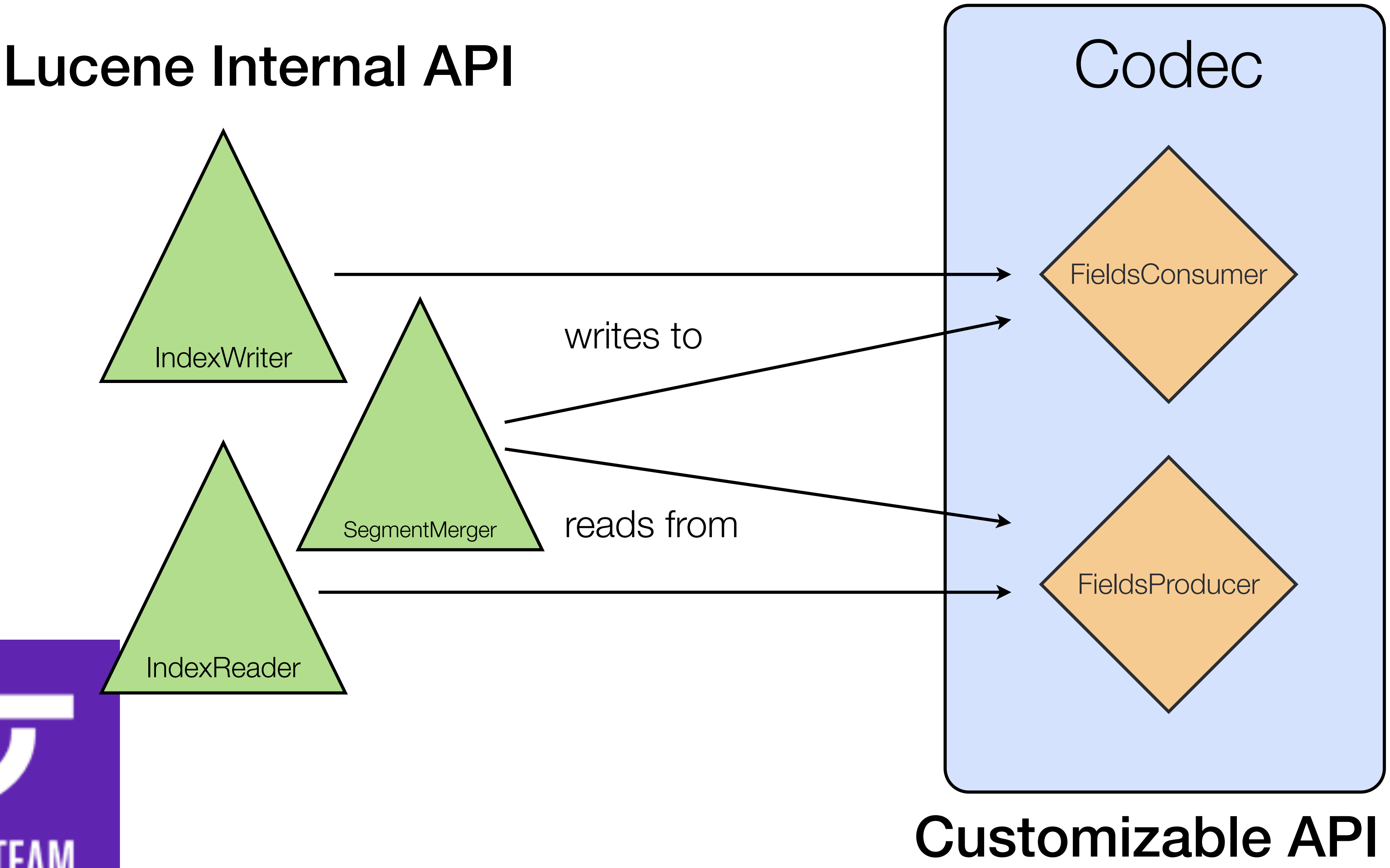
DocsEnum $\longrightarrow$

JTEAM

Flexible Indexing - A 4-dimensional API

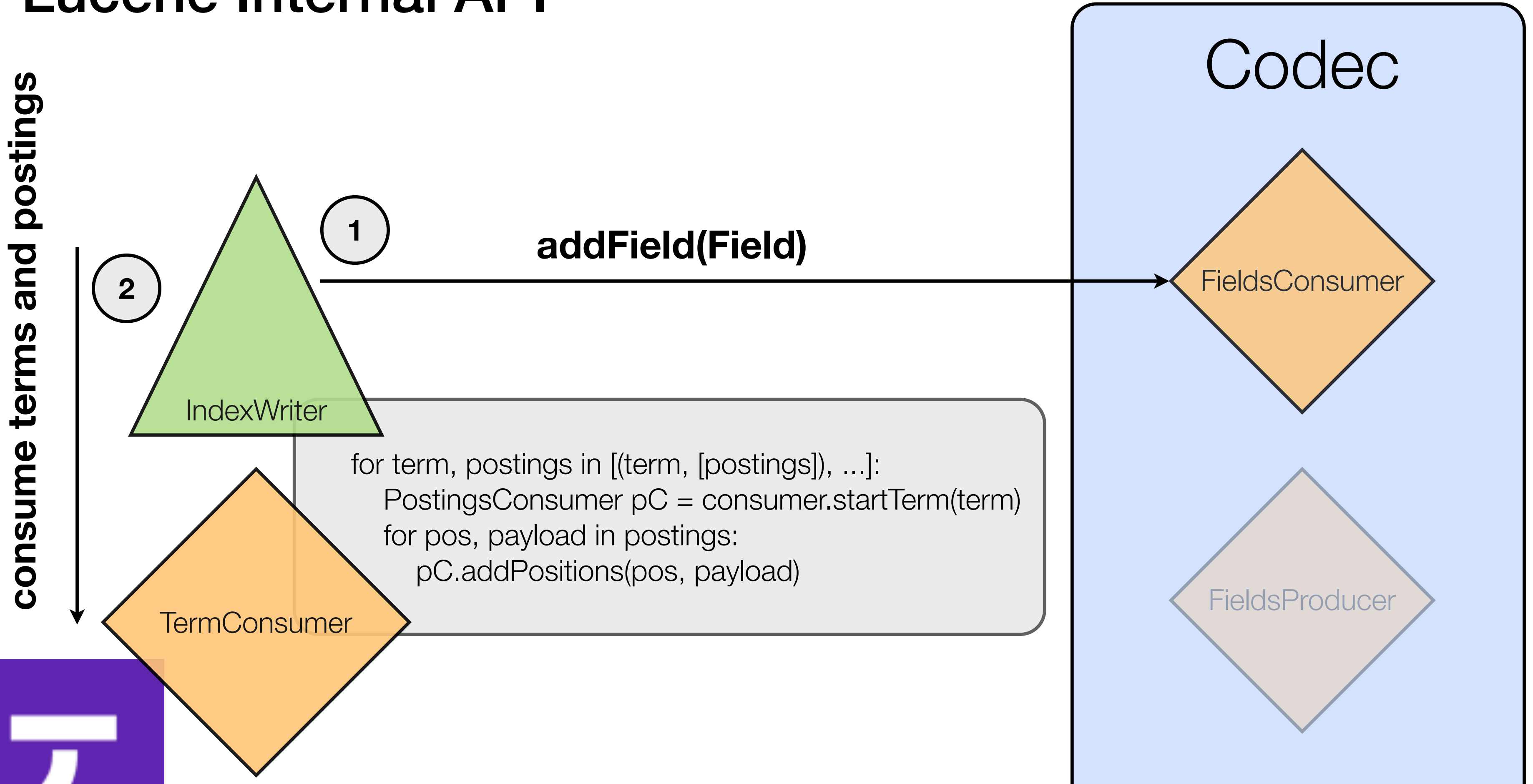


Introducing Codec



Introducing Codec

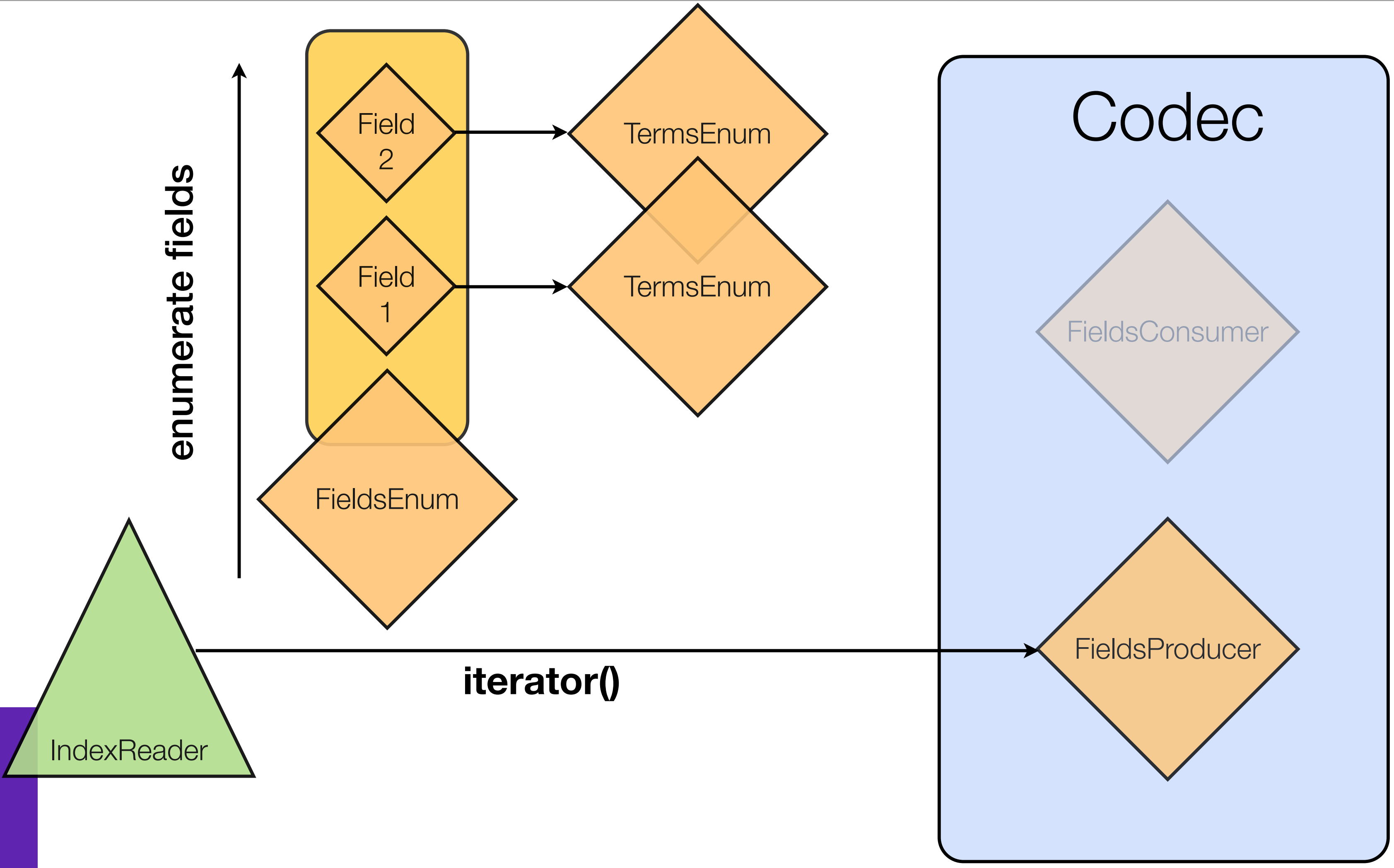
Lucene Internal API



```
for term, postings in [(term, [postings]), ...]:
    PostingsConsumer pC = consumer.startTerm(term)
    for pos, payload in postings:
        pC.addPositions(pos, payload)
```

Customizable API

Introducing Codec



Customizable API

Flex API - Some internals

- Separates Terms from Fields
- Full binary Term representation - can use any encoding (default is UTF-8)
- TermsEnum allows Random Access (plus seek by ord)
- Term sort order is determined by the Codec
- Codecs are per Field and Segment
- Codecs define data structures and RAM requirements per field

