

Practical Performance

Tips & Tricks



**Ronald Bradford
Senior Consultant
MySQL Inc
July 2007**

Agenda (1 hr)

- Basic (15 mins)
- ♦ Intermediate (15 mins)
- ♦ Advanced (15 mins)
- ♦ New Product - MySQL Proxy (15 mins)

Basic

Basic Agenda (15 mins)

- What to Monitor
- Essential Tools (OS/MySQL)
- MySQL Tools
- MySQL Commands

What to Monitor

- ♦ Operating System
 - ♦ Processes/Disk/Memory/Swap/Network
- ♦ MySQL Connections/Threads
- ♦ MySQL Status & Variables
- ♦ MySQL Table Status & Storage Engine Status
- ♦ MySQL Slow Queries
- ♦ MySQL All Queries

Monitor Everything!

Essential OS Tools

Operating System

- ♦ vmstat, iostat, netstat, sar, mpstat, prstat, prsinfo
- ♦ dstat, **mtstat**
- ♦ top, uptime, ps, dd
- ♦ free
- ♦ strace, dtrace

Learn to understand your OS tools

Essential OS Tools - Example

- ◆ vmstat output - Identify the 3 key points in 10 seconds

```
$ vmstat 5
```

kthr			memory		page							disk				faults			cpu		
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	s0	s1	s2	s4	in	sy	cs	us	sy	id
1	0	0	17114496	2962832	63	211	5180	71	71	0	0	0	0	71	0	4763	4625	3183	72	2	26
1	0	0	17115252	2963184	4	6	439	0	0	0	0	0	0	7	0	3251	4021	2795	79	2	20
1	0	0	17115252	2963144	2	1	0	0	0	0	0	0	0	16	0	3748	4427	3049	89	2	10
1	0	0	17115252	2962912	11	11	1360	0	0	0	0	0	0	47	0	4083	4210	2752	79	2	19
0	0	0	17115248	2962744	4	9	1428	0	0	0	0	0	0	10	0	1072	1015	754	17	1	82
1	0	0	17115248	2962664	4	0	0	0	0	0	0	0	0	42	0	3755	3818	2549	69	2	29
1	0	0	17115240	2962520	2	4	246	0	0	0	0	0	0	3	0	3231	3992	2833	79	1	19
1	0	0	17115228	2962400	3	8	1347	0	0	0	0	0	0	10	0	3706	4339	3063	88	2	10
1	0	0	17115220	2962256	16	11	853	0	0	0	0	0	0	52	0	4275	4201	2672	79	2	19
1	0	0	17115220	2962100	3	5	131	0	0	0	0	0	0	7	0	2742	3275	2385	62	2	36
0	0	0	17115220	2962064	3	0	0	0	0	0	0	0	0	29	0	1887	1247	917	19	1	81
1	0	0	17115216	2961664	7	13	1885	0	0	0	0	0	0	18	0	3597	4674	3270	82	2	17
2	0	0	17114344	2961428	75	281	1953	13	13	0	0	0	0	12	0	3697	4746	3203	87	2	11
1	0	0	17114332	2961396	19	14	211	6	6	0	0	0	0	53	0	4409	4476	2862	77	2	21
1	0	0	17115316	2961544	5	7	426	0	0	0	0	0	0	8	0	2658	3478	2371	64	1	34
0	0	0	17115316	2961500	5	0	0	0	0	0	0	0	0	39	0	2310	1802	1268	26	1	73
1	0	0	17115316	2961428	5	7	823	0	0	0	0	0	0	11	0	3454	4473	3096	82	2	16
1	0	0	17115308	2961224	7	11	920	0	0	0	0	0	0	13	0	3569	4377	3024	86	2	12
1	0	0	17115308	2961016	12	9	870	3	3	0	0	0	0	37	0	4687	5227	3250	82	2	17

Practical Performance Tips & Tricks

Essential OS Tools - Example

◆ vmstat output – Point 1

```
$ vmstat 5
```

kthr			memory		page							disk		faults			cpu	
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	s0	s1				us	sy
1	0	0	17114496	2962832	63	211	5180	71	71	0	0	0	0					
1	0	0	17115252	2963184	4	6	439	0	0	0	0	0	0					
1	0	0	17115252	2963144	2	1	0	0	0	0	0	0	0					
1	0	0	17115252	2962912	11	11	1360	0	0	0	0	0	0					
0	0	0	17115248	2962744	4	9	1428	0	0	0	0	0	10	0	1072	1015	75	1 82
1	0	0	17115248	2962664	4	0	0	0	0	0	0	0	42	0	3755	3818	2549	2 29
1	0	0	17115240	2962520	2	4	246	0	0	0	0	0	3	0	3231	3992	2833	7 19
1	0	0	17115228	2962400	3	8	1347	0	0	0	0	0	10	0	3706	4339	3063	88 10
1	0	0	17115220	2962256	16	11	853	0	0	0	0	0	52	0	4275	4201	2672	79 2 19
1	0	0	17115220	2962100	3	5	131	0	0	0	0	0	7	0	2742	3275	2385	62 2 36
0	0	0	17115220	2962064	3	0	0	0	0	0	0	0	29	0	1887	1247	917	19 1 81
1	0	0	17115216	2961664	7	13	1885	0	0	0	0	0	18	0	3597	4674	3270	82 2 17
2	0	0	17114344	2961428	75	281	1953	13	13	0	0	0	12	0	3697	4746	3203	87 2 11
1	0	0	17114332	2961396	19	14	211	6	6	0	0	0	53	0	4409	4476	2862	77 2 21
1	0	0	17115316	2961544	5	7	426	0	0	0	0	0	8	0	2658	3478	2371	64 1 34
0	0	0	17115316	2961500	5	0	0	0	0	0	0	0	39	0	2310	1802	1268	26 1 73
1	0	0	17115316	2961428	5	7	823	0	0	0	0	0	11	0	3454	4473	3096	82 2 16
1	0	0	17115308	2961224	7	11	920	0	0	0	0	0	13	0	3569	4377	3024	86 2 12
1	0	0	17115308	2961016	12	9	870	3	3	0	0	0	37	0	4687	5227	3250	82 2 17

