MediaWiki Performance Profiling

Asher Feldman
2012 WMF SF TechDays

Profiling in MediaWiki

- Profiling support is built-in but must be enabled
- Enabled via StartProfiler.php
  - Enable in your dev / labs instances
  - https://www.mediawiki.org/wiki/Manual:How_to_debug#Profiling
- Enabled in production for 2% of php requests

- Hooks are automatically profiled, so all extensions get overall coverage

- Other functions must contain wfProfile calls:

```
function doSomething() {
    wfProfileIn( __METHOD__ );

    # The actual function

    wfProfileOut( __METHOD__ );
}
```

- In the future we'll use xhprof and get automatic 100% coverage

- Default profiling might show an extension hook occasionally runs for 20 seconds

- Unless all supporting functions (and external resources) are profiled, expect lots of digging in order to debug

- But just being aware of a problem is half the battle!

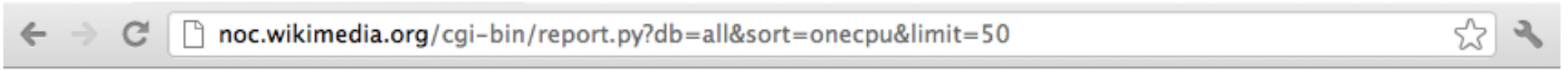
- Profile your PHP and SQL in development, but also check both in production after deployment

- The site didn't go down != everything is ok

- Look at 90th and 99th percentile metrics! Our performance long tail includes some of our most important pages, and today's edge case might be common tomorrow.

- noc.wikimedia.org provides a text interface to production profiling data

- Only displays averages for all data collected since the collector was restarted or manually cleared

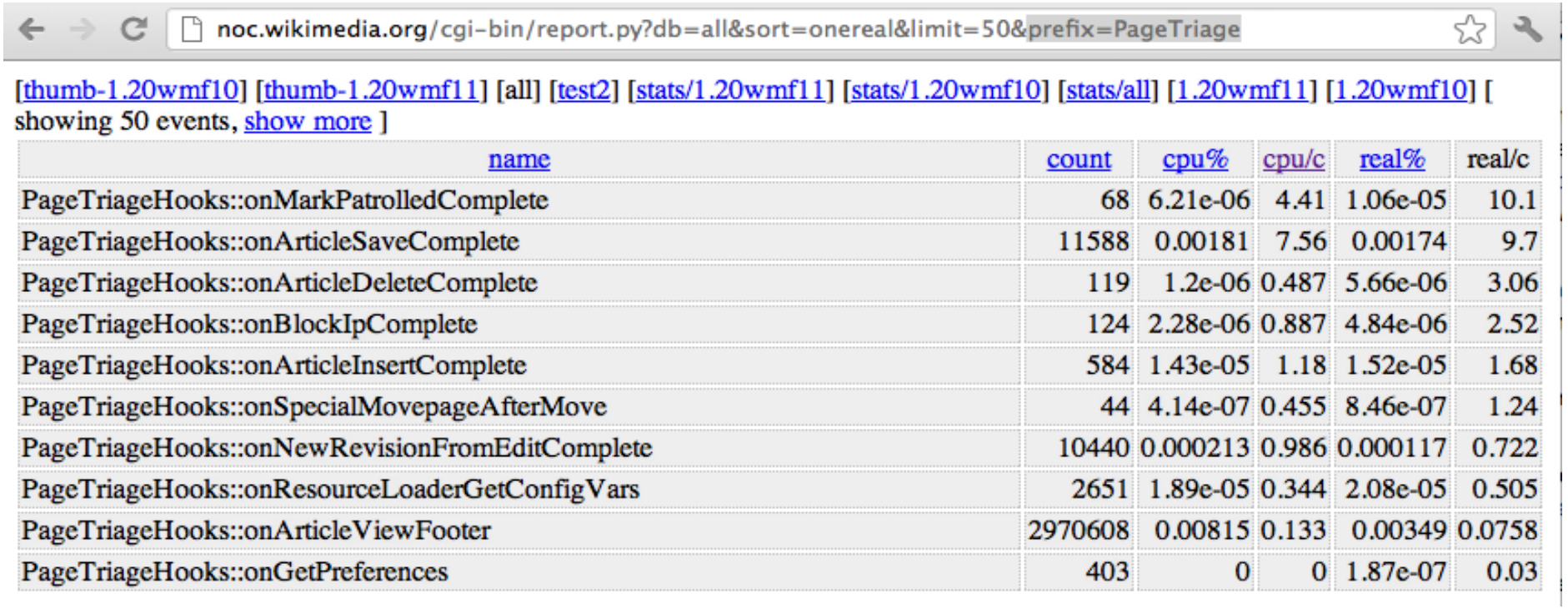


[thumb-1.20wmf10] [thumb-1.20wmf11] [all] [test2] [stats/1.20wmf11] [stats/1.20wmf10] [stats/all] [1.20wmf11] [1.20wmf10] [showing 50 events, show more]