Core Codecs

- **Standard Codec with PrefixCoded - TermIndex and VInt based postings**
  - HatihTrust TermIndex with 2.2 M indexed terms (Source Mike Mccandless)
    - Lucene 3.x: 3974 MB RAM, 72.8 sec to load
    - Lucene 4.0: 401 MB RAM, 2.2 sec to load – 9.9 X less RAM, 33X faster
- **Pulsing Codec - inlines low frequent terms into the Term dictionary**
  - 20% - 50% speedup for term lookups

More on Codecs

- Simple Text Codec for debugging and learning
  - Writes plain text
- PFOR / FOR / VSEncoding based Codecs promise further improvements over VarInt
- Special codec to write directly to HDFS
- Many more to come!

JTEAM

What’s new in Lucene 4.0

Agenda

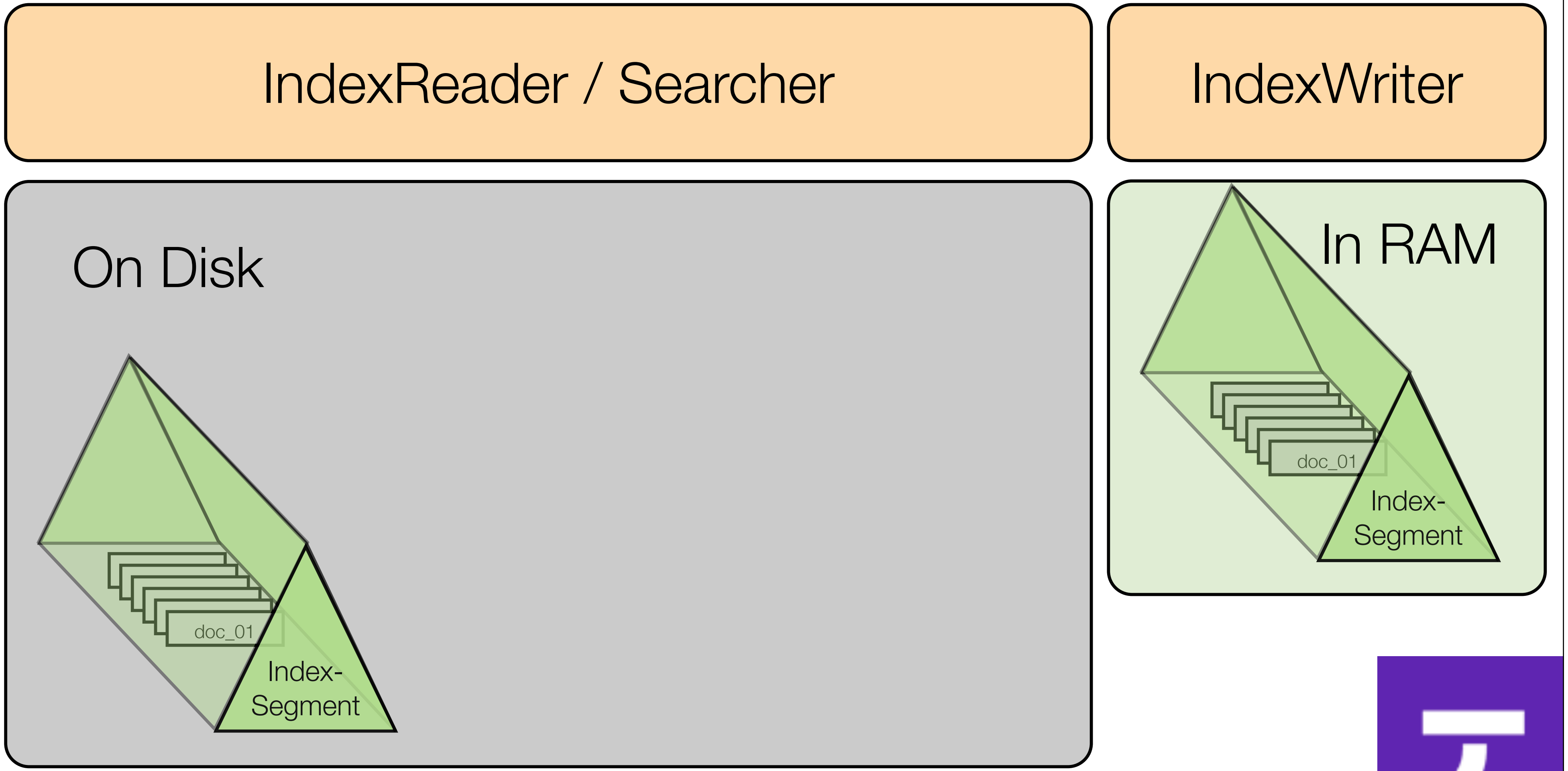
- Flexible Indexing
- ‣ Realtime Search
- Automaton Query
- PerDocument Payloads aka. Column-Stride Fields

