Obs Table Efficiency Review

Unknown macro: {float}

Task	Assignee	Completion	Expected Date
Create test environment	Jeremy		10 April 2012
Analyze slow logs	Jeremy		10 April 2012
Evaluate analysis results			
Build testing script			
Determine suggested changes			
Test with each suggestion			
Analyze results for each suggestion			
Test with final suggestions			
Analyze final results			
Write final presentation			23 April 2012
Write report			27 April 2012

Contributors

- Jeremy Keiper
- user-91d08

Project Description

Abstract

Purpose

One bottleneck for operations at AMPATH, a large medical facility in Kenya, is the performance of the Obs table in an implementation of OpenMRS running on MySQL. The table is constantly updated and queried, and contains thousands of observations per patient totaling over 123 million records. When the table was just under 100 million records, disaster recovery took upwards of 48 hours to completely rebuild this one table, not including the rest of the database. Improvements to the Obs table and its associated indexes will greatly increase the speed at which care can be provided, ultimately contributing to saving lives not only in Kenya but around the world.

Approach

1. Examine database logs to find most common operations on the Obs table.
2. Classify most common operations into insert, update, equality search or range searches.
3. Benchmark performance of most common operations on a large sample set of data.
4. Determine new indexes or changes to existing table structure or indexes that should improve speed of most common operations:
    - Tabulate accessed columns, joined tables, search types and indexes used for each query.
    - Rank existing indexes by amount of times accessed during most common operations.
    - Tally multiple-column searches, looking for possible aggregate indexing strategies.
    - Hypothesize improvements via adjustments to existing indexes or adding new ones.
5. Compile list of improvements using a reasonable threshold on results of evaluations.
6. For each identified improvement:
    - Write code changes for improvement into a custom OpenMRS build.
    - Implement changes on sample dataset.

    - Benchmark performance after improvement to identify benefits.
7. Write a report on improvements and recommendations.
8. Submit proposed code and table changes to OpenMRS for review.

Deliverables

- OpenMRS core software patch with recommended changes implemented
- SQL for identified improvements
- Report containing benchmark results and recommendations for OpenMRS

Progress Notes

This section should contain all notes relevant to the work we are doing. Remember to include the date of your observations. Latest observations come first in this section.

2012-04-17 – Jeremy – Analysis tweaking

In order to remove samples from the analysis, I removed lines matching this pattern from the file:

```
.*\\G\n
```

2012-04-17 – Jeremy – Slow log tweaking

I realized that aggregating results from several months would be tedious, especially since mk-query-digest is meant to do this automatically. I also realized that some very large files processed quickly while other smaller files would never finish. I used awk to split these files several times into different date ranges and discovered three dates that had something in the file causing the failures:

- 2011-09-22
- 2011-09-26
- 2011-09-29

With this in mind, I wrote the following awk script to split the original mysql-slow.log into a process-able file:

```
awk '{
        if(/# Time: 110524/) {
                p = 1;
        } else if(/# Time: 110922/ || /# Time: 110926/ || /# Time: 110929/) {
                p = 0;
        } else if(/# Time: 110923/ || /# Time: 110927/ || /# Time: 110930/) {
                p = 1;
        }
        if(p == 1) {
                print
        }
}' mysql-slow.log > mysql-slow-trimmed.log
```

This resulted in a 1.5GB file, officially (1,550,849,949 / 1,699,502,240) * 100 = 91.25% of the original.

2012-04-10 – Jeremy – Query analysis

We're using Maatkit to fingerprint, analyze and report on the queries in the slow log. The following command line filters out only those queries that hit the obs table in some way, generates a query review table (for us to easily get the fingerprints from) and stores the actual review in its own log file:

```
mk-query-digest --filter '$event->{fingerprint} =~ m/ obs /' --fingerprints --review h=localhost,D=obsreview,
t=queries,u=root,p=***** --create-review-table mysql-slow.2011-05.log > analysis-2011-05.log
```

Running this on large data files (largest is 285MB) apparently requires several hours on a dual core laptop with 4GB of RAM. I will let it continue running, but realize that the statistics we are getting will have to be aggregated. I have access to a server with more resources, so perhaps I can let it run on the total slow log (1.6GB) and see if we can get faster results.

2012-04-03 – Jeremy – Test environment creation

A benefit of using the OpenMRS platform for testing is that it can be run in a java-based standalone environment, with an on-demand MySQL instance. This standalone version of OpenMRS keeps all data within a single folder, and can easily be replicated. We can use it to set up a test environment with the OpenMRS-provided large testing data set (over 400,000 observations). This test environment can be unzipped into a test folder, turned on, modified by our proposed changes, and then tested against to find our results. It should provide a very consistent platform for detecting the influence of our structural changes.