name	count	cpu%	cpu/c	real%	real/c
API:rollback	93	0.00969	5.04e+03	0.00757	5.25e+03
Special:FundraiserStatistics	3	0.000267	4.3e+03	0.000201	4.32e+03
Parser::parse-AFComputedVariable::parseNonEditWikitext	296	0.0252	4.11e+03	0.0195	4.25e+03
Special:AbuseLog	261	0.0215	3.99e+03	0.0185	4.57e+03
Special:AbuseFilter	141	0.00858	2.94e+03	0.00668	3.06e+03
Parser::parse-FRInclusionCache::getRevIncludes	53	0.00265	2.42e+03	0.00206	2.51e+03
Parser::parse-FlaggedRevs::parseStableText	11590	0.433	1.81e+03	0.354	1.97e+03
API:pagetriagetagging	5	0.000186	1.8e+03	0.000169	2.18e+03
API:feedbackdashboardresponse	1	3.27e-05	1.58e+03	2.92e-05	1.88e+03
Special:Preferences	1320	0.0424	1.55e+03	0.0341	1.67e+03

- Search by prefix to see data for just your extension / class

- &prefix=PageTriage just returns PageTriageHooks methods == the extension contains no wfProfile calls



[thumb-1.20wmf10] [thumb-1.20wmf11] [all] [test2] [stats/1.20wmf11] [stats/1.20wmf10] [stats/all] [1.20wmf11] [1.20wmf10] [showing 50 events, show more]

name	count	cpu%	cpu/c	real%	real/c
PageTriageHooks::onMarkPatrolledComplete	68	6.21e-06	4.41	1.06e-05	10.1
PageTriageHooks::onArticleSaveComplete	11588	0.00181	7.56	0.00174	9.7
PageTriageHooks::onArticleDeleteComplete	119	1.2e-06	0.487	5.66e-06	3.06
PageTriageHooks::onBlockIpComplete	124	2.28e-06	0.887	4.84e-06	2.52
PageTriageHooks::onArticleInsertComplete	584	1.43e-05	1.18	1.52e-05	1.68
PageTriageHooks::onSpecialMovepageAfterMove	44	4.14e-07	0.455	8.46e-07	1.24
PageTriageHooks::onNewRevisionFromEditComplete	10440	0.000213	0.986	0.000117	0.722
PageTriageHooks::onResourceLoaderGetConfigVars	2651	1.89e-05	0.344	2.08e-05	0.505
PageTriageHooks::onArticleViewFooter	2970608	0.00815	0.133	0.00349	0.0758
PageTriageHooks::onGetPreferences	403	0	0	1.87e-07	0.03

- Problems surfaced in report.py are likely either problems on most calls, or extremely slow on some, enough so to skew the average

- Wallclock (but not cpu) times are fed into graphite every minute, allowing the data to be viewed as a time series and breaking free from averages (percentiles are calculated at ingestion time from buckets of the last 300 samples per metric)

- Cpu data from report.py can be a good augmentation but isn't always reliable

ApiMobileView tp{50,90,99}

- Graphite shows ApiMobileView::getData regularly takes 2-6+ seconds for 1% of requests (this could be a lot), but 90% take < 250ms, while 50% consistently < 14ms

- The avg in report.py is 179ms and 154ms cpu time

- Time spent waiting on network services doesn't count towards cpu time; if cpu and real are close, most of the time is spent in php code execution. No DB to blame here.

