Foreman - Bug #4114

Trends don't scale well and become very slow

01/17/2014 11:05 AM - Ewoud Kohl van Wijngaarden

Status:	Duplicate		
Priority:	Normal		
Assignee:			
Category:	Trends		
Target version:			
Difficulty:		**Fixed in Releases:**	
Triaged:		**Found in Releases:**	
Bugzilla link:		**Red Hat JIRA:**	
Pull request:			

Description

Loading trends for kernel versions (45 different versions, 152 hosts) for the past 3 days takes 97,45 seconds.

Unrelated to my environment here are some debug logs: https://gist.github.com/xorpaul/8456338.

Related issues:

Related to Foreman - Bug #5568: Possible memory leak in trends	**Closed**	**05/05/2014**
Is duplicate of Foreman - Bug #7505: Trends are not aggregrated	**Closed**	**09/17/2014**

History

#1 - 01/18/2014 12:06 AM - Robert Birnie

A lot of this is based on the number of data points as there is a loop that loops over each one to format it properly. If performance is slowing down best option is to change how often you are collecting trends data from the cronjob. Default is 30minutes, change it to hourly or every two hours would halve the time it takes to load.

If you have sampling every half hour, with 45 versions, its looping over an array of 6480 items: 3 * 24 * 2 * 45

#2 - 05/06/2014 07:50 AM - Dominic Cleal

- Related to Bug #5568: Possible memory leak in trends added

#3 - 08/18/2014 11:44 PM - Glen Ogilvie

We had this break badly for us. We monitored the trend uptime.
Here are some details of what we saw.

```
select count(*) from trend_counters;
count
----------
24613273
```

The database was 3GB in size.

Trying to view the trends resulted in 100% CPU load, with ruby consuming lots of memory and throwing poor performing queries at the database.

Saw lots of:

```
2014-08-14 23:06:31.209 NZST foreman 2014-08-14 23:00:10 NZST foreman LOG:  duration: 3640.003 ms  statement: SELECT  1 AS one FROM "trend_counters"  WHERE ("trend_counters"."created_at" = '2014-08-14 11:00:14.286739' AND "trend_counters"."trend_id" = 1145) LIMIT 1
2014-08-14 23:06:33.401 NZST  2014-08-14 19:33:36 NZST  LOG:  checkpoints are occurring too frequently (7 seconds apart)
2014-08-14 23:06:33.401 NZST  2014-08-14 19:33:36 NZST  HINT:  Consider increasing the configuration parameter "checkpoint_segments".
2014-08-14 23:06:33.890 NZST foreman 2014-08-14 23:00:10 NZST foreman LOG:  duration: 2283.587 ms  statement: SELECT  1 AS one FROM "trend_counters"  WHERE ("trend_counters"."created_at" = '2014-08-14 11:00:14.286739' AND "trend_counters"."trend_id" = 1146) LIMIT 1
```

Explaining some of the queries, they didn't seem to be using the indexes efficiently.

```
QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Limit  (cost=174.34..26198.90 rows=1 width=0)
   ->  Bitmap Heap Scan on trend_counters  (cost=174.34..26198.90 rows=1 width=0)
         Recheck Cond: (trend_id = 1233)
         Filter: (created_at = '2014-08-14 11:00:14.286739'::timestamp without time zone)
         ->  Bitmap Index Scan on index_trend_counters_on_trend_id  (cost=0.00..174.34 rows=7820 width=0)
               Index Cond: (trend_id = 1233)
(6 rows)
```

And when we try to remove the trend, we got:

DELETE FROM "trend_counters" WHERE "trend_counters"."trend_id" IN (8, 9, 10, 11, 1 ....)

And explaining it,

Delete on trend_counters (cost=5945.25..207753.75 rows=271905 width=6)
-> Bitmap Heap Scan on trend_counters (cost=5945.25..207753.75 rows=271905 width=6)
Recheck Cond: (trend_id = ANY ('{8,9,10,11,19,20,21,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41}'::integer[]))
-> Bitmap Index Scan on index_trend_counters_on_trend_id (cost=0.00..5877.28 rows=271905 width=0)
Index Cond: (trend_id = ANY ('{8,9,10,11,19,20,21,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41}'::integer[]))

We also saw:
SELECT 1 AS one FROM "trend_counters" WHERE ("trend_counters"."created_at" = '201
4-08-14 11:30:18.548358' AND "trend_counters"."trend_id" = 2260) LIMIT 1

It appears the code is not using the database efficiently, and is looping over records and using the "IN" syntax.

I suggest that we re-think the way trends work.   The type of data lends itself to being stored in a graphing database, such as Graphite or RRD.   My recommendation would be, to change trends so that instead of writing them to Postgresql, they are sent to a statsd target.. which can do whatever it wants, including sending them to graphite. The trends link would then include graphs from graphite.

To re-create the problems, populate trends with lots of data, then try to do stuff.

#4 - 02/04/2015 09:56 AM - Shimon Shtein

- Related to Bug #7505: Trends are not aggregrated added

#5 - 02/04/2015 10:04 AM - Shimon Shtein

When #7505 will be merged, it should solve the issue, since it reduces the number of data points in the DB.

Besides it, "uptime" is a bit tricky to monitor using trends, because trends graphs are defined as "Number of hosts with selected fact, grouped by fact value".
So it will create meaningless groups each of them will contain a single host (or more, if by coincidence more than one server recorded the same uptime).

#6 - 05/09/2015 12:15 PM - The Foreman Bot

- Status changed from New to Ready For Testing

- Pull request https://github.com/theforeman/foreman/pull/2365 added

- Pull request deleted ()

#7 - 05/09/2015 12:17 PM - Jon McKenzie

Ignore the referenced pull request, that was meant for issue #5568.

#8 - 05/11/2015 04:37 AM - Dominic Cleal

- Status changed from Ready For Testing to New

- Pull request added

- Pull request deleted (https://github.com/theforeman/foreman/pull/2365)

#9 - 07/05/2015 04:06 AM - Ohad Levy

- Related to deleted (Bug #7505: Trends are not aggregrated)

#10 - 07/05/2015 04:06 AM - Ohad Levy

- Is duplicate of Bug #7505: Trends are not aggregrated added

#11 - 07/05/2015 04:06 AM - Ohad Levy

- Status changed from New to Duplicate