Significant drop
in CPU usage

Essential OS Tools - Example

- ◆ vmstat output – Point 2

```
$ vmstat 5
```

r b w			memory				page				disk				faults			cpu			
sw	fr	re	mf	pi	po	fr	de	sr	s0	s1	s2	s4	in	sy	cs	us	sy	id			
1	0	0	17114496	2962832	63	211	5180	71	71	0	0	0	71	0	4763	4625	3183	72	2	26	
1	0	0	17115228	2962832	63	211	5180	71	71	0	0	0	7	0	3251	4021	2795	79	2	20	
1	0	0	17115228	2962832	63	211	5180	71	71	0	0	0	16	0	3748	4427	3049	89	2	10	
1	0	0	17115228	2962832	63	211	5180	71	71	0	0	0	47	0	4083	4210	2752	79	2	19	
0	0	0	17115228	2962832	63	211	5180	71	71	0	0	0	10	0	1072	1015	754	17	1	82	
1	0	0	17115228	2962832	63	211	5180	71	71	0	0	0	42	0	3755	3818	2549	69	2	29	
1	0	0	17115240	2962832	63	211	5180	71	71	0	0	0	3	0	3231	3992	2833	79	1	19	
1	0	0	17115228	2962832	63	211	5180	71	71	0	0	0	10	0	3706	4339	3063	88	2	10	
1	0	0	17115228	2962832	63	211	5180	71	71	0	0	0	52	0	4275	4201	2672	79	2	19	
1	0	0	17115228	2962832	63	211	5180	71	71	0	0	0	7	0	2742	3275	2385	62	2	36	
0	0	0	17115228	2962832	63	211	5180	71	71	0	0	0	29	0	1887	1247	917	19	1	81	
1	0	0	17115216	2961664	7	13	1885	0	0	0	0	0	18	0	3597	4674	3270	82	2	17	
2	0	0	17114344	2961428	75	281	1953	13	13	0	0	0	12	0	3697	4746	3203	87	2	11	
1	0	0	17114332	2961396	19	14	211	6	6	0	0	0	53	0	4409	4476	2862	77	2	21	
1	0	0	17115316	2961544	5	7	426	0	0	0	0	0	8	0	2658	3478	2371	64	1	34	
0	0	0	17115316	2961500	5	0	0	0	0	0	0	0	39	0	2310	1802	1268	26	1	73	
1	0	0	17115316	2961428	5	7	823	0	0	0	0	0	11	0	3454	4473	3096	82	2	16	
1	0	0	17115308	2961224	7	11	920	0	0	0	0	0	13	0	3569	4377	3024	86	2	12	
1	0	0	17115308	2961016	12	9	870	3	3	0	0	0	37	0	4687	5227	3250	82	2	17	

Practical Performance Tips & Tricks

Essential OS Tools - Example

◆ vmstat output – Point 3

```
$ vmstat 5
```

kthr			memory		page				disk				faults		cpu						
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	s0	s1	s2	s4	in	sy	cs	us	sy	id
1	0	0	17114496	2962832	63	211	5180	71	71	0	0	0	0	71	0	4763	4625	3183	72	2	26
1	0	0	17115252	2963184	4	6	439	0	0	0	0	0	0	7	0	3251	4021	2795	79	2	20
1	0	0	17115252	2963144	2	1	0	0	0	0	0	0	0	0	0	0	0	49	89	2	10
1	0	0	17115252	2962912	11	11	1360	0	0	0	0	0	0	0	0	0	0	52	79	2	19
0	0	0	17115248	2962744	4	9	1428	0	0	0	0	0	0	0	0	0	0	54	17	1	82
1	0	0	17115248	2962664	4	0	0	0	0	0	0	0	0	0	0	0	0	49	69	2	29
1	0	0	17115240	2962520	2	4	246	0	0	0	0	0	0	0	0	0	0	33	79	1	19
1	0	0	17115228	2962400	3	8	1347	0	0	0	0	0	0	10	0	3706	4339	3063	88	2	10
1	0	0	17115220	2962256	16	11	853	0	0	0	0	0	0	52	0	4275	4201	2672	79	2	19
1	0	0	17115220	2962100	3	5	131	0	0	0	0	0	0	7	0	2742	3275	2385	62	2	36
0	0	0	17115220	2962064	3	0	0	0	0	0	0	0	0	29	0	1887	1247	917	19	1	81
1	0	0	17115216	2961664	7	13	1885	0	0	0	0	0	0	18	0	3597	4674	3270	82	2	17
2	0	0	17114344	2961428	75	281	1953	13	13	0	0	0	0	12	0	3697	4746	3203	87	2	11
1	0	0	17114332	2961396	19	14	211	6	6	0	0	0	0	53	0	4409	4476	2862	77	2	21
1	0	0	17115316	2961544	5	7	426	0	0	0	0	0	0	8	0	2658	3478	2371	64	1	34
0	0	0	17115316	2961500	5	0	0	0	0	0	0	0	0	39	0	2310	1802	1268	26	1	73
1	0	0	17115316	2961428	5	7	823	0	0	0	0	0	0	11	0	3454	4473	3096	82	2	16
1	0	0	17115308	2961224	7	11	920	0	0	0	0	0	0	13	0	3569	4377	3024	86	2	12
1	0	0	17115308	2961016	12	9	870	3	3	0	0	0	0	37	0	4687	5227	3250	82	2	17

Significant paging

Monitoring Analysis

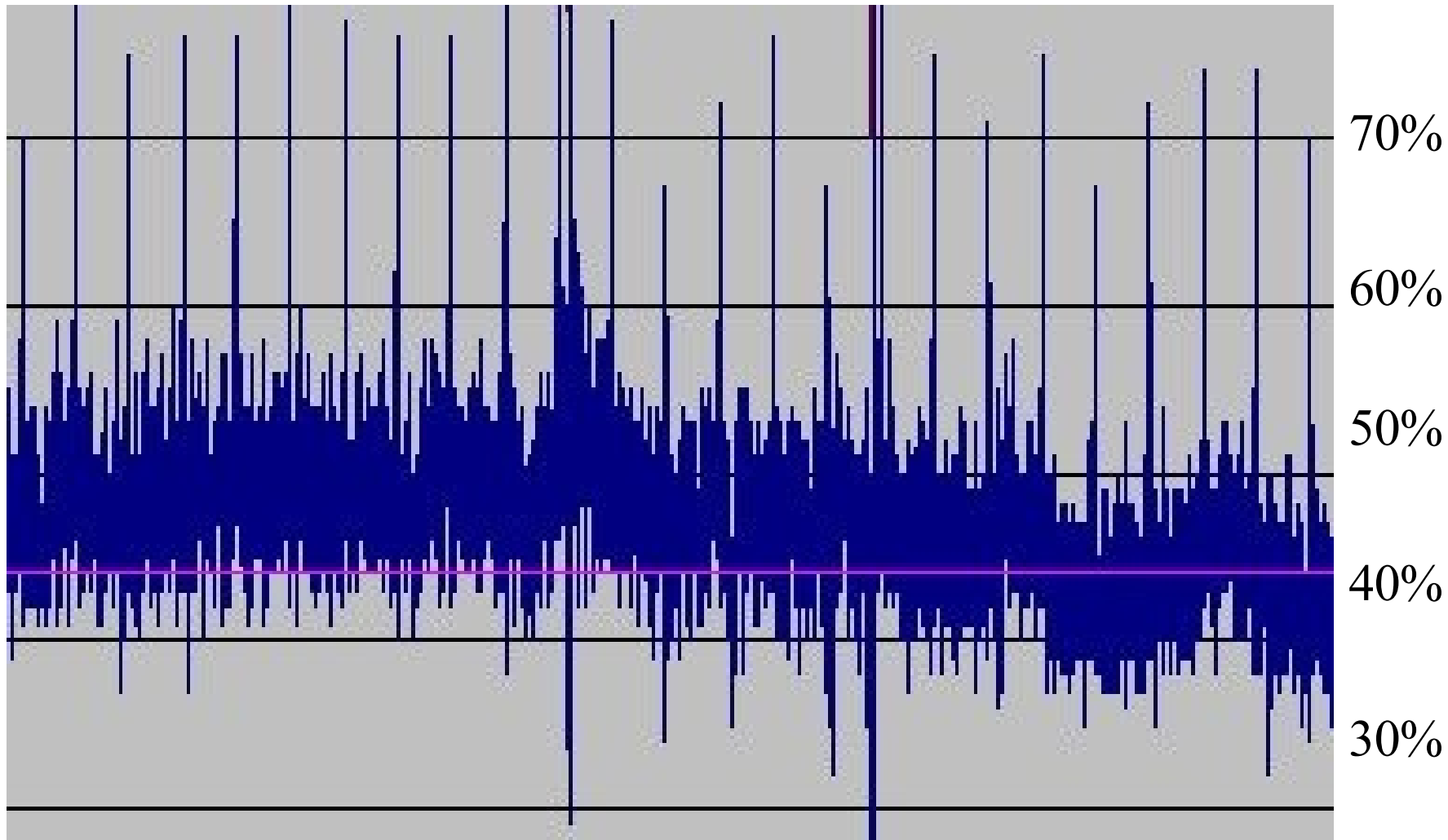
- ♦ Difficult to crunch data
- ♦ What do you look for?