- Some functions could use multiple wfProfile points

```
127         private function getData( Title $title, $noImages ) {
130                 wfProfileIn( __METHOD__ );                          …
159                         $parserOutput = $wp->getParserOutput( $parserOptions );
...
185                         $data['sections'] = $parserOutput->getSections();
186                         $chunks = preg_split( '/<h(?=[1-6]\b)/i', $html );
187                         if ( count( $chunks ) != count( $data['sections'] ) + 1 ) {
...
195                         foreach ( $chunks as $chunk ) {
196                                 if ( count( $data['text'] ) ) {
197                                         $chunk = "<h$chunk";
198                                 }
199                                 if ( $wgUseTidy && count( $chunks ) > 1 ) {
200                                         $chunk = MWTidy::tidy( $chunk );
201                                 }
202                                 if ( preg_match( '/<ol\b[^>]*?class="references"/', $chunk ) )
{
203                                         $data['refsections'][count( $data['text'] )] = true;
204                                 }
205                                 $data['text'][] = $chunk;
206                         }
...
210                 wfProfileOut( __METHOD__ );
211                 return $data;
212         }
```

- ApiMobileView::getData slow times are probably due to getParserOutput → pcache misses, but lots happens after that might be expensive

MySQL Query Profiling

- MediaWiki profiling captures queries and times
- $wgDebugDumpSql can be configured to log all queries regardless of if a request is being profiled
- mysql slow query logging is much more detailed, especially in newer versions. On a dev instance, set long_query_time=0 and get that detail for everything
- All queries are commented with methods and user name / ip address – makes it easy to correlate queries to a specific request or action

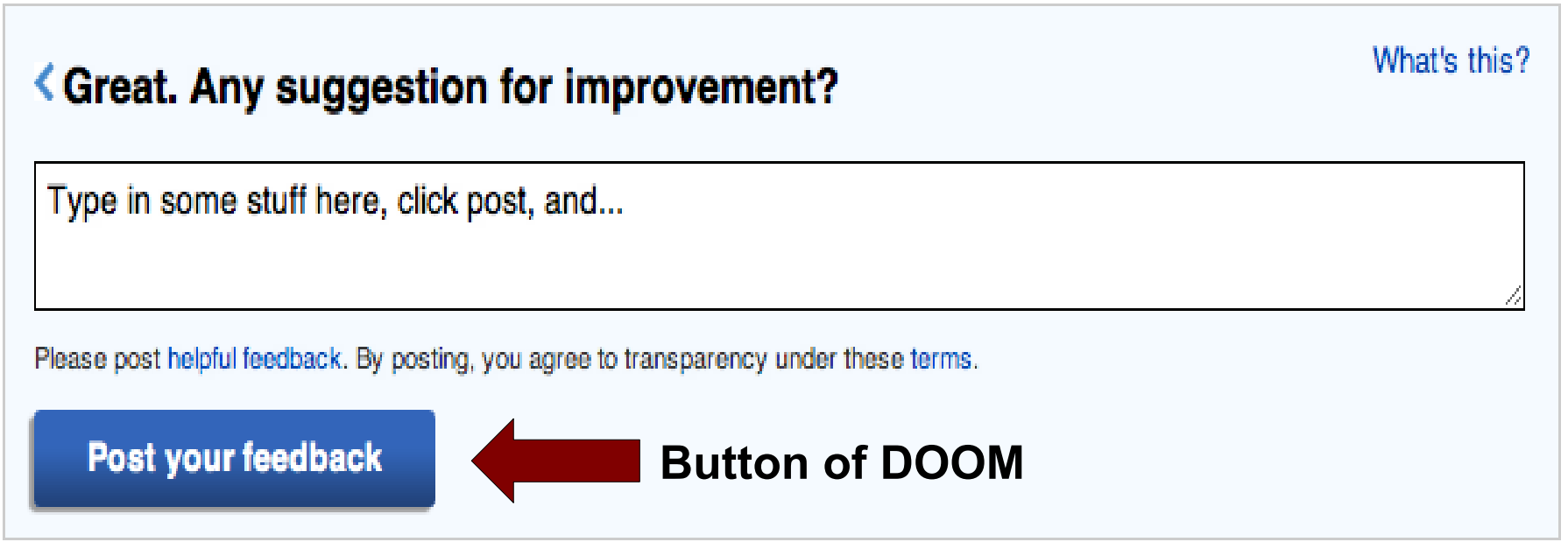
- Writing a new extension that hits the db?

- Inspect a log of generated queries as part of basic QA

- Check for unnecessary or duplicate queries and EXPLAIN read queries to ensure efficient index utilization

“Why should I look at a query log?”

