Infrastructure - Task #7491

Feature # 6498 (Closed): V2 Metacat MN and CN Support

The performance issue on MN.getLogRecord

2015-11-17 18:31 - Jing Tao

Status:	Closed	**Start date:**	2015-11-17
Priority:	Normal	**Due date:**	
Assignee:	Ben Leinfelder	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	CCI-2.0.0		
Milestone:	None	**Story Points:**	
Product Version:	*		

Description

This join on concatenated values is probably costly:
al.docid = id.docid||'.'||id.rev
I vaguely remember trying to find a way to index the concatenated value since I doubt a compound index that contains both docid and rev will actually help sine the period isn't included. The access_log table probably should have had two columns (like every other table in metacat) which would have made index-based joins possible.
Maybe you will have better luck googling indexes based on concatenation values, however
-ben

> On Nov 16, 2015, at 10:04 PM, Matt Jones jones@nceas.ucsb.edu wrote:
>
> Thanks for the overview, Jing.
>
> I think you are right that we probably don't need the join of the identifier to just get the count in the first query.
>
> But, I think we need the join to provide the GUID rather than the docid in the second query. Joins can be expensive. You can find out the cost model for the different parts of the query using the psql EXPLAIN command. That should tell you whether an index is being used versus a full table scan. For an index to be used, it must precisely match your query conditions, so may need to be a multi-column index. In addition, one reason count(*) can be slow in postgres is if there are a lot of dead rows in the database. Can you confirm that we have autovacuum set for the database? If not and if it isn't vacuumed regularly, then a lot of cruft can accumulate.
>
> Matt
>
> On Mon, Nov 16, 2015 at 6:36 PM, Jing tao@nceas.ucsb.edu wrote:
> Hi Ben, Chris and Matt:
>
> Today Robert reported a timeout issue when the sandbox cn tried to harvest the log records from mn-sandbox-ucsb-1. The java client uses the command like https://mn-sandbox-ucsb-1.test.dataone.org:443/knb/d1/mn/v1/log?start=0&count=10000 to harvest the log records. The number of the count keeps the constant of 10000, but the number of the start keeps increasing. The harvest worked at the beginning point, then stopped working (got the timeout exception from the mn) after a while. The timeout setting is about 300 seconds, I believe.
>
> I used the curl command - curl --cert /etc/dataone/client/private/urn_node_cnSandboxUCSB1.pem " https://mn-sandbox-ucsb-1.test.dataone.org:443/knb/d1/mn/v1/log?start=0&count=1000" and it took about 220 seconds to get the result.
>
> I dug the the EventLog.getD1Report method and found there two queries to run there:
>
> 1. Counting query - select count(*) from access_log al, identifier id where al.docid = id.docid||'.'||id.rev and date_logged >= '1969-12-31 16:00:00.001-08' and date_logged <'2015-11-16 14:43:01.807-08';
>
> I ran this command on the psql of the mn-sandbox-ucsb-1. It returns more than 8,700,000 records and took about 215 seconds! It is the most expensive part in the whole process (whole process took about 220 seconds). I don't understand why we need to join the access_log table (more than 8,700,000 records) with identifier table (more than 160,000 records) in this query. I looked that method and found the only reason we need join the two tables is we need return the identifiers back, but the access_log only has the docids. But for the counting, we don't care. If we don't join the table, the query - select count(*) from access_log where date_logged >= '1969-12-31 16:00:00.001-08' and date_logged <'2015-11-16 14:43:01.807-08'; - returns the same result

and only took 3 seconds. I looked the code and found even though the most complicated where clause doesn't include any columns in the identifier table. Did I miss anything here?

1. Selection query - select entryid, id.guid as identifier, ip_address, user_agent, principal, case when event = 'insert' then 'create' else event end as event, date_logged from access_log al, identifier id where al.docid = id.docid||'.'||id.rev and date_logged >= '1969-12-31 16:00:00.001-08' and date_logged < '2015-11-16 14:43:01.807-08' order by entryid LIMIT 10000 OFFSET 0;

The query only took 0.17 second! Oh, if we didn't join the two tables in the first counting query, the whole query process will take less than 4 second and we are all set! But Robert said it worked at the beginning, but then stopped working. So I thought maybe when the OFFSET is huge, the query will be every expensive. So I increased the OFFSET to 8,000,000 and the query took 253 seconds. I think we need to do some work on this query as well.

I added a new index on the column date_logged, but it didn't helpful. I bet the issue is still the joint of two big tables. The reason that we have to join them is the access_log table doesn't have the identifiers, only has the docids. If we don't join the tables and only get docids back, the query only took 6 seconds even though the OFFSET is 8,000,000. So I am thinking if we can add the identifier as the new column in the access_log table and we can avoid the joint. If we do this, we need to log the identifier as well in all around the metacat code. We also need a upgrade script to fill the new column for the existing records (it maybe will take a long time).

Any thought and comment?

Thank you very much!

Jing

History

#1 - 2015-11-17 18:33 - Jing Tao

Will we consider to add a new column - guid on the access_log table if the indexing doesn't work well?

#2 - 2015-11-17 19:26 - Ben Leinfelder

- Status changed from New to In Progress

- % Done changed from 0 to 30

Summarizing Jings email:
For both the count(*) query and the large OFFSET query, we end up performing millions of joins (all records are processed).
Without the join and even with a high OFFSET value, we get much better performance.

#3 - 2015-11-17 23:40 - Ben Leinfelder

- Status changed from In Progress to Closed

- % Done changed from 30 to 100

- translation missing: en.field_remaining_hours set to 0.0

I've updated the implementation to use a subquery that does not join to the identifier table to do the paging. The join is then only performed on the page range being returned.
This is deployed on mn-sandbox-ucsb-1.test.dataone.org. I ran the following queries and they all complete in under 10 seconds:

sudo curl --cert /etc/dataone/client/private/urn_node_cnSandboxUCSB1.pem "
https://mn-sandbox-ucsb-1.test.dataone.org:443/knb/d1/mn/v1/log?start=0&count=0"
sudo curl --cert /etc/dataone/client/private/urn_node_cnSandboxUCSB1.pem "
https://mn-sandbox-ucsb-1.test.dataone.org:443/knb/d1/mn/v1/log?start=0&count=1000"
sudo curl --cert /etc/dataone/client/private/urn_node_cnSandboxUCSB1.pem "
https://mn-sandbox-ucsb-1.test.dataone.org:443/knb/d1/mn/v1/log?start=8000000&count=1000"