APACHE SLING & FRIENDS TECH MEETUP
BERLIN, 25-27 SEPTEMBER 2017

Optimizing OAK repository search

O. Jentsch, Condat AG

Introduction

adaptTo()

Presentation Goals

- Reporting experiences made upgrading CRX2 to OAK
- Best practices for optimizing OAK repository search performance

About me

- Olaf Jentsch
- Senior Software Engineer at Condat AG
- working with Day CQ/AEM since 2006

adaptTo()

Agenda

- Customer‘s Situation
- Migrating repository search to OAK
- Measuring search improvements
- Tools

adaptTo()

Customer‘s situation

Migration project – initial situation

- CQ 5 / AEM 6.0 -> AEM 6.2
  - CRX2 -> OAK
- Large repository (> 1 mio documents ~ 500GB)
- Mix of (customized) XPath, SQL2 queries
- Search features should be the same
- Improving Performance

adaptTo()

Migrating repository search to OAK

Steps to migrate repository search to OAK

1. Achieve your queries to run
2. Ensure your queries to use indexes
3. Optimize index definitions
4. Optimize queries

Query Syntax

- Adapt syntactical changes

  [cond1][cond2] -> [cond1 and cond2]

```
/jcr:root/content/site//element(*)
[sling:resourceType= 'project/components/schreiben' ][
jcr:like(@number,'prefix%')]

/jcr:root/content/site//element(*)
[sling:resourceType= 'project/components/schreiben' and
jcr:like(@number,'prefix%')]
```

adaptTo()

Ensure your queries to use indexes

- Prefer out-of-the-box indexes
  - by slight changes to the query
  - by slight changes to existing out-of-the-box index
- Create your own custom indexes
  - If you search for a custom property
  - If you need aggregation, fulltext, multiple prop.

Slight changes to the query: example

General nodetype change to ...

```
// works in CRX2
select [jcr:path], [jcr:score], * from [nt:base]
   as a where [rep:principalName] is not null
   and isdescendantnode(a, '/home') ...
```

... a more specific nodetype

```
// works in OAK, uses nodetype index
select [jcr:path], [jcr:score], * from [rep:User]
   as a where ...
```

adaptTo()

Slight changes to OOTB index: example

- Fulltext query combined with property condition does not work on OOTB index

```
/jcr:root//element(*, cq:Page)
[jcr:contains(jcr:content/@jcr:title, 'Rezension') and
jcr:content/@hideInNav]
```

Slight changes to OOTB index

- Index definition: /oak:index/cqPageLucene

```
...
"jcrTitle": {
        "jcr:primaryType": "nt:unstructured",
        "nodeScopeIndex": true,
        "useInSuggest": true,
        "propertyIndex": true,
        "analyzed": true,
        "useInSpellcheck": true,
        "name": "jcr:content/jcr:title",
        "type": "String"
}, ...
```

adaptTo()

Custom index definition

- Define index not at root level but at deeper nodes in order to
  - Restrict scope (/content/site)
  - Reduce amount of data
  - Avoid conflicts with system queries and out-of-the-box indexes

adaptTo()

Custom index definition

- Include all properties used in query expression

```
/jcr:root//element(*, cq:Page)
[ not(jcr:like(jcr:content/@propA, '%that%')) and
jcr:like(jcr:content/@propB, '%this%') ]
order by jcr:content/@propC descending
```

Custom index definition

```
"propA": {
        "ordered": false,
        "propertyIndex": true,
        "name": "jcr:content/propA", ...
},
"propC": {
        "ordered": true,
        "propertyIndex": true,
        "name": "jcr:content/propC", ...
},
"propB": {
        "propertyIndex": true, ...
}
```

adaptTo()

Optimize index definitions

- Expand index definition by properties for special search features
  - Fulltext search
  - Facets
  - Excerpts
  - Suggestions

Fulltext search

- oak:index/myindex

```
"compatVersion": 2,
"type": "lucene",
"async": "fulltext-async",
"codec": "Lucene46",
"indexRules": { ...
... "properties": { ...
        "jcr:title": { ...
                "analyzed": true,
                "name": "jcr:title",
                "nodeScopeIndex": true,
                "propertyIndex": false,
                ...
```

Put all properties you want to search through, including those from subnodes of jcr:content

Fulltext search

- Search for:

```
Grippeimpfstoff* -Ausschreibung "2016/2017"
```

- Characteristics
  - AND Operation !
  - Wildcards *?
  - Negation -
  - Phrases