Monitor Change

- ♦ Visual Analysis is a great guide

Graph Results

CPU Graph (5 minute Spikes) – Example 2



Practical Performance Tips & Tricks

Monitor Everything

Essential Tools - MySQL

Tool Set Technologies

- ♦ Perl DBI, DBD::mysql, Term:ReadKey, Time::HiRes
- ♦ Python 2.2+, MySQLdb

Practical Performance Tips & Tricks

Essential Tools - MySQL

MySQL Related

- ♦ mytop
- ♦ innotop
- ♦ mysqltoolkit
- ♦ mybench
- ♦ statpack
- ♦ mtstat
- ♦ MySQL Proxy

MySQL Threads

- ◆ Each MySQL Connection takes a thread
- ◆ Threads may come from thread cache

```
mysql> SHOW [FULL] PROCESSLIST;
```

```
mysql> SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST; (5.1)
```

```
$ mysqladmin [-v|--verbose] processlist
```

Common Observation:

“too many connections”

MySQL will keep 1 thread available for SUPER privileges

MySQL Status

- ◆ Underlying counters of Server Internals
 - ◆ 5.0 has 249 values
 - ◆ 5.1 has 258 values
 - ◆ 6.0 has 260 values

```
mysql> SHOW [GLOBAL|SESSION] STATUS;
```

```
mysql> SELECT * FROM INFORMATION_SCHEMA.[GLOBAL|  
    SESSION]_STATUS; (5.1)
```

```
$ mysqladmin extended-status
```

MySQL Variables

- ◆ Configurable Variables
 - ◆ 5.0 224 values
 - ◆ 5.1 235 values
 - ◆ 6.0 249 values
- ◆ Key to monitor with STATUS over time

```
mysql> SHOW [GLOBAL|SESSION] VARIABLES;
```

```
mysql> SELECT * FROM INFORMATION_SCHEMA.[GLOBAL|  
    SESSION]_VARIABLES; (5.1)
```

```
$ mysqladmin variables
```

General Query Log

- ♦ Capture of all SQL Statements
 - ♦ Per instance only
 - ♦ Instance reboot required
 - ♦ --log
- ♦ Bulk Explain Analysis
- ♦ Log Tables, no reboot required 5.1
 - ♦ --log-output=TABLE,FILE

```
mysql> SET GLOBAL general_log = 'ON';
```

```
mysql> SELECT * FROM general_log ORDER BY start_time DESC
```

Slow Query Log

- ◆ Captures Slow SQL Statements
 - ◆ queries not using indexes
 - ◆ `--log-slow-queries`
 - ◆ `--log-queries-not-using-indexes`
 - ◆ `long_query_time = 1`
- ◆ Bulk Explain Analysis
 - ◆ Identify top offenders (time, occurrence)

```
$ mysqldumpslow -s t slow.log > slow.time.log
```

```
$ mysqldumpslow -s c slow.log > slow.count.log
```

- ◆ Log Tables, no reboot required 5.1

Review

- ♦ Monitor Everything
- ♦ Know the basic MySQL commands
- ♦ Install all tools ahead of time

Intermediate

Practical Performance Tips & Tricks

Intermediate Agenda (15 mins)

- Monitoring MySQL Status
- Analyzing MySQL Status
- Reviewing SQL Statements
- Benchmark

Monitoring MySQL Status

```
$ mysqladmin -i 1 -r extended-status | grep -v "| 0 "
```

- ♦ -i 1 Iteration Interval (1 second)
- ♦ -r Show relative change between values
- ♦ -c 999 Number of iterations
- ♦ grep -v "| 0 " Note: space before & after 0

Monitoring MySQL Status

◆ Determine Idle State

```
$ mysqladmin -r -i 1 -c 3 extended-status | grep -v "| 0 "
```

Variable_name	Value
Bytes_received	35
Bytes_sent	6265
Com_show_status	1
Created_tmp_tables	1
Handler_read_rnd_next	250
Handler_write	249
Questions	1
Select_scan	1
Uptime	1

Practical Performance Tips & Tricks

Monitoring MySQL Status

Example 1: (1 second snapshot)

Variable_name	Value	
Bytes_received	14079832	← 13M in
Bytes_sent	23142360	← 22M out
Com_insert	4084	←
Com_select	3075	← True TPS = 7159
Com_show_status	1	
Com_stmt_execute	3075	
Created_tmp_tables	1	
Handler_read_key	10234	
Handler_read_rnd_next	248	
Handler_update	4084	
Handler_write	4331	
Questions	21448	
Select_scan	1	
Table_locks_immediate	9085	
Table_locks_waited	1136	
Threads_running	-2	
Uptime	1	

Practical Performance Tips & Tricks

Monitoring MySQL Status

Example 2: (1 second snapshot)

Variable_name	Value		
Aborted_clients	4	Key_read_requests	13310
Bytes_received	265345	Key_reads	1
Bytes_sent	347071	Key_write_requests	31
Com_admin_commands	64	Key_writes	15
Com_delete	4	Qcache_free_blocks	-178
Com_insert	4	Qcache_free_memory	-277232
Com_select	254	Qcache_hits	118
Com_set_option	146	Qcache_inserts	222
Com_show_status	1	Qcache_not_cached	33
Com_update	28	Qcache_queries_in_cache	188
Created_tmp_disk_tables	19	Qcache_total_blocks	200
Created_tmp_tables	266	Questions	555
Handler_delete	2	Select_full_join	230
Handler_read_first	34	Select_full_range_join	1
Handler_read_key	3022	Select_range	5
Handler_read_next	2587	Select_scan	167
Handler_read_rnd	291	Sort_range	1
Handler_read_rnd_next	112730	Sort_rows	291
Handler_update	1359	Sort_scan	258
Handler_write	745	Table_locks_immediate	1025
		Threads_cached	2
		Threads_connected	-2
		Uptime	1