Let's look at how ArticleFeedback (v5) as currently installed on enwiki writes to the database



19 write statements!

```
1. INSERT /* DatabaseBase::insert Asher Feldman */  INTO `aft_article_feedback`
(af_page_id,af_revision_id,af_created,af_us
er_id,af_user_ip,af_user_anon_token,af_form_id,af_experiment,af_link_id,af_has_comment) VALUES
('534366','506813755','20120813223135','14719981',NULL,'','6','M5_6','0','1')

2. INSERT /* ApiArticleFeedbackv5::saveUserRatings Asher Feldman */  INTO `aft_article_answer`
(aa_field_id,aa_response_rating,aa_response_text,aa_response_boolean,aa_response_option_id,aa_feedb
ack_id,aat_id) VALUES ('16',NULL,NULL,'1',NULL,'253294',NULL),('17',NULL,'Well sourced article!
(this is a test comment) ',NULL,NULL,'253294',NULL)

3. UPDATE /* ApiArticleFeedbackv5::saveUserRatings Asher Feldman */  `aft_article_feedback` SET
af_cta_id = '2' WHERE af_id = '253294'

4. INSERT /* ApiArticleFeedbackv5::saveUserProperties Asher Feldman */  INTO
`aft_article_feedback_properties` (afp_feedback_id,afp_key,afp_value_int) VALUES
('253294','contribs-lifetime','3'),('253294','contribs-6-months','0'),('253294','contribs-3-
months','0'),('253294','contribs-1-months','0')

5. INSERT /* ApiArticleFeedbackv5::updateRollupRow Asher Feldman */ IGNORE INTO
`aft_article_revision_feedback_ratings_rollup`
(afrr_page_id,afrr_revision_id,afrr_field_id,afrr_total,afrr_count) VALUES
('534366','506813755','16','0','0')

6. UPDATE /* ApiArticleFeedbackv5::updateRollupRow Asher Feldman */
`aft_article_revision_feedback_ratings_rollup` SET afrr_total = afrr_total + 1,afrr_count =
afrr_count + 1 WHERE afrr_page_id = '534366' AND afrr_revision_id = '506813755' AND afrr_field_id =
'16'

7. DELETE /* ApiArticleFeedbackv5::updateRollupRow Asher Feldman */ FROM
`aft_article_feedback_ratings_rollup` WHERE arr_page_id = '534366' AND arr_field_id = '16'

8. INSERT /* ApiArticleFeedbackv5::updateRollupRow Asher Feldman */ IGNORE INTO
`aft_article_feedback_ratings_rollup` (arr_page_id,arr_field_id,arr_total,arr_count) VALUES
('534366','16','9','42')
```

```
9. INSERT /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */ IGNORE INTO
`aft_article_filter_count` (afc_page_id,afc_filter_name,afc_filter_count) VALUES
('534366','visible','0'),('0','visible','0'),('534366','notdeleted','0'),('0','notdeleted','0'),
('534366','all','0'),('0','all','0'),('534366','visible-comment','0'),('0','visible-comment','0'),
('534366','visible-relevant','0'),('0','visible-relevant','0')

10. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '534366'
AND afc_filter_name = 'visible'

11. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '0' AND
afc_filter_name = 'visible'

12. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '534366'
AND afc_filter_name = 'notdeleted'

13. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '0' AND
afc_filter_name = 'notdeleted'

14. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '534366'
AND afc_filter_name = 'all'

15. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '0' AND
afc_filter_name = 'all'

16. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '534366'
AND afc_filter_name = 'visible-comment'

17. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '0' AND
afc_filter_name = 'visible-comment'
```

```
18. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '534366'
AND afc_filter_name = 'visible-relevant'

19. UPDATE /* ApiArticleFeedbackv5Utils::updateFilterCounts Asher Feldman */
`aft_article_filter_count` SET afc_filter_count = afc_filter_count + 1 WHERE afc_page_id = '0' AND
afc_filter_name = 'visible-relevant'
```

- Note the multiple counter rows with id = '0' updated every time feedback is given on any page

- Note the use of DELETE + INSERT IGNORE to update a single row

- Both result in locks that prevent >1 feedback submission saving at a time (due to the use of txns, these locks persist beyond than the time needed by the individual statements)

- Note all the dead server kitties!