JTEAM

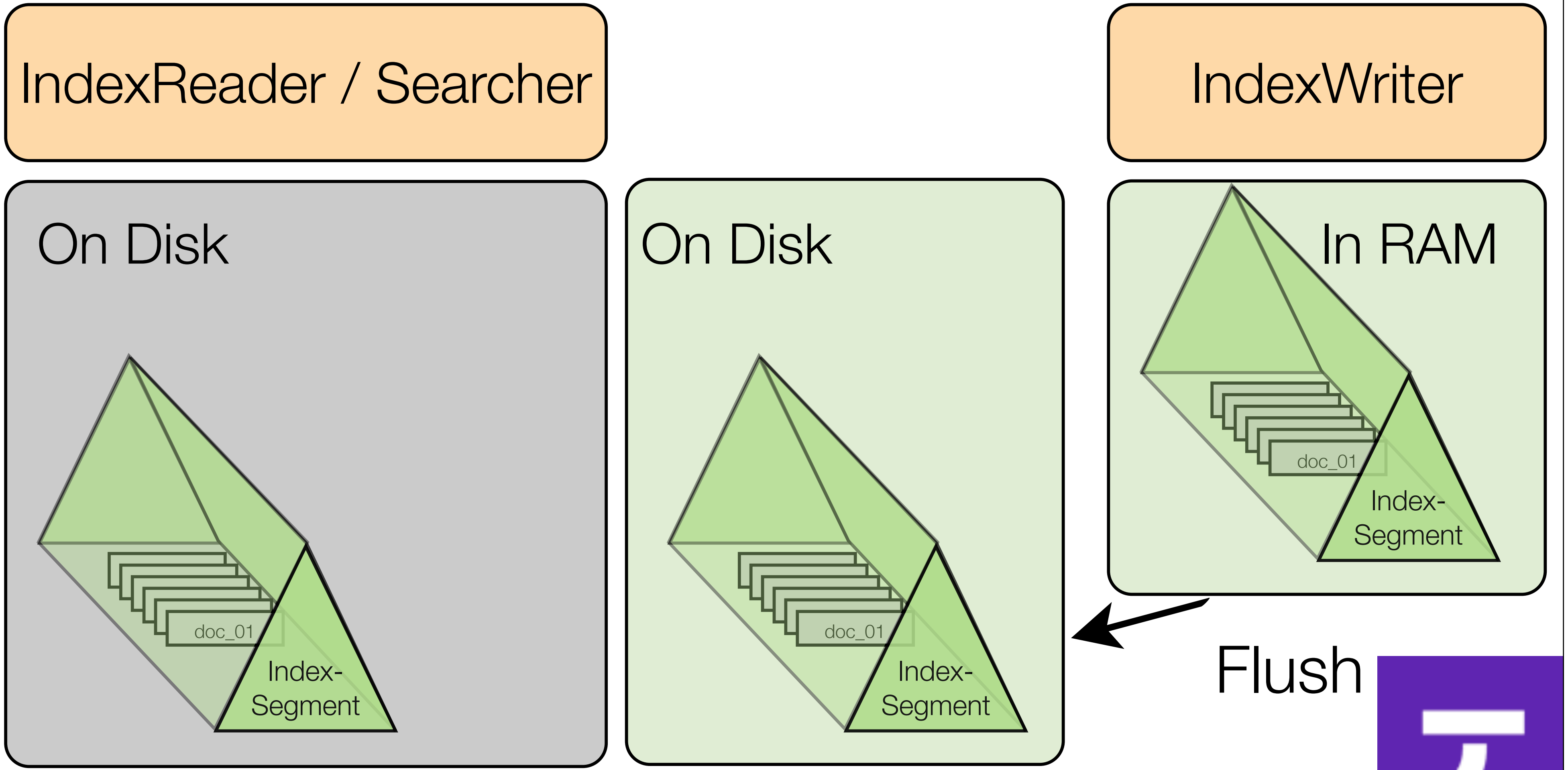
Realtime Search - current state

- **Flushing a segment is expensive**
  - Syncs FS Caches and can trigger merges
  - Writing the RAM buffer to disc is single threaded
- **Reopen IndexReader can be expensive too**
  - loading all segments that have changed (thanks to PerSegment-Search)
  - purge FieldCache per segment
  - Search speed will suffer badly if done too often