Practical Performance Tips & Tricks

Monitoring MySQL Status

Example 2: (1 second snapshot)

Variable_name	Value
Aborted_clients	4
Bytes_received	265345
Bytes_sent	347071
Com_admin_commands	64
Com_delete	4
Com_insert	4
Com_select	254
Com_set_option	146
Com_show_status	1
Com_update	28
Created tmp disk tables	19
Created tmp tables	266
Handler_delete	2
Handler_read_first	34
Handler_read_key	3022
Handler_read_next	2587
Handler_read_rnd	291
Handler_read_rnd_next	112730
Handler_update	1359
Handler_write	745

Likely unnecessary commands causing network traffic.
e.g. SET autocommit=1;
Caused by Cold Fusion configuration

Tmp Tables being flushed to disk (per second)
Cause: filesort to disk and TEXT/BLOB's in join/sort results

Tmp Tables being created (per second) Cause: filesort

See Select_scan – full table scan

Practical Performance Tips & Tricks

Monitoring MySQL Status

Joins without Indexes
Join with full table scan

Key_read_requests	13310
Key_reads	1
Key_write_requests	31
Key_writes	15
Qcache_free_blocks	-178
Qcache_free_memory	-277232
Qcache_hits	118
Qcache_inserts	222
Qcache_not_cached	33
Qcache_queries_in_cache	188
Qcache_total_blocks	200
Questions	555
Select_full_join	230
Select_full_range_join	1
Select_range	5
Select_scan	167
Sort_range	1
Sort_rows	291
Sort_scan	258
Table_locks_immediate	1025
Threads_cached	2
Threads_connected	-2
Uptime	1

Practical Performance Tips & Tricks

Analyzing MySQL Status

- ◆ Stat Pack (following output)

```
./statpack.py --files=file1.txt,file2.txt
```

Analyzing MySQL Status

Variable	Delta/Percentage	Per Second	Total
Threads Connected:	92		343
Threads Running:	0		1
Questions:	1,581,058	456.03	10,743,965
Bytes Recieved:	430M	127K	3G
Bytes Sent:	-2,784,778,849B	-803,224B	1G
Aborted Clients:	8,428	2.43	51,727
Aborted Connects:	1,439	0.42	10,314

Statement Activity			
SELECT:	520,435	150.11	3,658,966 (79.53%)
INSERT:	15,933	4.60	125,693 (2.73%)
UPDATE:	88,059	25.40	701,451 (15.25%)
DELETE:	14,472	4.17	114,465 (2.49%)
REPLACE:	0	0.00	0 (0.00%)
INSERT ... SELECT:	0	0.00	0 (0.00%)
REPLACE ... SELECT:	0	0.00	0 (0.00%)
Multi UPDATE:	1	0.00	16 (0.00%)
Multi DELETE:	0	0.00	0 (0.00%)
COMMIT:	0	0.00	0 (0.00%)
ROLLBACK:	0	0.00	0 (0.00%)

Practical Performance Tips & Tricks

Analyzing MySQL Status

Variable	Delta/Percentage	Per Second	Total
=====			
Thread Cache			
=====			
Thread Efficiency:	100.00%		
Connections:	7,288	2.10	47,758
Threads Created:	239	0.07	1,018
=====			
Table Cache			
=====			
table_cache Efficiency:	99.79%		
Open Tables:	113	0.03	3,343
Opened Tables:	113	0.03	3,349
=====			
MyISAM Key Cache			
=====			
Cache Read Efficiency:	99.79%		
Cache Write Efficiency:	54.56%		
Memory Used:	0B		14K
Memory Free:	-307B		1B
Key Reads:	91,479	26.39	684,541
Key Read Requests:	45,513,248	13,127.56	325,559,458
Key Writes:	84,152	24.27	571,374
Key Write Requests:	183,810	53.02	1,257,477
Blocks Not Flushed:	0	0.00	0

Practical Performance Tips & Tricks

Analyzing MySQL Status

Variable	Delta/Percentage	Per Second	Total
=====			
Index Usage			
=====			
Index Efficiency:	23.48%		
Full Index Scans:	33,877	9.77	239,484
Full Table Scans:	177,534	51.21	1,312,803
Full Join Scans:	238,381	68.76	1,758,889
Handler_read_first:	33,877	9.77	239,484 (0.01%)
Handler_read_key:	10,825,843	3,122.54	85,491,377 (3.44%)
Handler_read_next:	81,424,947	23,485.71	497,768,588 (20.02%)
Handler_read_prev:	111,451	32.15	455,774 (0.02%)
Handler_read_rnd:	4,616,452	1,331.54	31,472,251 (1.27%)
Handler_read_rnd_next:	251,403,184	72,513.18	1,871,522,609 (75.25%)
=====			
Temporary Space			
=====			
tmp_table_size Efficiency:	89.93%		
Memory Temp Tables:	340,965	98.35	2,469,961
Disk Temp Tables:	34,048	9.82	248,682
Temp Files:	94	0.03	640

Practical Performance Tips & Tricks

Analyzing MySQL Status

Variable	Delta/Percentage	Per Second	Total
=====			
Lock Contention			
=====			
Percent of Locks Waited:	0.06%		
Table Locks Waited:	968	0.28	6,600
Table Locks Immediate:	1,494,632	431.10	10,758,207
=====			
Sorting			
=====			
Rows Sorted:	4,749,099	1,369.80	32,630,832
Sort Range:	51,689	14.91	325,228
Sort Scan:	329,879	95.15	2,395,994
Sort Merge Passes:	47	0.01	325
Full Range Joins:	2,010	0.58	14,484

Explain Everything

- ♦ Enable General Log
- ♦ Access every page in single user mode
- ♦ Run & Time every SQL Statement
- ♦ Explain every statement

Benchmark

- ♦ Identified Slow Query
- ♦ Determined improvement (rewrite, index etc)
- ♦ Benchmark (in isolation, under load)
- ♦ mybench is a quick & simple infrastructure

Benchmark (mybench)

```
$ ./test.mybench.sh -t 1 -i 100
```

```
test: 237 0.040418 0.163335 0.0689579578059071 16.343036  
      14.501589545541  
clients : 1  
queries : 237  
fastest  : 0.040418  
slowest  : 0.163335  
average  : 0.0689579578059071  
serial   : 16.343036  
q/sec    : 14.501589545541
```

Test with 5 threads, then 10 etc

Example Memory Benchmark

Evaluating two different
H/W platforms for client

```
call load_ids(1000000);
+-----+
| COUNT(*) |
+-----+
| 1000000 |
+-----+
1 row in set (19.12 sec)
```

kthr			memory			page								disk		faults				cpu	
r	b	w	swap	free	re	mf	pi	po	fr	de	sr	s0	s2	sd	sd	in	sy	cs	us	sy	id
0	0	40	14582440	279416	0	0	0	0	0	0	0	0	0	1	1	316	217	123	0	0	100
0	0	40	14582440	279272	1	0	0	0	0	0	0	0	0	3	3	362	6539	180	0	0	99
0	0	40	14582440	278096	0	0	0	0	0	0	0	0	0	19	19	368	38703	126	3	1	97
0	0	40	14582440	276952	0	0	0	0	0	0	0	0	0	18	18	375	37558	135	3	1	97
0	0	40	14582440	275832	0	0	0	0	0	0	0	0	0	17	17	381	36803	151	3	1	97
0	0	40	14582440	274704	0	0	0	0	0	0	0	0	0	18	18	372	36930	134	3	1	97

```
call load_ids(1000000);
+-----+
| COUNT(*) |
+-----+
| 1000000 |
+-----+
1 row in set (3.26 sec)
```