All slow (>= long_query_time) queries are logged

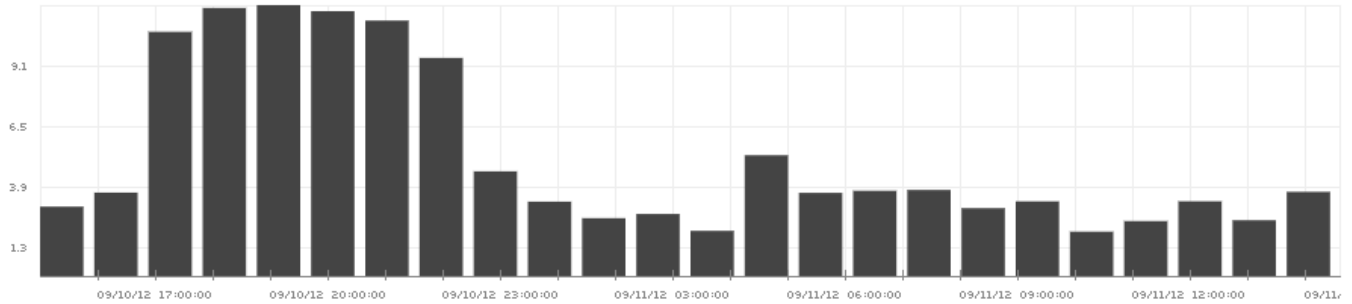
https://ishmael.wikimedia.org/?host=db63&hours=24&sort=ratio

ishmael a UI for pt-query-digest

last 24 hours on db63 go

Dashboard for db63

Average Slow Query Time



Slow Query Overview

Rank	% time	% queries	Ratio	sample query	
1	31.44 (2624.257)	8.08 (140)	3.89	`/* MathRenderer::render */ replace into math (math_inputhash, math_outputhash, math_html_conservativeness, math_html, math_mathml) values(?+)`	more
2	0.23 (19.195)	0.12 (2)	1.99	`/* SpecialEditWatchlist::clearWatchlist */ DELETE FROM watchlist WHERE wl_user = ?`	more
3	49.32 (4116.837)	38.97 (675)	1.27	`/* User::invalidateCache */ UPDATE user set user_touched = ? WHERE user_id = ? AND (user_touched < ?)`	more
4	0.07 (5.544)	0.06 (1)	1.15	`/* LinksUpdate::incrTableUpdate */ DELETE FROM pagelinks WHERE pl_from = ? AND ((pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title = ?) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title = ?))`	more
5	0.06 (5.344)	0.06 (1)	1.11	`/* LinksUpdate::incrTableUpdate */ DELETE FROM pagelinks WHERE pl_from = ? AND ((pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title = ?) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title = ?))`	more
6	0.06 (5.014)	0.06 (1)	1.04	`/* LinksUpdate::incrTableUpdate */ DELETE FROM pagelinks WHERE pl_from = ? AND ((pl_namespace = ? AND pl_title IN(?+) ) OR (pl_namespace = ? AND pl_title = ?) OR (pl_namespace = ? AND pl_title = ?))`	more
7	0.06 (4.904)	0.06 (1)	1.02	`/* LinksUpdate::incrTableUpdate */ DELETE FROM pagelinks WHERE pl_from = ? AND ((pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title = ?) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title = ?))`	more
8	0.05 (4.424)	0.06 (1)	0.92	`/* WikiPage::updateRevisionOn */ UPDATE page set page_latest = ?,page_touched = ?,page_is_new = ?,page_is_redirect = ?,page_len = ? WHERE page_id = ? AND page_latest = ?`	more

And 90 seconds of all queries per hour (2.5%)

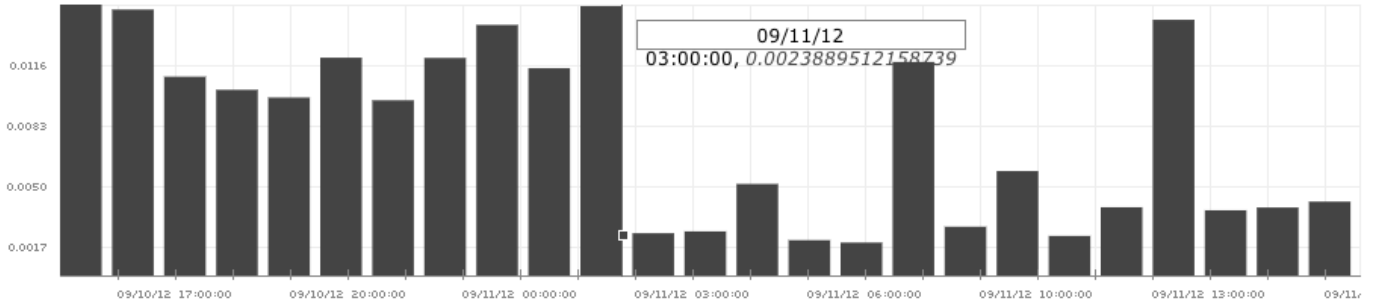
https://ishmael.wikimedia.org/sample/?host=db63&hours=24&sort=ratio