Unfortunately, Mario was unable to get this "bubble" running right away, and it resulted in a blank database rather than the one with the large dataset, and Mario could not fully import the dataset I posted on this page. I will work on getting one of these ready and reliable in time for testing.

2012-04-01 – Jeremy – Query retrieval

The AMPATH Production and Research machines are both configured to write queries to the slow log, but only the Production machine is doing so. Fortunately we have a lot of data to work with: from late May 2011 until late March 2012. The down side is that the file size of this log (uncompressed) is 1.6GB.

Discussion

obs table description:

Field	Type	Null	Key	Default	Extra
obs_id	int(11)	NO	PRI	NULL	auto_increment
person_id	int(11)	NO	MUL	NULL	
concept_id	int(11)	NO	MUL	0	
encounter_id	int(11)	YES	MUL	NULL	
order_id	int(11)	YES	MUL	NULL	
obs_datetime	datetime	NO	MUL	0000-00-00 00:00:00	
location_id	int(11)	YES	MUL	NULL	
obs_group_id	int(11)	YES	MUL	NULL	
accession_number	varchar(255)	YES		NULL	
value_group_id	int(11)	YES		NULL	
value_boolean	tinyint(1)	YES		NULL	
value_coded	int(11)	YES	MUL	NULL	
value_coded_name_id	int(11)	YES	MUL	NULL	
value_drug	int(11)	YES	MUL	NULL	
value_datetime	datetime	YES		NULL	
value_numeric	double	YES		NULL	
value_modifier	varchar(2)	YES		NULL	
value_text	text	YES		NULL	
comments	varchar(255)	YES		NULL	
creator	int(11)	NO	MUL	0	
date_created	datetime	NO		0000-00-00 00:00:00	
voided	tinyint(1)	NO		0	
voided_by	int(11)	YES	MUL	NULL	
date_voided	datetime	YES		NULL	
void_reason	varchar(255)	YES		NULL	
value_complex	varchar(255)	YES		NULL	
uuid	char(38)	NO	UNI	NULL	
previous_version	int(11)	YES	MUL	NULL	

Indexes on obs table in current data model

Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
0	PRIMARY	1	obs_id	A	477066	NULL	NULL		BTREE
0	obs_uuid_index	1	uuid	A	477066	NULL	NULL		BTREE
1	answer_concept	1	value_coded	A	12	NULL	NULL	YES	BTREE
1	encounter_observations	1	encounter_id	A	26503	NULL	NULL	YES	BTREE
1	obs_concept	1	concept_id	A	354	NULL	NULL		BTREE
1	obs_enterer	1	creator	A	10	NULL	NULL		BTREE
1	obs_location	1	location_id	A	12	NULL	NULL	YES	BTREE
1	obs_order	1	order_id	A	8	NULL	NULL	YES	BTREE
1	patient_obs	1	person_id	A	14908	NULL	NULL		BTREE
1	user_who_voided_obs	1	voided_by	A	12	NULL	NULL	YES	BTREE

1	answer_concept_drug	1	value_drug	A	8	NULL	NULL	YES	BTREE
1	obs_grouping_id	1	obs_group_id	A	8	NULL	NULL	YES	BTREE
1	obs_name_of_coded_value	1	value_coded_name_id	A	8	NULL	NULL	YES	BTREE
1	obs_datetime_idx	1	obs_datetime	A	158	NULL	NULL		BTREE
1	previous_version	1	previous_version	A	8	NULL	NULL	YES	BTREE

Resources

- Maatkit - utility pack with scripts for analyzing log files and generating traffic, among others.

Attachments

File	Modified
ZIP Archive demo-trunk.sql.gz	2012-03-27 by Jeremy Keiper
Text File analysis-baseline.log	2012-04-22 by Jeremy Keiper
Text File analysis-finalRecommendations.log	2012-04-22 by Jeremy Keiper
Text File analysis-obsDateCreated.log	2012-04-22 by Jeremy Keiper
Text File analysis-obsLocationPerson.log	2012-04-22 by Jeremy Keiper
Text File analysis-obsVoided.log	2012-04-22 by Jeremy Keiper
Text File analysis-obsVoidedConcept.log	2012-04-22 by Jeremy Keiper
Text File analysis-obsVoidedConceptDatetimePerson.log	2012-04-22 by Jeremy Keiper
Text File analysis-obsVoidedLocation.log	2012-04-22 by Jeremy Keiper
Text File analysis-obsVoidedLocationConcept.log	2012-04-22 by Jeremy Keiper
Text File analysis-obsVoidedPerson.log	2012-04-22 by Jeremy Keiper

Download All