0	0	34	10021408	743724	0	0	0	0	0	0	0	0	0	0	0	673	183	507	0	0	100
0	0	34	10021408	743724	0	0	0	0	0	0	0	0	0	0	0	675	192	520	0	1	99
0	0	34	10021408	743724	0	1	0	0	0	0	0	0	0	1	0	3144	40355	524	3	1	96
0	0	34	10021408	743720	0	0	0	0	0	0	0	0	0	7	0	14777	213483	489	20	5	74
0	0	34	10021408	743720	0	0	0	0	0	0	0	0	0	8	0	16316	214255	503	21	5	74
0	0	34	10021408	743720	0	0	0	0	0	0	0	0	0	8	0	14690	210905	509	21	7	73

Practical Performance Tips & Tricks

Review

- ♦ Monitor Everything
- ♦ If you don't know what to look for - Monitor Change
- ♦ Capture all SQL via application
- ♦ Explain everything
- ♦ Time everything
- ♦ Monitor SQL & Time over time

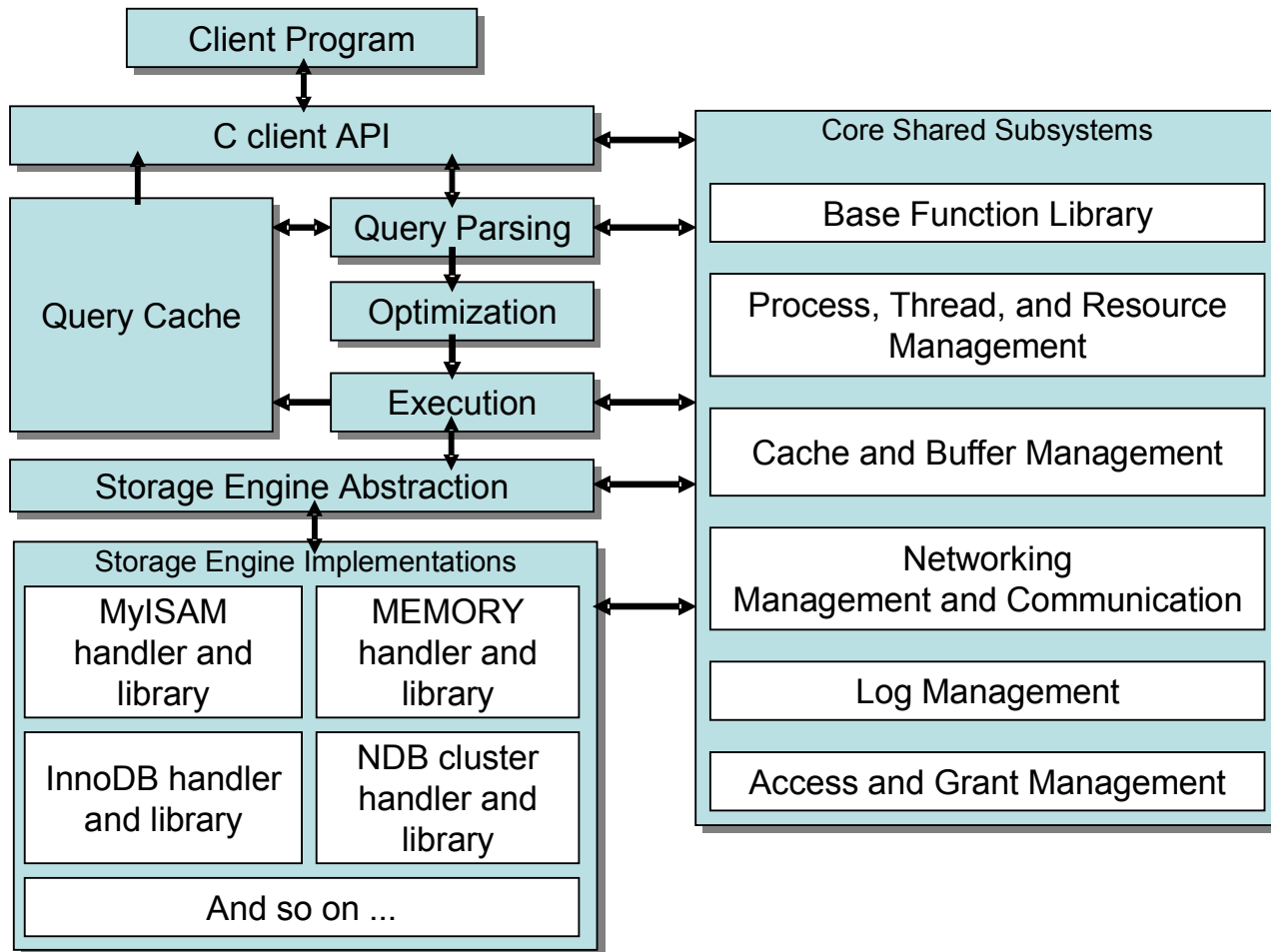
Advanced

Practical Performance Tips & Tricks

Advanced Agenda (15 mins)

- Understanding MySQL Architecture
 - ♦ Query Path
 - ♦ MyISAM Layout
 - ♦ Innodb Layout
- Other Tools
- Design Analysis
- Data Analysis

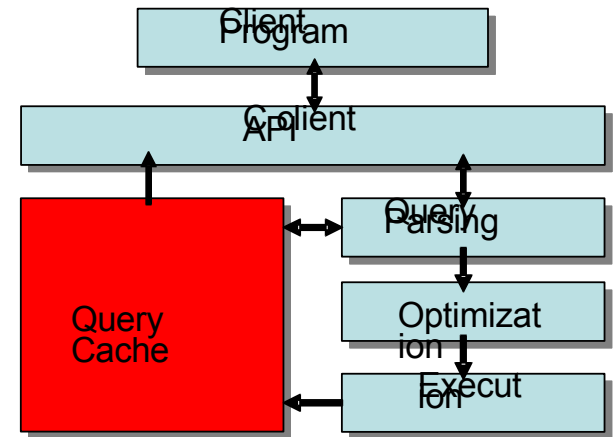
MySQL Architecture (High Level)



Practical Performance Tips & Tricks

Query Cache

- ◆ Caches SQL query result
 - ◆ queries must be absolutely the same
- ◆ Caches in MySQL server memory
- ◆ Fully transparent for application
- ◆ No invalidation control
 - ◆ query invalidated when involved table is updated
- ◆ Does not work with prepared statements! (before 5.1.17)
- ◆ Can be ON, OFF, or DEMAND



TIP: Convert non-deterministic function to deterministic (e.g. CURDATE() to actual date value)