ishmael a UI for mk-query-digest

last 24 hours on db63 go

Dashboard for db63

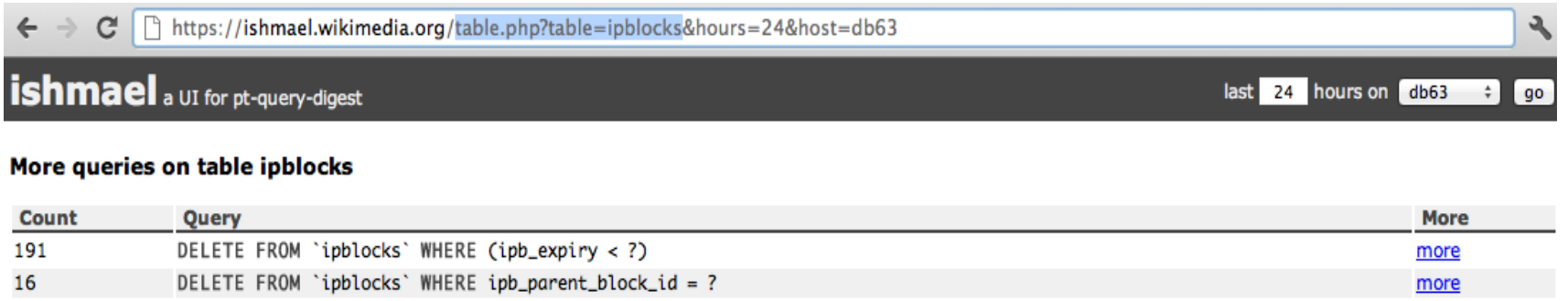
Average Slow Query Time



Slow Query Overview

Rank	% time	% queries	Ratio	sample query	
1	0.22 (22.115)	0.00 (24)	117.59	/* Revision::fetchFromConds */ SELECT rev_id, rev_page, rev_text_id, rev_timestamp, rev_comment, rev_user_text, rev_user, rev_minor_edit, rev_deleted, rev_len, rev_parent_id, rev_sha?,page_namespace, page_title, page_id, page_latest, page_is_redirect, page_len, user_name FROM `revision` inner JOIN `page` ON ((page_id = rev_page)) LEFT JOIN `user` ON ((rev_user != ?) AND (user_id = rev_user)) WHERE rev_timestamp = ? AND page_namespace = ? AND page_title = ? limit ?	more
2	0.05 (4.964)	0.00 (11)	57.55	/* LinksUpdate::incrTableUpdate */ DELETE FROM `page_props` WHERE pp_page = ? AND (pp_propname IN(?+))	more
3	1.86 (190.126)	0.03 (453)	53.56	/* Job::pop_type */ DELETE FROM `job` WHERE job_id = ?	more
4	0.22 (22.675)	0.00 (63)	45.92	/* LinksUpdate::incrTableUpdate */ DELETE FROM `templatelinks` WHERE tl_from = ? AND ((tl_namespace = ? AND tl_title = ?))	more
5	0.43 (44.125)	0.01 (124)	45.41	/* MathRenderer::render */ replace into `math` (math_inputhash, math_outputhash, math_html_conservativeness, math_html, math_mathml) values(?+)	more
6	1.08 (110.536)	0.03 (369)	38.23	/* HTMLCacheUpdate::invalidateTitles */ UPDATE `page` set page_touched = ? WHERE page_id IN(?+)	more
7	13.40 (1367.157)	0.38 (4927)	35.41	commit	more
8	1.34 (137.096)	0.04 (511)	34.24	/* Job::removeDuplicates */ DELETE FROM `job` WHERE job_cmd = ? AND job_namespace = ? AND job_title = ? AND job_timestamp = ? AND job_params = ?	more
9	0.00 (0.264)	0.00 (1)	32.73	/* LinksUpdate::incrTableUpdate */ DELETE FROM `pagelinks` WHERE pl_from = ? AND ((pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title IN(?+)) OR (pl_namespace = ? AND pl_title = ?) OR (pl_namespace = ? AND pl_title IN(?+)))	more

Searchable by table



- Slow and sample reports linked to for all dbs from http://noc.wikimedia.org/dbtree/

- Check if new code shows up in slow at all

- Check sample to see if queries are executing more often than necessary (put a cache on it)

- Check the master and one slave for the largest wiki the code is live on