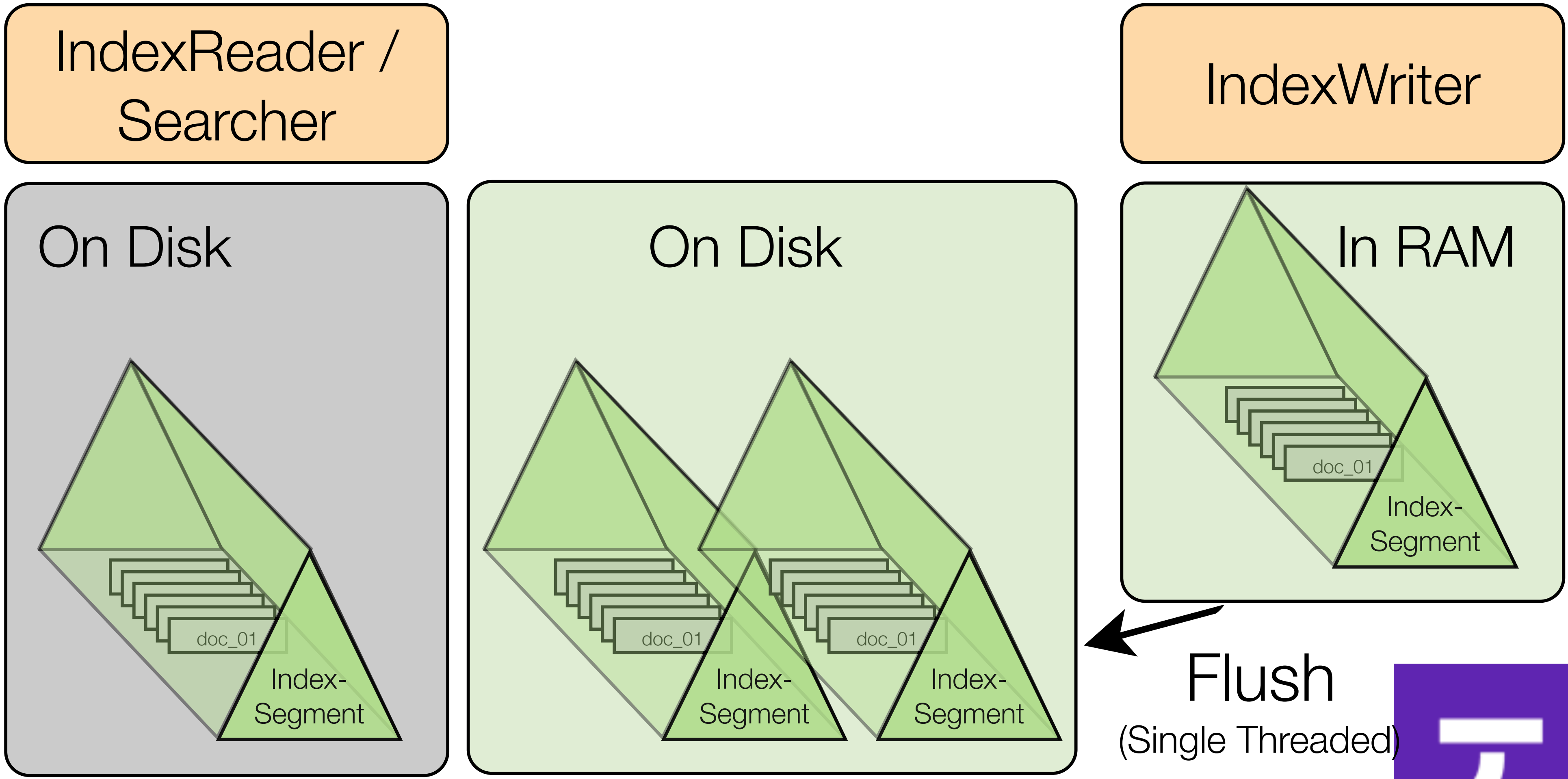
Realtime Search - Indexing 101



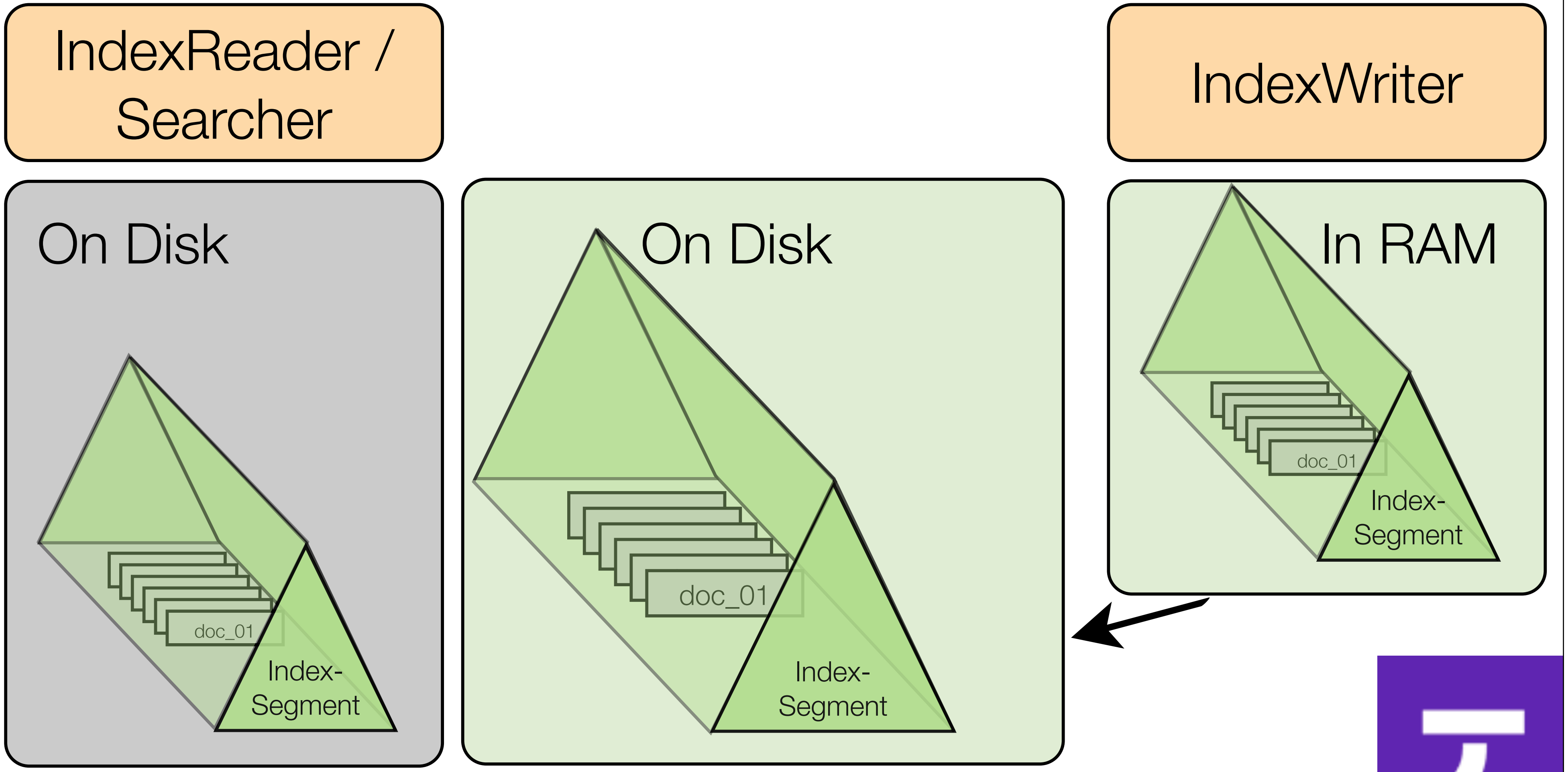
Realtime Search - Indexing 101