Query Cache

- ♦ Query Cache adds an overhead to all queries
- ♦ When is it good? when is it bad?
- ♦ What sizes are appropriate

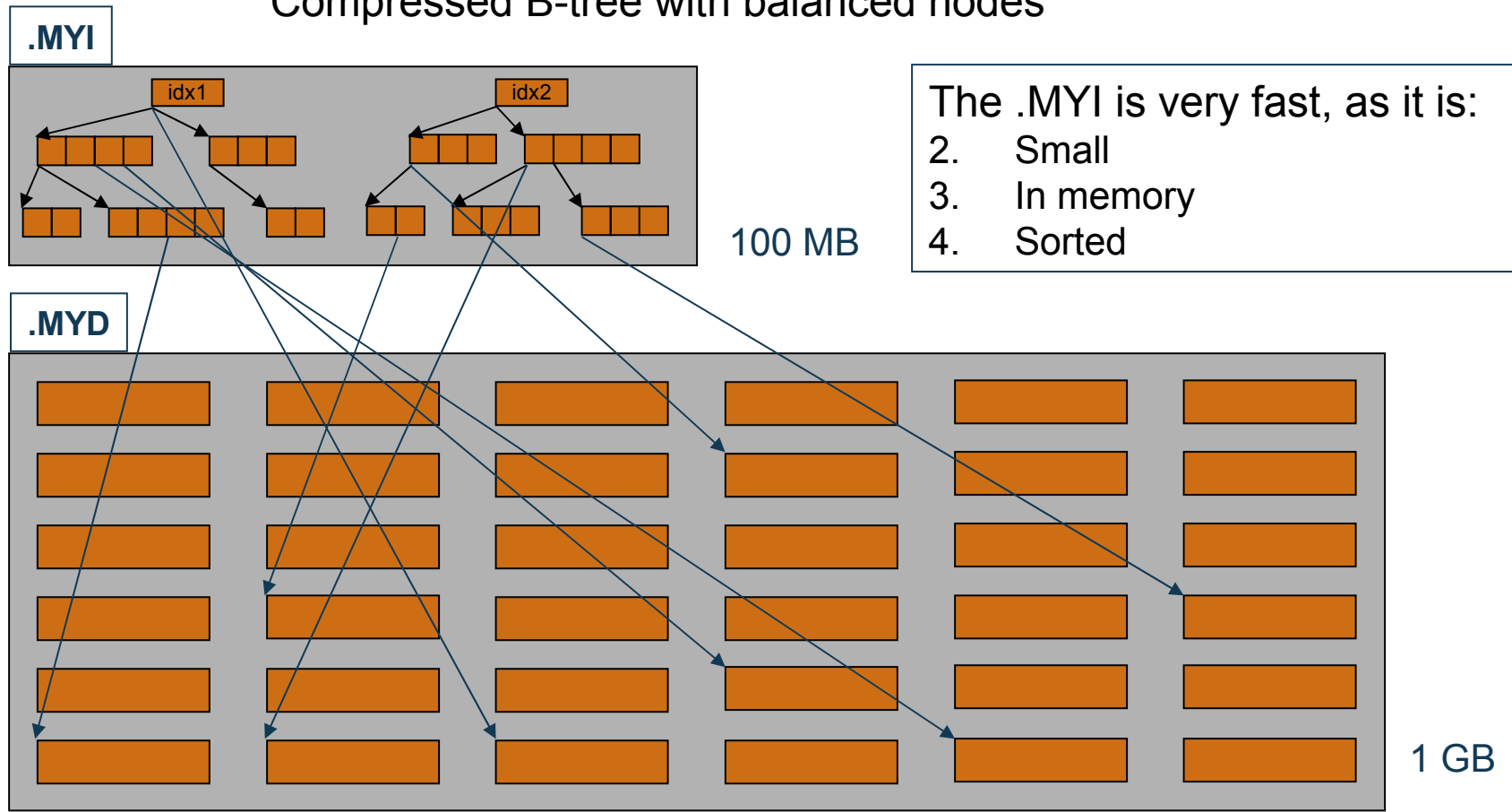
```
query_cache_type = 0|1|2  
query_cache_size = 32M  
query_cache_limit = 1M  
mysql> SHOW STATUS LIKE 'Qcache%';
```

