Metacat - Bug #1879

Metacat Performance: Summary

01/18/2005 01:42 PM - Saurabh Garg

Status:	New	**Start date:**	01/18/2005
Priority:	Immediate	**Due date:**	
Assignee:	Jing Tao	**% Done:**	0%
Category:	metacat	**Estimated time:**	0.00 hour
Target version:	Unspecified	**Spent time:**	0.00 hour
Bugzilla-Id:	1879		

Description

These are notes based on the changes I did in Metacat source for improving the performance. I was not able to make the below given changes due to lack of time and because these changes would require a more thorough testing.

1. xml_index is a large table and most of the time we are searching for paths which are needed by the web interface and Morpho for displaying the results. So it might be a good idea to create a seperate table similar to xml_index table which has only got some predefined paths in it. For current knb skin and morpho this table on would have about 1/200th the number of records that xml_index has right now. The code that would need to be modified would include both insertion and deletion of documents.

2. For searching data in particular given paths (e.g. geographic query) the current query uses both xml_index and xml_nodes. This can be improved by just using xml_index table which has nodedata in it. But there is a lot of repetition of data in xml_index table. So it has to be tested and checked if this would result in better performance or otherwise. This would require rewriting QueryTerm.java.

Related issues:

Blocks Metacat - Bug #2157: Metacat Performance: Optimize Postgres and Tomcat...	**Resolved**	**07/13/2005**
Blocks Metacat - Bug #2155: Metacat Performace: Rewrite the xml_nodes queries	**In Progress**	**07/13/2005**
Blocks Metacat - Bug #2153: Metacat Performace: Add/drop indices on key columns	**Resolved**	**07/13/2005**
Blocks Metacat - Bug #2557: Metacat Performace: Rewrite the xml_access part o...	**Resolved**	**09/28/2006**
Blocks Metacat - Bug #2175: Metacat Performace: Upgrading hardware setup	**Resolved**	**09/05/2005**

History

#1 - 04/08/2005 08:43 AM - Saurabh Garg

Set of changes made in regards to performance for release 1.5:

- Performance impovement done in searching of datasets. A part of the search time was taken up because of generation of resultset after the search had be done. So this was the sequence of events in earlier metacat
  1) Search for the given text
  2) A list of docids generated
  3) For each of the docids
  3a) Use xml_index and xml_nodes table to generate the string descibing the document including the returnfields requested in the search
  4) Add all the strings from step 3a to send back the resultant document.
  Here a decent amount
  of time was being taken by step 3a. The algorithm is
  now modified by addition of two tables xml_queryresult and xml_returnfields and a user defined parameter xml_returnfield_count. The new algorithm works as follows
  1) Search for the given text
  2) A list of docids generated
  3) For the given set of return fields generate a unique string and check if that string exists in xml_returnfields
  3a) If string does not exist in xml_returnfield, then enter a new record usage_count as 1 and returnfield_string as the unique string generated above.

3b) Else if the string exists, increment usage_count
4) Given the docids from step 2 and the id of the returnfield record from step 3, query for any docids that already exist in xml_queryresult. For the docids that do exist, get the queryresult_string.
5) For rest of the docids from step2 which were not found in step 4, do the following for each of the documents:
5a) Use xml_index and xml_nodes table to generate the string descibing the document including the returnfields requested in the search
5b) If usage_count of the record from step is more than xml_returnfield_count set in metacat.properties, then store the string in xml_queryresult as a record which has the returnfield_id representing the set of returnfields, docid representing the document and the string as queryresult_string.
6) Add all the strings from step 4 and step 5a to send back the resultant document
So the results from step 3a in previous algorithm are effectively cached and hence same computation is not done again and again for each search. When a document is deleted, all the entries for that document in xml_queryresult table are also deleted. When a document is updated, all the entries for that document in xml_queryresult table are deleted. This works fine because those entries will be generated and cached again the next time the document is part of a search is requested.

- Performance impovement done for % search. Now % search doesnt include searching the xml_nodes table.

#2 - 09/28/2006 08:40 AM - Matt Jones

The growth of Metacat's document store has eroded the performance gains we made previously. We are now back to having simple queries take ~25 seconds or more. Of this, it appears that the SQL query is taking about 12 secs, and the XSLT transform and download to client is taking about 13 secs. Additional performance improvements can be made. Here are some ideas:

1) Rewrite the SQL to eliminate unneccesary subqueries. This seems to have the potential of reducing SQL query time from 12 secs to about 1 sec. See bug #2155 for additional details.

2) Add indices on the major columns that are queried and rewrite the queries to avoid full table scans. This should make a major difference, but it is hard to acheive because we make use of the 'LIKE' operator in order to perform substring searches using wildcards. It appears that we need to use the '=' operator in order for the indices to be used, and this eliminates some of our search capability. So far, i've determined we need the following indices for the search:

```
xml_path_index on upper(nodedata)
xml_access     on lower(principal_name)
xml_access     on perm_type
xml_access     on permission
xml_access     on perm_order
xml_access     on subtreeid
xml_documents  on lower(user_owner)
```

3) Use paged query returns to cut down on the document size returned. This helps in 2 ways. First, it reduces the size of the XSLT transform, which appears to be slow on larger documents. Second, it reduces the size of the HTMl document to be transferred to the client. See bug #129.

#3 - 11/09/2007 04:02 PM - Jing Tao

Move to 1.7.1 release

#4 - 03/27/2013 02:18 PM - Redmine Admin

Original Bugzilla ID was 1879