Realtime Search - Indexing 101



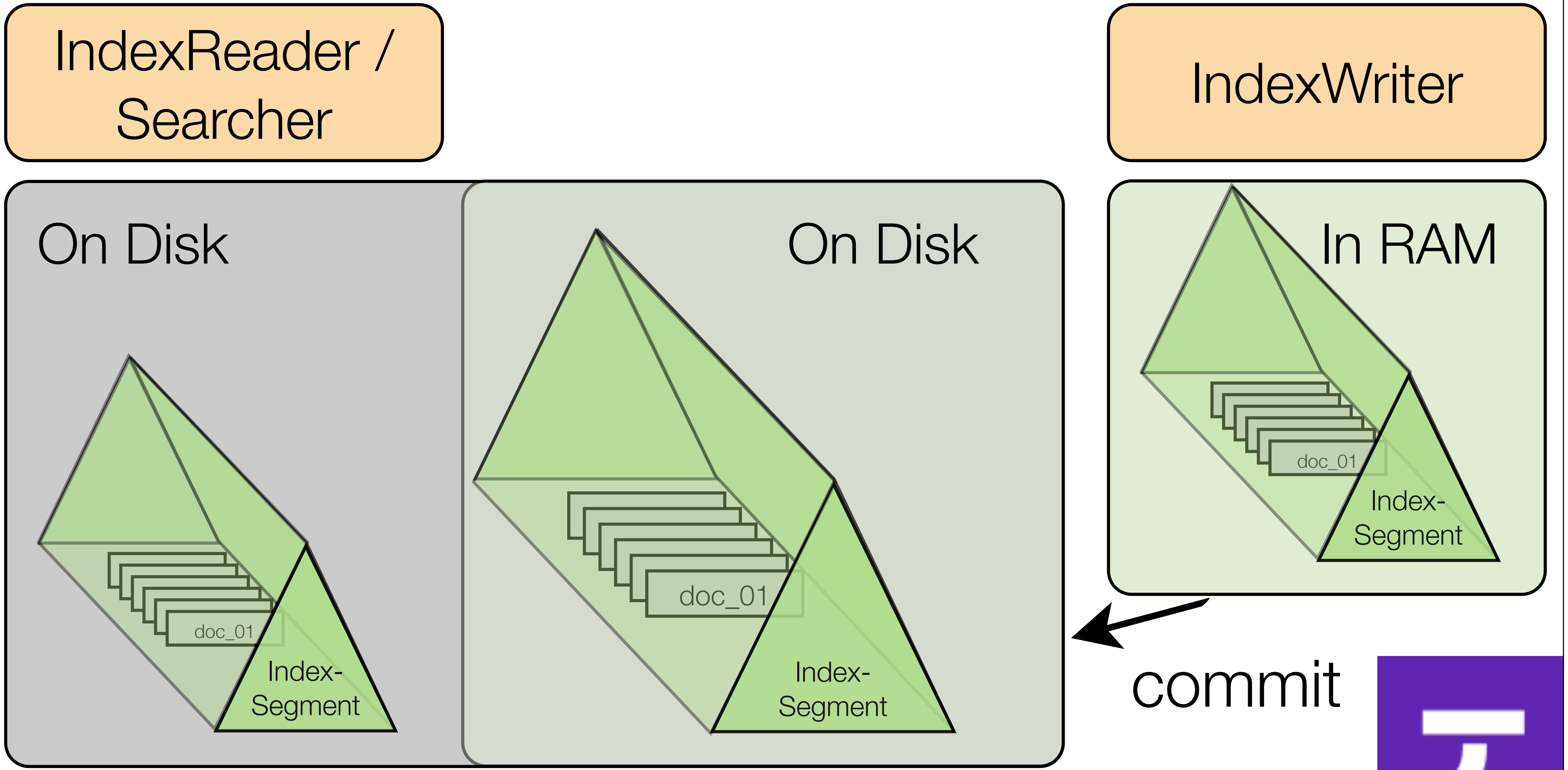
Realtime Search - Indexing 101



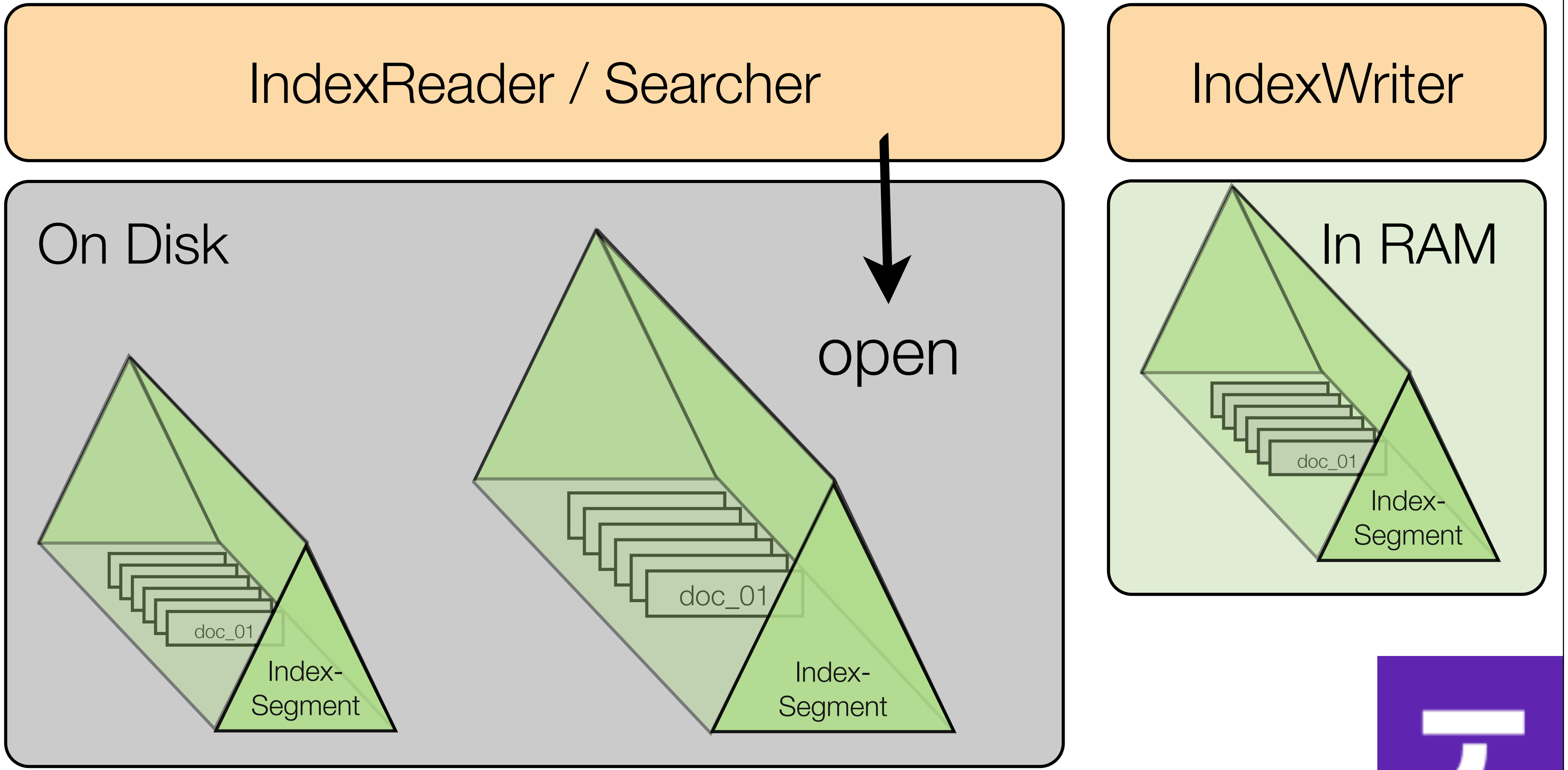
flush merge

(MergeFactor 3)