In some environments, OFF is best

<http://dev.mysql.com/doc/refman/5.0/en/query-cache.html>

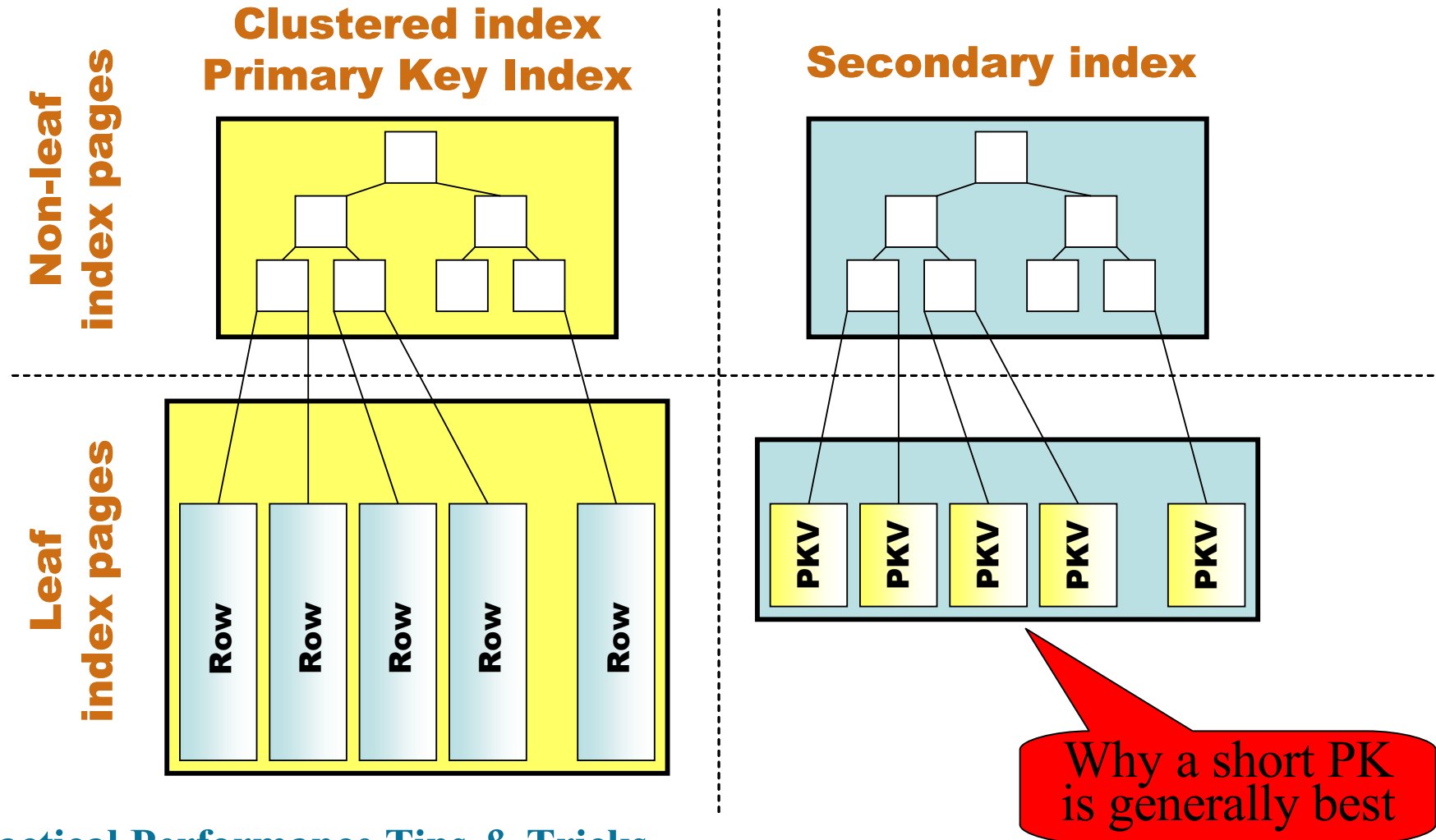
MyISAM Index Structure

Compressed B-tree with balanced nodes



Practical Performance Tips & Tricks

Innodb Index Structure



Practical Performance Tips & Tricks

Memory Usage

- ♦ MyISAM
 - ♦ Indexes Only (`key_buffer_size`)
 - ♦ File System Cache
- ♦ InnoDB
 - ♦ Data + Indexes (`innodb_buffer_pool_size`)

Other Tools

- SHOW PROFILE (5.0.37+ Community Edition)
 - ♦ Example of Community Contribution

Show Profile

```
mysql> show profile SOURCE, MEMORY for query 4;
```

Status	Duration	Source_function	Source_file	Source_line
Opening tables	0.00013200	open_tables	sql_base.cc	2106
System lock	0.00001800	mysql_lock_tables	lock.cc	153
Table lock	0.00000600	mysql_lock_tables	lock.cc	162
init	0.00001300	mysql_select	sql_select.cc	2073
optimizing	0.00004800	optimize	sql_select.cc	617
statistics	0.00002500	optimize	sql_select.cc	773
preparing	0.00005200	optimize	sql_select.cc	783
executing	0.00002200	exec	sql_select.cc	1407
Sending data	0.00000500	exec	sql_select.cc	1925
end	0.00786600	mysql_select	sql_select.cc	2118
query end	0.00001400	mysql_execute_command	sql_parse.cc	5085
freeing items	0.00000700	mysql_parse	sql_parse.cc	5973
closing tables	0.00001900	dispatch_command	sql_parse.cc	2120
logging slow query	0.00001000	log_slow_statement	sql_parse.cc	2178
cleaning up	0.00000500	dispatch_command	sql_parse.cc	2143

```
15 rows in set (0.01 sec)
```

Practical Performance Tips & Tricks

Show Profile

```
mysql> show profile source for query 14;
```

Status	Duration	Source_function	Source_file	Source_line
Opening tables	0.00006025	open_tables	sql_base.cc	2100
System lock	0.00004875	mysql_lock_tables	lock.cc	153
Table lock	0.00000400	mysql_lock_tables	lock.cc	162
init	0.00001600	mysql_select	sql_select.cc	2073
optimizing	0.00005675	optimize	sql_select.cc	617
statistics	0.00001250	optimize	sql_select.cc	773
preparing	0.00005175	optimize	sql_select.cc	783
Creating tmp table	0.00001275	optimize	sql_select.cc	1200
executing	0.00006025	exec	sql_select.cc	1407
Copying to tmp table	0.00000400	exec	sql_select.cc	1547
converting HEAP to MyISAM	0.04820900	create_myisam_from_heap	sql_select.cc	9914
Copying to tmp table on disk	0.04049075	create_myisam_from_heap	sql_select.cc	9968
Sending data	1.29302000	exec	sql_select.cc	1925
end	0.09398425	mysql_select	sql_select.cc	2118
removing tmp table	0.00004975	free_tmp_table	sql_select.cc	9856
end	0.00089125	free_tmp_table	sql_select.cc	9884
query end	0.00001850	mysql_execute_command	sql_parse.cc	5085
freeing items	0.00000825	mysql_parse	sql_parse.cc	5973
closing tables	0.00003425	dispatch_command	sql_parse.cc	2120
logging slow query	0.00001325	log_slow_statement	sql_parse.cc	2178
cleaning up	0.00000675	dispatch_command	sql_parse.cc	2143

```
21 rows in set (0.00 sec)
```

Practical Performance Tips & Tricks

Show Profile

- Set appropriate tmp_table_size

mysql> show profile source for query 14;			mysql> show profile source for query 18;		
Status	Duration	Source	Status	Duration	Source
Opening tables	0.00006025	open	Opening tables	0.00006050	open_table
System lock	0.00004875	mysq	System lock	0.00001250	mysql_lock
Table lock	0.00000400	mysq	Table lock	0.00000400	mysql_lock
init	0.00001600	mysq	init	0.00000775	mysql_sele
optimizing	0.00005675	opti	optimizing	0.00005475	optimize
statistics	0.00001250	opti	statistics	0.00001225	optimize
preparing	0.00005175	opti	preparing	0.00005075	optimize
Creating tmp table	0.00001275	opti	Creating tmp table	0.00001350	optimize
executing	0.00006025	exec	executing	0.00006125	exec
Copying to tmp table	0.00000400	exec	Copying to tmp table	0.00000375	exec
converting HEAP to MyISAM	0.04820900	crea	Sending data	0.29110925	exec
Copying to tmp table on disk	0.04049075	crea	end	0.08023800	mysql_sele
Sending data	1.29302000	exec	removing tmp table	0.00001525	free_tmp_t
end	0.09398425	mysq	end	0.05971400	free_tmp_t
removing tmp table	0.00004975	free	query end	0.00001925	mysql_exc
end	0.000089125	free	freeing items	0.00000425	mysql_pars
query end	0.00001850	mysq	closing tables	0.00004625	dispatch_c
freeing items	0.00000825	mysq	logging slow query	0.00000800	log_slow_s
closing tables	0.00003425	disp	cleaning up	0.00000300	dispatch_c
logging slow query	0.00001325	log			
cleaning up	0.00000675	disp			
21 rows in set (0.00 sec)			19 rows in set (0.00 sec)		

Design & Data Analysis

Practical Performance Tips & Tricks

Design & Data Analysis

- ♦ Optimal Indexes
- ♦ Optimal Data Types
- ♦ Data Analysis Tools

Indexes

- ♦ Removing duplicate indexes
- ♦ Removing unused indexes
- ♦ Partial Indexes
- ♦ Covering Indexes
- ♦ Index for filesort

Generally MySQL will only use one index per table

Partial Index

- ♦ MySQL allows for a Partial Index

```
CREATE INDEX i1 ON t1 last_name(10);
```

