

**From:** Marko Zaric  
**To:** [DL-dev](#)  
**Subject:** [OAK][TraversingIndex] Incorrect cost calculation?  
**Date:** Monday, June 16, 2014 6:34:00 PM

---

Hello,

I believe we've run into what appears to be a bug in the cost calculation method of the TraversingIndex in OAK, but wanted to double-check first before opening a JIRA issue.

When looking into some performance issues on the AEM Communities for Learning Beta cluster, we've noticed that some queries take an unusual amount of time. One of the queries in question, when translated into XPath query looks like this:

```
/jcr:root/content/learncube/data/testing2/resources//*[ @catalog-visibility = 'true' and  
@sling:resourceType = 'learncube/components/resource'] order by @date-created descending
```

Basically, it looks for nodes of `sling:resourceType='learncube/components/resource'` under the given (tenant-specific) path - nodes that we are looking for are not direct children of the given path, but one level deeper. `sling:resourceType` property is indexed, and so the expectation here is that the system will:

1. lookup the `sling:resourceType` index and return all nodes of type `'learncube/components/resource'` (this should be pretty fast)
2. go through all the returned nodes and filter out those that don't satisfy the path restriction (and then order them by date-created)

What we are seeing though is that when the number of `'learncube/components/resource'` type of nodes in the system is relatively small (i.e. < 100), then it works as expected, but if the number of these nodes is larger than 100, then the process is significantly different.

Instead of looking up the `sling:resourceType` index, the system basically traverses all child nodes of the given path and looks for the nodes of type `'learncube/components/resource'`. This takes a significant amount of time (especially if done for the first time when nodes are not cached) and basically results in the following warnings in the logs :

```
09.06.2014 22:26:22.069 *WARN* [10.27.7.165 [1402377981671] GET  
/content/learncube/data/testing2/resources.resourcelist.json HTTP/1.0]  
org.apache.jackrabbit.oak.spi.query.Cursors$TraversingCursor Traversed 1000 nodes with filter  
Filter(query=select [jcr:path], [jcr:score], * from [nt:base] as a where [sling:resourceType] =  
'learncube/components/resource' and isdescendantnode(a,  
'/content/learncube/data/testing2/resources') order by [date-created] desc /* xpath:  
/jcr:root/content/learncube/data/testing2/resources//*[ @sling:resourceType =  
'learncube/components/resource'] order by @date-created descending */,  
path=/content/learncube/data/testing2/resources/*, property=  
[sling:resourceType=learncube/components/resource]); consider creating an index or changing the
```

## query

...

```
09.06.2014 22:26:25.832 *WARN* [10.27.7.165 [1402377981671] GET
/content/learncube/data/testing2/resources.resourcelist.json HTTP/1.0]
org.apache.jackrabbit.oak.spi.query.Cursors$TraversingCursor Traversed 4000 nodes with filter
Filter(query=select [jcr:path], [jcr:score], * from [nt:base] as a where [sling:resourceType] =
'learncube/components/resource' and isdescendantnode(a,
'/content/learncube/data/testing2/resources') order by [date-created] desc /* xpath:
/jcr:root/content/learncube/data/testing2/resources//*[ @sling:resourceType =
'learncube/components/resource'] order by @date-created descending */,
path=/content/learncube/data/testing2/resources/*, property=
[sling:resourceType=learncube/components/resource]); consider creating an index or changing the
query
```

At first, it wasn't clear why the system decides to traverse the nodes instead of using the index, so I enabled DEBUG logs and noticed the following:

```
09.06.2014 22:26:21.711 *DEBUG* [10.27.7.165 [1402377981671] GET
/content/learncube/data/testing2/resources.resourcelist.json HTTP/1.0]
org.apache.jackrabbit.oak.query.QueryImpl cost for property is 196.0
```

...

```
09.06.2014 22:26:21.712 *DEBUG* [10.27.7.165 [1402377981671] GET
/content/learncube/data/testing2/resources.resourcelist.json HTTP/1.0]
org.apache.jackrabbit.oak.query.QueryImpl cost for traverse is 100.0
```

The number displayed as the 'cost for property' corresponds to the overall number of overall 'learncube/components/resource' nodes, but what was not clear is why the cost of traversal would be only 100. After all, we clearly see that the actual cost of traversing all the child nodes is more than 4000, so why does the system think that the cost of doing this kind of traversal would be only 100?

I guess that the source code of the method that calculates the cost of traversal can be found here:

<https://github.com/apache/jackrabbit-oak/blob/trunk/oak-core/src/main/java/org/apache/jackrabbit/oak/query/index/TraversingIndex.java>

If I understand it correctly, what it's trying to do is to estimate the cost of doing the traversal of all child nodes for a node with the given path – it doesn't know the actual number of child nodes before it tries fetching them, and so it can only try to estimate the cost. Now, for the case when the path restriction is DIRECT\_CHILDREN, it estimates the cost to 1,000,000 (i.e. 1 million child nodes, which I guess is the worst case scenario).

However, when it tries to estimate the traversal of ALL\_CHILDREN (i.e. not only direct child nodes) then the logic becomes unclear:

```
double nodeCount = 100000000;
String path = filter.getPath();
```

```

PathRestriction restriction = filter.getPathRestriction();
switch (restriction) {
    ...
case ALL_CHILDREN:
    if (!PathUtils.denotesRoot(path)) {
        for (int depth = PathUtils.getDepth(path); depth > 0; depth--) {
            // estimate 10 child nodes per node
            nodeCount /= 10;
        }
        break;
    ...
case DIRECT_CHILDREN:
    // we expect 1 million child nodes in the worst case
    nodeCount = 1000000;
    break;
}

```

As a consequence of this though, instead of getting a larger number for the cost, it basically yields 100 in our scenario – the larger the depth of the initial provided path, the smaller the number (the path in our case is `/content/learncube/data/testing2/resources`, which has a depth of 5, and so the result is 100).

This logic seems flawed - I was thinking of opening a JIRA issue for this, but wanted to double-check just in case we are missing something here.

Thanks,  
Marko.