Realtime Search - Indexing 101



Realtime Search - Indexing 101

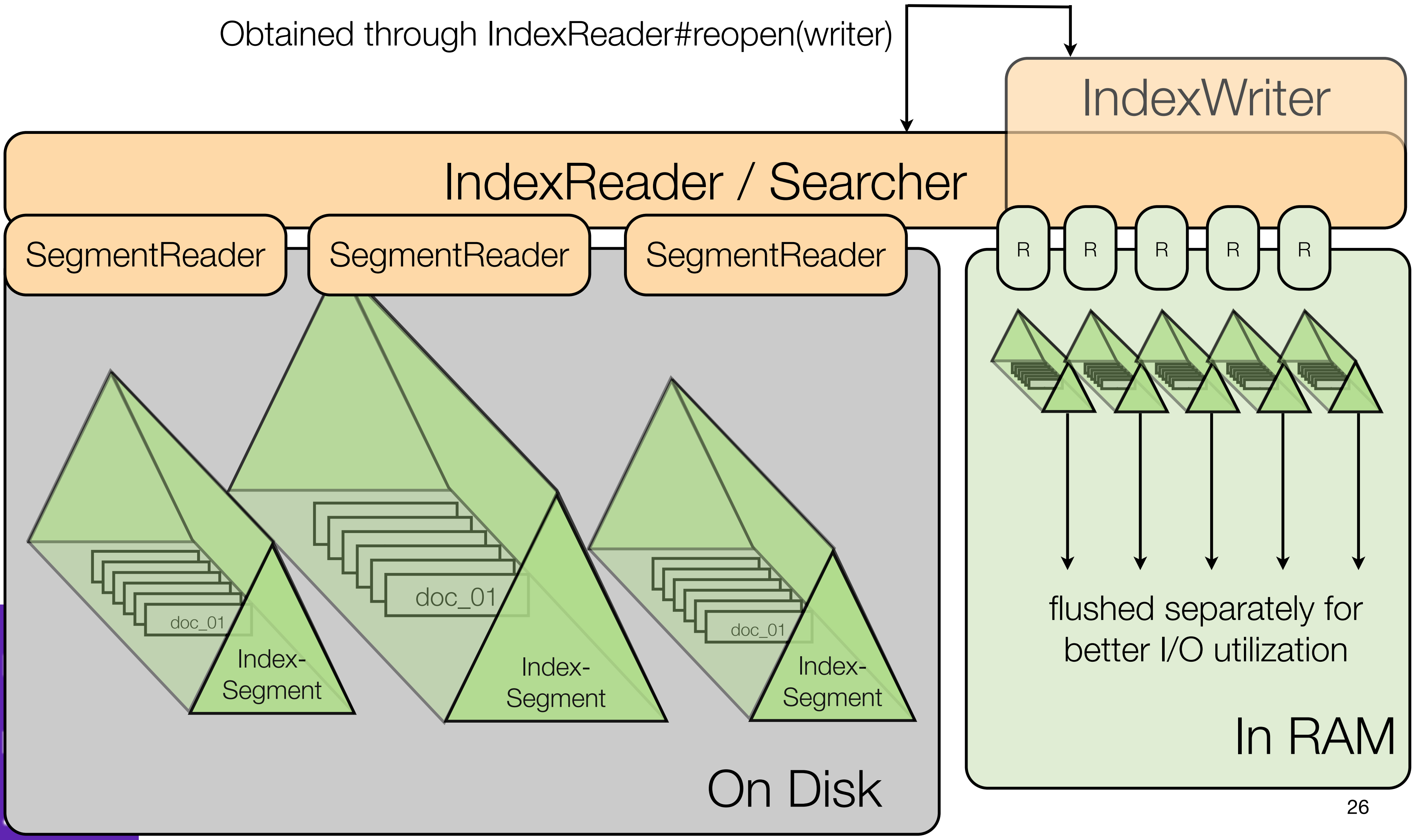


Realtime Search - the improvements

- **Indexing Threads write their own private segments (LUCENE-2324)**
  - Each indexing thread writes its own segment on disc
  - will exploit full CPU / IO concurrency
- **Direct Searchable DocumentWriter's RAM Buffer**
  - Reopen a IndexReader becomes super cheap
  - Index / Searchable latency drops close to zero

JTEAM

Realtime Search - The rough picture



Realtime Search - already committed improvement

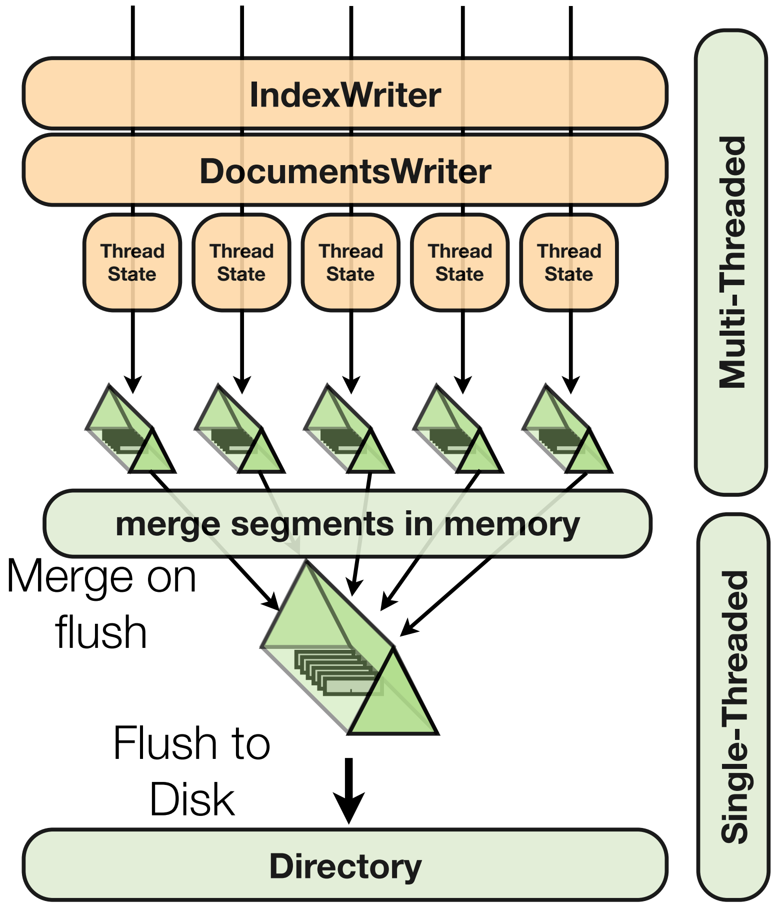
ParallelPostingsArray - LUCENE-2329

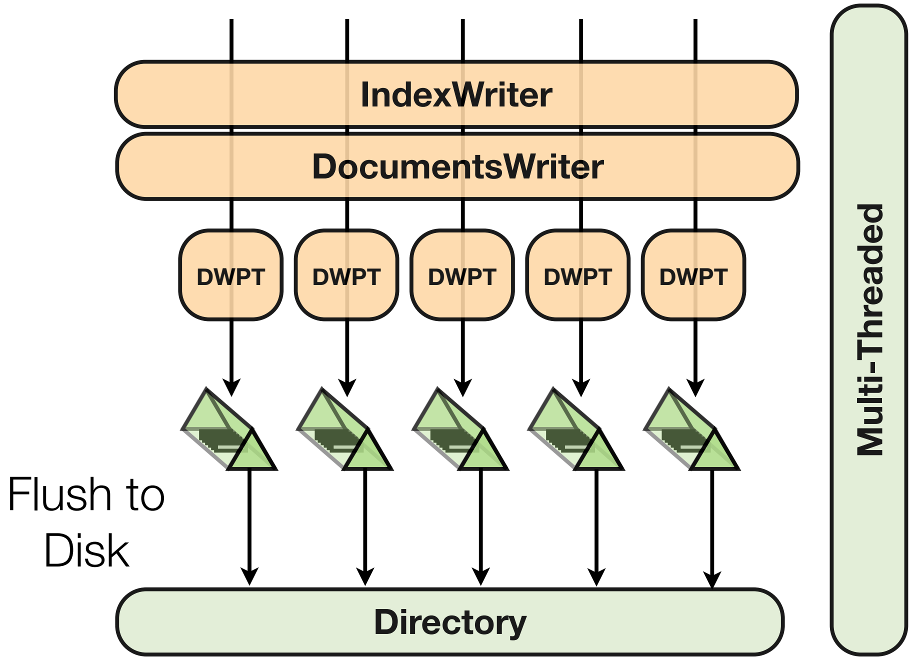
```
class PostingList {
   int textPointer;
   int postingsPointer;
   int frequency;
}
```

```
class ParallelPostingsArray {
  int[] text;
  int[] postings;
  int[] frequencies;
}
```

- Instead of PostingList[] use ParallelPostingsArray
- Reduces the number of long living objects dramatically
- Dramatic speed improvements when memory is tight (up to 400% according to buschmi@apache.org)