Facets

Put the properties from which you want to extract facets

```
...
"region": {
        "facets": true,
        "analyzed": true,
        "name": "region",
        "propertyIndex": true,
        ...
```

adaptTo()

Facets result

- Search result arranged into category, numerical count of how many matching documents

```
region :
 Berlin / Brandenburg (3)
 Hessen (4)
 keiner Region zugeordnet (6)
 Hamburg (3)
 Bundesweit (802)
 Rheinland-Pfalz (3)
 Baden-Wuerttemberg (5)
```

adaptTo()

Facets search

- Query-Debugger

```
0_property=region
```

- QueryBuilder API

```
Predicate p = new Predicate("region",
        JcrPropertyPredicateEvaluator.PROPERTY);
p.set(JcrPropertyPredicateEvaluator.OPERATION,
        JcrPropertyPredicateEvaluator.OP_EQUALS);
p.set(JcrPropertyPredicateEvaluator.VALUE, null);

query.getPredicates().add(p);
Map<String, Facet> m = query.getResult().getFacets();
```

Excerpts

```
...
"jcr:title": { ...
        "analyzed": true,
        "name": "jcr:title",
        "nodeScopeIndex": true,
        "propertyIndex": false,
        "useInExcerpt": true, ...
},
"excerpt": { ...
        "notNullCheckEnabled": true,
        "propertyIndex": true,
        "name": "rep:excerpt", ...
}
...
```

Set special property rep:excerpt if you want to show excerpt's from query results

adaptTo()

Suggestions

```
"compatVersion": 2,
"type": "lucene",
"async": "async",
"suggestion": {
        "suggestAnalyzed": true,
        "suggestUpdateFrequencyMinutes": 480
},
"indexRules": { ...
... "properties": { ...
        "jcr:title": { ...
                "useInSuggest": true,
                "analyzed": true,
                "name": "jcr:title",
                "propertyIndex": true, ...
        }, ...
```

adaptTo()

Optimize queries

- Switch from XPATH to SQL-2
  - Avoid translation costs
  - Avoid too many OR-Conditions that lead to many union select statements
- Modify filter condition
  - Reduce the number of conditions
  - Find simpler conditions

XPATH to SQL-2 Translation: example

- XPATH

```
/jcr:root/content/siteOne//element(*, cq:Page)

[(jcr:like(jcr:content/@number,'prefixA%') or
jcr:like(jcr:content/@number,'prefixB%') or
jcr:like(jcr:content/@number,'prefixC%'))

and

(jcr:like(jcr:content/@thema,'%topicA%') or
jcr:like(jcr:content/@thema,'%topicB%') or
jcr:like(jcr:content/@thema,'%topicC%'))]
```

XPATH to SQL-2 Translation: example

- SQL-2 generated

```
select [jcr:path], [jcr:score], * from [cq:Page] as a where
    isdescendantnode(a, '/content/siteOne')
    and [jcr:content/number] like 'prefixA%'
    and [jcr:content/thema] like '%topicA%'
union select [jcr:path], [jcr:score], * from [cq:Page] as a where
    isdescendantnode(a, '/content/siteOne')
    and [jcr:content/number] like 'prefixA%'
    and [jcr:content/thema] like '%topicB%'
union select [jcr:path], [jcr:score], * from [cq:Page] as a where
    isdescendantnode(a, '/content/siteOne')
    and [jcr:content/number] like 'prefixB%'
    and [jcr:content/thema] like '%topicA%'
union select [jcr:path], [jcr:score], * from [cq:Page] as a where
    ... etc.
```

XPATH to SQL-2 Translation: example

- SQL-2 optimised („manually“)