- ♦ No Function based indexes

Covering Index

♦ Covering SELECT,WHERE,ORDER BY columns

```
mysql> explain select last_name from actor where last_name='kilmer';
```

id	select_type	table	type	possible_keys	key	key_len
1	SIMPLE	actor	ref	idx_actor_last_name	idx_actor_last_name	137
const 5 Using where; Using index						

```
mysql> explain select last_name,first_name from actor where last_name='kilmer';
```

id	select_type	table	type	possible_keys	key	key_len
1	SIMPLE	actor	ref	idx_actor_last_name	idx_actor_last_name	137
const 5 Using where						

```
mysql> alter table actor add index (last_name,first_name);
```

```
Query OK, 200 rows affected (0.06 sec)
Records: 200 Duplicates: 0 Warnings: 0
```

```
mysql> explain select last_name,first_name from actor where last_name='kilmer';
```

id	select_type	table	type	possible_keys	key	key_len
1	SIMPLE	actor	ref	idx_actor_last_name,last_name	last_name	137
const 5 Using where; Using index						

Practical Performance Tips & Tricks

'filesort' Index

- ◆ When Ordering without Index, filesort is used
- ◆ Filesort is temporary table, that could be flushed to disk

```
mysql> explain select * from actor order by first_name;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | actor | ALL | NULL | NULL | NULL | NULL | 200 | Using filesort |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> alter table actor add index (first_name);
Query OK, 200 rows affected (0.10 sec)
Records: 200 Duplicates: 0 Warnings: 0
```

```
mysql> explain select * from actor order by first_name;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | actor | index | NULL | first_name | 137 | NULL | 200 | |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

Practical Performance Tips & Tricks

Data Analysis Tool

```
mysql> SELECT * FROM table PROCEDURE ANALYSE (5,2000) \G
```

```
***** 2. row *****
      Field name: sakila.actor.first_name
      Min_value: ADAM
      Max_value: ZERO
      Min_length: 2
      Max_length: 11
      Empties_or_zeros: 0
      Nulls: 0
      Avg_value_or_avg_length: 5.3050
      Std: NULL
      Optimal_fieldtype: VARCHAR(11) NOT NULL
***** 3. row *****
      Field name: sakila.actor.last_name
      Min_value: AKROYD
      Max_value: ZELLWEGER
      Min_length: 3
      Max_length: 12
      Empties_or_zeros: 0
      Nulls: 0
      Avg_value_or_avg_length: 6.2300
      Std: NULL
      Optimal_fieldtype: VARCHAR(12) NOT NULL
```

```
mysql> desc actor;
```

Field	Type	Null	Key	Default	Extra
actor_id	smallint(5) unsigned	NO	PRI	NULL	auto_increment
first_name	varchar(45)	NO			
last_name	varchar(45)	NO	MUL		
last_update	timestamp	NO		CURRENT_TIMESTAMP	

4 rows in set (0.01 sec)

Practical Performance Tips & Tricks

Data Analysis Tool

Most common easy improvements

- ♦ Use UNSIGNED when data reflects this
- ♦ Use NOT NULL when possible
- ♦ Use ENUM for small static string ranges
- ♦ Unless you need more than 4 billion rows, BIGINT is simply a waste, use an appropriate 'integer' data type especially for Indexes
- ♦ When value is 0, 1 or single digits, use TINYINT(1) rather than INT(1)

Data Types - Number

- ♦ Integer
 - ♦ TINYINT 1 byte 0 - 255 unsigned
 - ♦ SMALLINT 2 bytes 0 - 65K unsigned
 - ♦ MEDIUMINT 3 bytes 0 - 16M unsigned
 - ♦ INT 4 bytes 0 - 4B unsigned
 - ♦ BIGINT 8 bytes
- ♦ Floating Point
 - ♦ FLOAT 4-8 bytes
 - ♦ DOUBLE 8 bytes
- ♦ Fixed Point
 - ♦ NUMBER(n,m) 8+ bytes

Example Index

Table 'users'

- ♦ marketing int(1), advertising int(1), sales int(1), other int(1)
- ♦ All Nullable
- ♦ Index on (marketing, advertising,sales,other)
- ♦ Index is **20 bytes per row**
- ♦ INT(1) is not 1 digit
- ♦ Optimizations
 - ♦ All columns made tinyint(1)
 - ♦ All columns converted to NOT NULL (99% values not null)
 - ♦ Index is now **4 bytes per row**, more rows per data page, less disk, less memory
- ♦ Greater question?
 - ♦ Is this index even the best choice due to cardinality

Memory Analysis

- ♦ MySQL Uses Memory:
 - ♦ Startup
 - ♦ Per Connection
 - ♦ Memory Tables
 - ♦ Temporary Tables

Memory Analysis

MyISAM Environment

Pre-allocated on startup

$\text{initial} = \text{key_buffer_size} + \text{query_cache_size}$

Per Connection (maximum)

$\text{per_connection} = (\text{sort_buffer_size} + \text{read_rnd_buffer_size} + \text{join_buffer_size} + \text{read_buffer_size} + \text{thead_stack} + \text{binlog_cache_size})$

Total: $\text{initial} + \text{max_connections} * \text{per_connection}$

Conservative: $\text{initial} + ((\text{max_connections} * \text{per_connection})/3)$

Practical Performance Tips & Tricks

Memory Analysis

InnoDB Environment

Pre-allocated on startup

$\text{initial} = \text{innodb_buffer_pool_size} + \text{query_cache_size}$

Per Connection (maximum)

$\text{per_connection} = (\text{sort_buffer_size} + \text{join_buffer_size} + \text{thead_stack} + \text{binlog_cache_size})$

Total: $\text{initial} + \text{max_connections} * \text{per_connection}$

Conservative: $\text{initial} + ((\text{max_connections} * \text{per_connection})/3)$

Practical Performance Tips & Tricks

Memory Analysis

Memory Tables

- ♦ No total limit
- ♦ Per table limit is max_heap_table_size
- ♦ Calculate and monitor present usage

```
select table_schema,count(*) as table_count,  
       sum(data_length+index_length)/1024/1024 as size_MB  
from   information_schema.tables  
where  engine='MEMORY'  
group by table_schema;
```

Memory Analysis

Temporary Tables

- ♦ Need to determine portion of connections creating tables
- ♦ Determine portion of tmp_table_size used

Memory Limitations

- ♦ No means to limit total connection usage
- ♦ No means to limit total MEMORY/tmp table usage
- ♦ No means to PIN data or Indexes
- ♦ MyISAM can allocate dedicated index caches (to pin)
- ♦ TEXT/BLOB for temporary tables cause disk write

Connection Buffers Memory Usage

- ♦ e.g. `sort_buffer`
- ♦ Allocated for all columns retrieved
 - ♦ (why '*' is not optimal)

Write optimal SELECT Statements

- ♦ `varchar(45)` for `utf_8` is 45×3 bytes in length no what is stored on disk (e.g. which may be 11 bytes average)

Use Optimal Varchar Data Types

MySQL Proxy

About MySQL Proxy

- ♦ New Alpha product (released June 2007)
- ♦ Common uses
 - ♦ load balancing
 - ♦ failover
 - ♦ query analysis
 - ♦ query filtering
 - ♦ query modification
 - ♦ lots' of opportunity

http://forge.mysql.com/wiki/MySQL_Proxy

proxy - a person who speaks or acts on one's behalf