Realtime Search - DocumentsWriterPerThread

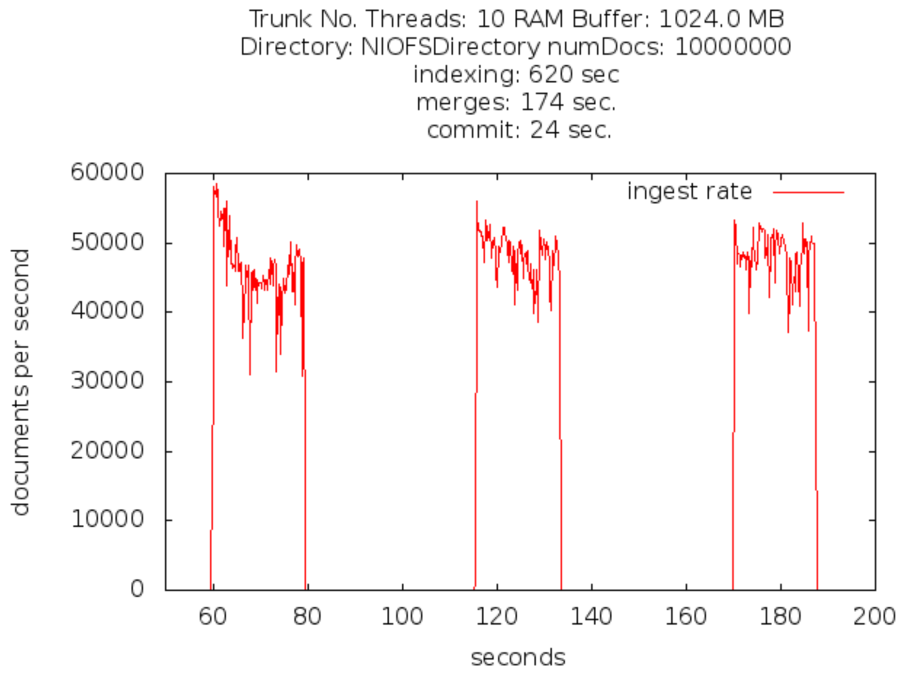


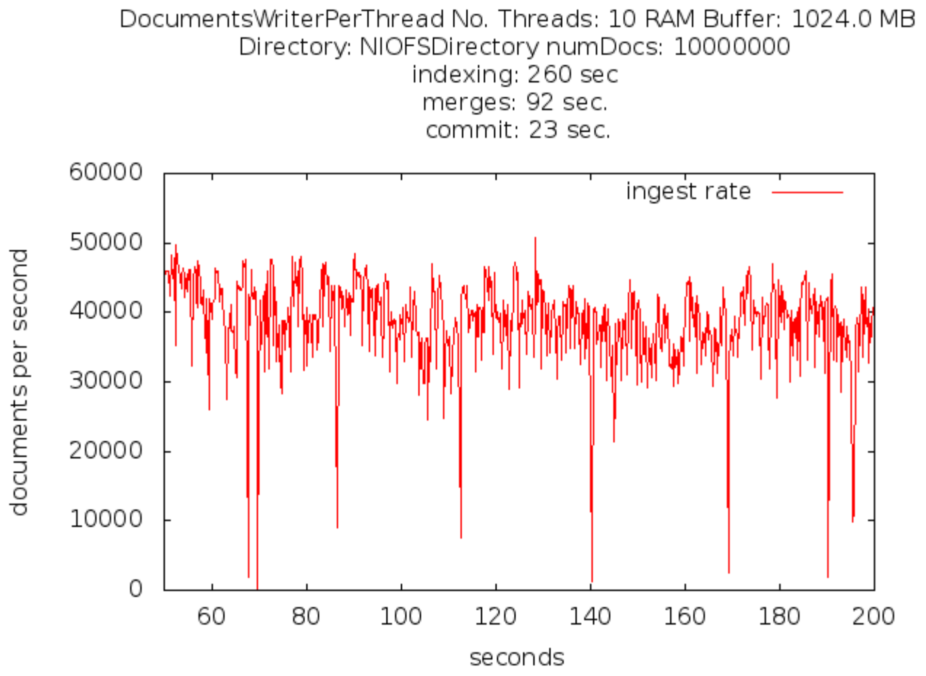


Current Model

Realtime Branch

Realtime Search - Ingest Rate Comparison





Current Model

13 min 40 sec

Realtime Branch

6 min 15 sec

Realtime Search - stay tuned

- **Active Development on “realtime” branch**
  - http://svn.apache.org/repos/asf/lucene/dev/branches/realtime
- **Preliminary Results promise extreme improvements**
  - Michael Busch: *“at Twitter we open a billion IndexReaders per day!”*
    - LUCENE-2346: Change in-memory postinglist format
    - LUCENE-2312: Search on DocumentsWriters RAM buffer

JTEAM

What’s new in Lucene 4.0

Agenda

- Flexible Indexing
- Realtime Search
- ‣ Automaton Query
- PerDocument Payloads aka. Column-Stride Fields

Automaton Query

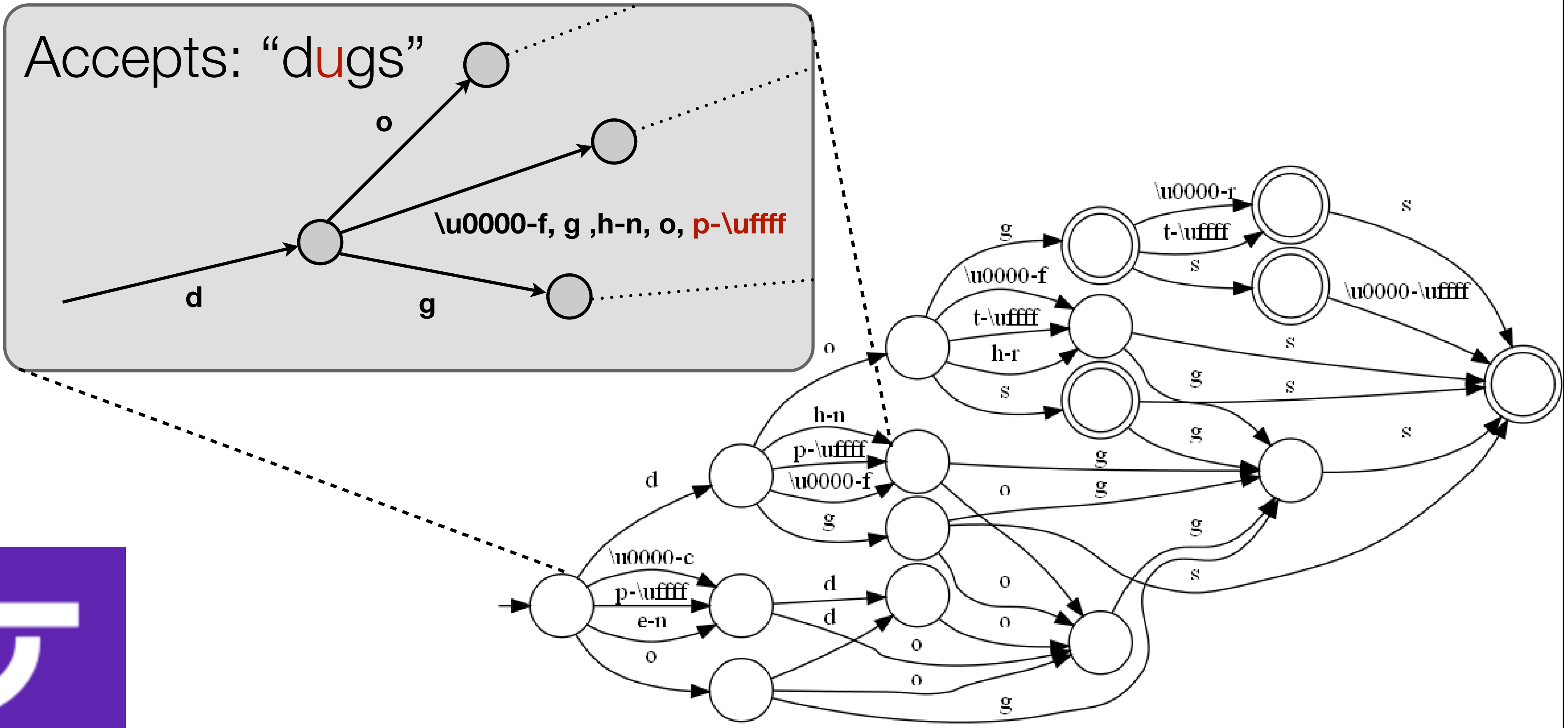
- Enables **FAST** inexact queries like
  - Regexp-, Wildcard- and FuzzyQyery
- Build as a FSM traversed in parallel with the term dictionary
- Operates on Per-Segment level to prevent unnecessary term lookups