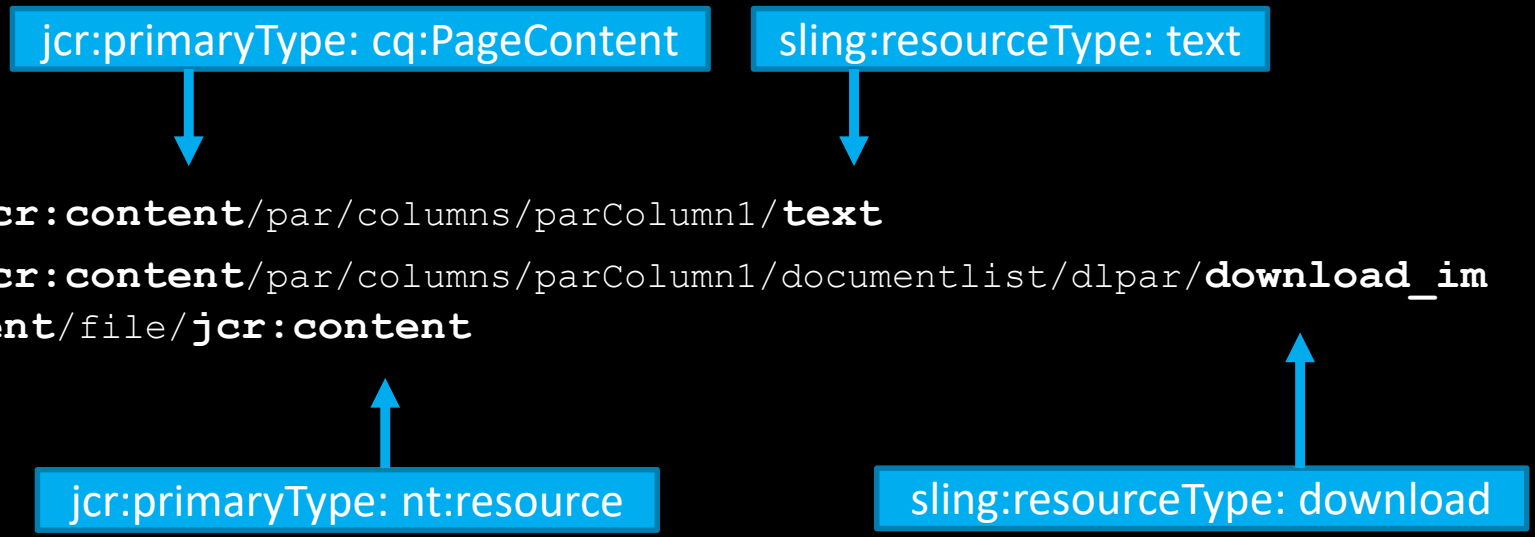
```
select [jcr:path], [jcr:score], * from [cq:Page] as a where
    isdescendantnode(a, '/content/siteOne’)

    and ([jcr:content/number] like 'prefixA%’
            or [jcr:content/number] like 'prefixB%’
            or [jcr:content/number] like 'prefixC%’)
    and ([jcr:content/thema] like '%topicA%’
            or [jcr:content/thema] like '%topicB%’
            or [jcr:content/thema] like '%topicC%')
```

adaptTo()

Modify filter condition: requirements

- Deep content structure inside a cq:Page
- Hierarchical mix of text, list and download
- Required results of type cq:PageContent and download
- Results ordered by jcr:score

Modify filter condition: content



Modify filter condition: query

```
/jcr:root/content/siteOne//* [(jcr:contains(., 'query'))
and

(@jcr:primaryType = 'cq:PageContent'

or

@sling:resourceType = 'foundation/components/download')]
```

adaptTo()

Modify filter condition: simplify condition

- Simple solution: add a further property to both nodes, that
  - Ideally already exists in one of the them
  - Ideally makes sense to show in result page
  - Is mandatory or automatically set by node creation
  - Is included in index definition

Modify filter condition: modify content

```
page/jcr:content/par/columns/parColumn1/text/@text = "query"
page/jcr:content/@region = "Berlin"
```

```
page/jcr:content/par/columns/parColumn1/documentlist/dlpar/download_im
Dokument/@jcr:title = "query"

page/jcr:content/par/columns/parColumn1/documentlist/dlpar/download_im
Dokument/@region = "Berlin"
```

Modify filter condition: simplify condition

```
/jcr:root/content/siteOne//* [(jcr:contains(., 'query'))
and

        @region
```

- and re-gain performance
  and sort order by jcr:score as well!

adaptTo()

Measuring search improvements

- By the old way: log-message
  - Compare log-message timestamps
- Take time output from Query-Debug-Log
  - Query time
  - Facet extraction time
- Test with Live Content !
- Bookkeeping in (excel) sheet

Tools

- Query-Debug-Log
- Diagnosis-Tool
- Query-Debugger
- Last but not least CRXDE

adaptTo()

Further information

- https://docs.adobe.com/docs/en/aem/6-3/deploy/platform/queries-and-indexing.html
- https://docs.adobe.com/docs/en/aem/6-3/administer/operations/operations-dashboard.html
- https://docs.adobe.com/docs/en/aem/6-3/develop/search/querybuilder-api.html
- https://hashimkhan.in/2015/12/02/query-builder/
- https://jackrabbit.apache.org/oak/docs/query/lucene.html
- http://www.aemstuff.com/blogs/feb/aemindexcheatsheat.html
- http://oakutils.appspot.com/

adaptTo()

Questions ???