Automaton Query - Example FuzzyQuery

- FuzzyQuery - finds terms within a certain String-Distance (Levenshtein)
  - Historically **very very very** slow - unusable on large dictionaries
  - Without constant prefix LD (Levenshtein Distance) for every term was calculated.
- New FuzzyQuery builds a Levenshtein Automaton and intersects it with the term dictionary.
- Flex - TermEnums provided necessary Random-Access API for fast seeks

Automaton Query - Example FuzzyQuery

Example FSM for the term “dogs~1”



Automaton Query - Benchmark Results

Query	QPS (3.x)	QPS (4.0)	Pct diff
united~0.6	0.41	24.70	5858.2%
united~0.7	0.44	94.77	21454.8%

Optimized 7M Document Wikipedia index
Dual 6 Core Xeon / 12GB RAM

What’s new in Lucene 4.0

Agenda

- Flexible Indexing
- Realtime Search
- Automaton Query
- PerDocument Payloads aka. Column-Stride Fields

JTEAM

PerDocument Payloads (LUCENE-2186)

- Dense column based per document storage (Typed Payload per Field / Doc)
- Extends Stored Field functionality
  - Fast for loading all fields
  - Slow for loading single fields
- Replaces weird Payload-Base storage tricks
- Enables Scoring with others than Norms (Boost)

JTEAM

PerDocument Payloads - Types

- More than just strings
- Variable length Integers (via PackedInts)
- Fload32 and Float64
- Fixed / Variable Length Bytes
- Deferred / Straight Bytes
- Sorted Byte Variants

PerDocument Payloads - Rough Picture

Number of bit depend on the numeric range in the field:

```
Math.max(1, (int) Math.ceil(
          Math.log(1+maxValue)/Math.log(2.0))
          );
```

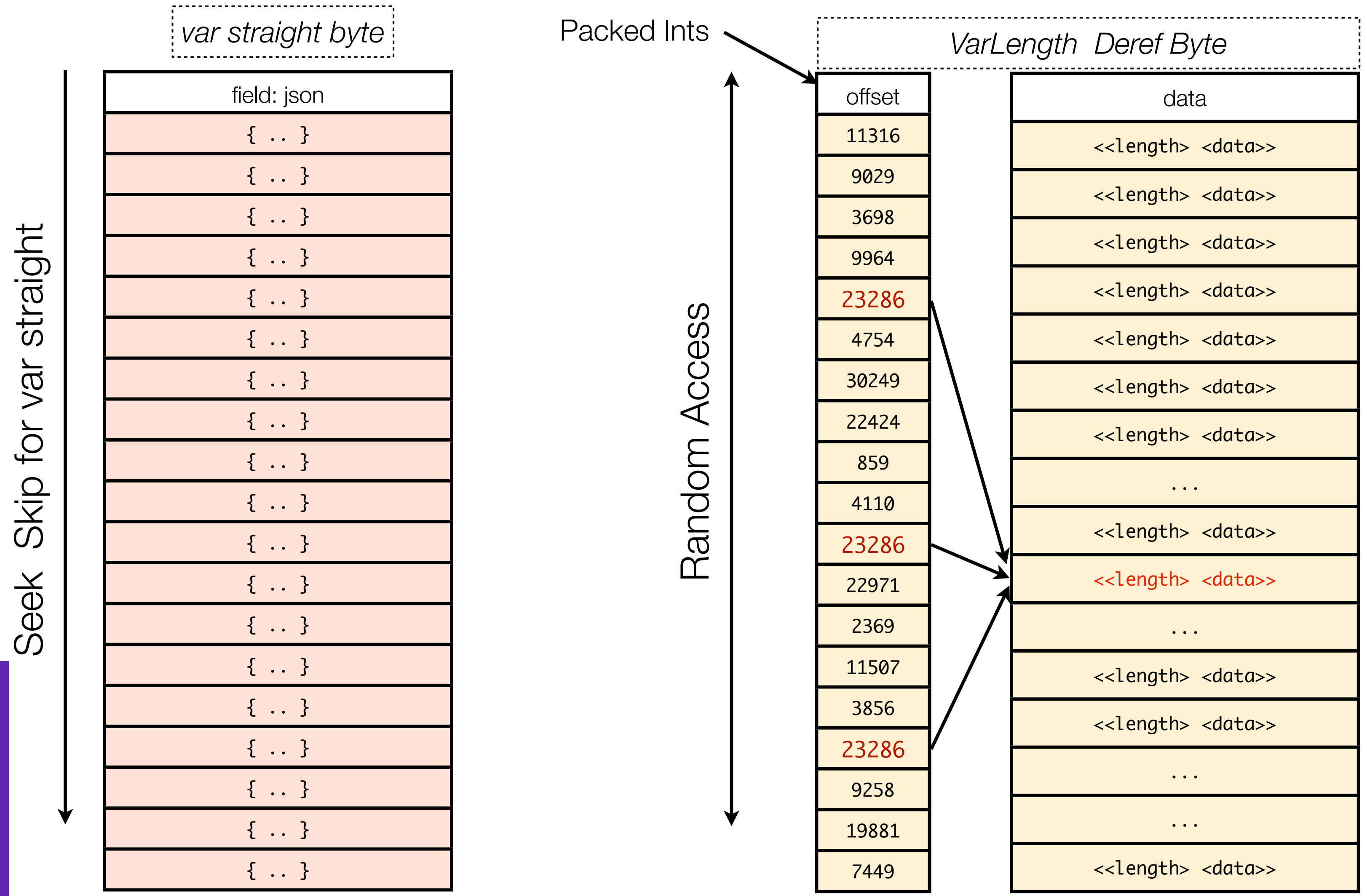
7 - bit per doc

Random Access

Indexed	*Indexed*	
int	*int*	*float32*
field: time	field: id (searchable)	field: page_rank
1288271631431	1	3.2
1288271631531	5	4.5
1288271631631	3	2.3
1288271631732	4	4.44
1288271631832	6	6.7
1288271631932	9	7.8
1288271632032	8	9.9
1288271632132	7	10.1
1288271632233	12	11.0
1288271632333	14	33.1
1288271632433	22	0.2
1288271632533	32	1.4
1288271632637	100	55.6
1288271632737	33	2.2
1288271632838	34	7.5
1288271632938	35	3.2
1288271633038	36	3.4
1288271633138	37	5.6
1288271632333	38	45.0

JTEAM

PerDocument Payloads - Rough Picture



PerDocument Payloads - Features

- Full control over memory consumption / speed for per doc values
- Fully Customizable via Flex API
- Fast loading times (No un-inverting indexed fields like FieldCache)
- Suitable for tight memory environments (mobile phones)
- compact numeric value representation
- Entirely RAM resident if desired (on per field basis)
- Updateable in the future!

PerDocument Payloads - Current State

- Developed in Branch (`lucene/dev/branches/docvalues`)
- In Memory Random Access API (`ValuesSource.get(docid)`)
- On disk Iterator API (`ValuesEnum.advance(docid) / next()`)
- Currently integrated into Flex API
- Tests are stable

What’s new in Lucene 4.0

Questions?

What’s new in Lucene 4.0

The one that always comes:

When will Lucene 4 be released?

JTEAM