The Guide To Understanding mysqlreport

This guide to understanding **mysqlreport** explains everything that mysqlreport can report. It also teaches how to interpret and understand all the values in context so that after reading a mysqlreport report ("a report") the fundamental question that mysqlreport was created to answer can be answered: How well is the MySQL server running?

The current version of mysqlreport automatically generates a complete report that covers practically every applicable MySQL status value. (This is different from all previous versions which required the --all command line option to generate a complete report.) The complete report has 14 report sections for a total of 121 lines. Depending on the MySQL server configuration, some of the report sections may not be generated. For example, if query caching is turned off, the forth report section, Query Cache, will not be generated. Therefore, the report can vary in length.

To facilitate better understanding and insight, this guide is written as a walk-through interpretation of a complete report. The guide begins with the very first line and examines and interprets every report section and line to the end.

The complete report that this guide will interpret is given below. Line numbers are added to make examining the report easier. Most line numbers are links to the explanation of that line.

```
 1 MySQL 5.0.3               uptime 0 0:34:26       Fri Sep  1 19:46:02 2006
 2
 3 __ Key _________________________________________________________________
 4 Buffer used   380.00k of 512.00M  %Used:   0.07
 5   Current      59.32M            %Usage:  11.59
 6 Write hit      97.04%
 7 Read hit       99.58%
 8
 9 __ Questions ___________________________________________________________
10 Total           98.06k   47.46/s
11   DMS           81.23k   39.32/s  %Total:  82.84
12   QC Hits       16.58k    8.02/s           16.91
13   COM_QUIT         200    0.10/s            0.20
14   Com_             131    0.06/s            0.13
15   -Unknown          82    0.04/s            0.08
16 Slow 5 s             0    0.00/s            0.00  %DMS:   0.00  Log:  ON
17 DMS             81.23k   39.32/s           82.84
18   SELECT        64.44k   31.19/s           65.72         79.33
19   INSERT        16.75k    8.11/s           17.08         20.61
20   UPDATE            41    0.02/s            0.04          0.05
21   REPLACE            0    0.00/s            0.00          0.00
22   DELETE             0    0.00/s            0.00          0.00
23 Com_               131    0.06/s            0.13
24   change_db        119    0.06/s            0.12
25   show_fields        9    0.00/s            0.01
26   show_status        2    0.00/s            0.00
27
28 __ SELECT and Sort ____________________________________________________
29 Scan                38    0.02/s %SELECT:   0.06
30 Range               14    0.01/s            0.02
31 Full join            3    0.00/s            0.00
32 Range check          0    0.00/s            0.00
33 Full rng join        0    0.00/s            0.00
34 Sort scan           14    0.01/s
35 Sort range          26    0.01/s
36 Sort mrg pass        0    0.00/s
37
38 __ Query Cache ________________________________________________________
39 Memory usage    17.81M of  32.00M  %Used:  55.66
40 Block Fragmnt   13.05%
41 Hits            16.58k    8.02/s
42 Inserts         48.50k   23.48/s
43 Prunes          33.46k   16.20/s
44 Insrt:Prune     1.45:1    7.28/s
45 Hit:Insert      0.34:1
46
47 __ Table Locks ________________________________________________________
48 Waited           1.01k    0.49/s  %Total:   1.24
49 Immediate       80.04k   38.74/s
50
51 __ Tables _____________________________________________________________
52 Open              107 of 1024    %Cache:  10.45
53 Opened            118    0.06/s
54
55 __ Connections ________________________________________________________
56 Max used           77 of  600      %Max:  12.83
57 Total             202    0.10/s
58
59 __ Created Temp _______________________________________________________
60 Disk table         10    0.00/s
61 Table              26    0.01/s    Size:  4.00M
62 File                3    0.00/s
63
64 __ Threads ____________________________________________________________
65 Running            55 of   77
66 Cache               0             %Hit:    0.5
67 Created           201    0.10/s
68 Slow                0    0.00/s
69
70 __ Aborted ____________________________________________________________
```

```
71 Clients              0     0.00/s
72 Connects             8     0.00/s
73
74 __ Bytes ____________________________________________________________
75 Sent            38.46M  18.62k/s
76 Received         7.98M   3.86k/s
77
78 __ InnoDB Buffer Pool _______________________________________________
79 Usage            3.95M of    7.00M  %Used:  56.47
80 Read hit        99.99%
81 Pages
82   Free             195            %Total:  43.53
83   Data             249                     55.58 %Drty:   0.00
84   Misc               4                      0.89
85   Latched            0                      0.00
86 Reads          574.56k      0.6/s
87   From file        176      0.0/s            0.03
88   Ahead Rnd          4      0.0/s
89   Ahead Sql          2      0.0/s
90 Writes         160.82k      0.2/s
91 Flushes          1.04k      0.0/s
92 Wait Free            0        0/s
93
94 __ InnoDB Lock ______________________________________________________
95 Waits                0        0/s
96 Current              0
97 Time acquiring
98   Total              0 ms
99   Average            0 ms
100  Max                0 ms
101
102 __ InnoDB Data, Pages, Rows ________________________________________
103 Data
104   Reads            225      0.0/s
105   Writes           799      0.0/s
106   fsync            541      0.0/s
107   Pending
108     Reads            0
109     Writes           0
110     fsync            0
111
112 Pages
113   Created           23      0.0/s
114   Read             226      0.0/s
115   Written         1.04k      0.0/s
116
117 Rows
118   Deleted       25.04k      0.0/s
119   Inserted      25.04k      0.0/s
120   Read          81.91k      0.1/s
121   Updated            0        0/s
```

Report Header: Line 1

The first line of a report, the report header, has three pieces of information: the MySQL server version, the MySQL server uptime, and the server's current date and time.

The MySQL server version indicates what features the MySQL server does or does not have. The MySQL server uptime indicates how representative the report values are. The MySQL server uptime is important for assessing the report because the report values will be skewed and misleading if the MySQL server has not been running for at least a few hours. However, even a few hours may not be enough if, for example, the MySQL server has only been running for six hours in the middle of the night with almost no usage. Ideally, the MySQL server should be up for at least one day before using mysqlreport. The report values will better represent the MySQL server the longer that the MySQL server is up.

In this example, the MySQL server has only been up for 34 minutes. Consequently, the report is not very representative.

Key Report: Lines 3 - 7

The first major report section of a mysqlreport report is the Key report because keys (indexes) are the most important part of a MySQL server. Although the report cannot indicate if the MySQL server is well indexed or not, it can generally indicate how well the shared key buffer is being utilized. This report section only applies to the default shared key buffer for MyISAM tables; it currently does not look at other key buffers (created by the administrator, like hot and cold key buffers).

Buffer used: Line 4

The first question to ask of a MySQL server is: how much of the key buffer is being used? If it is not being used a lot, that is okay because MySQL only allocates system RAM for the key buffer when needed. That means that if the key buffer size is set to 512M, MySQL does not automatically allocate 512M of system RAM when it starts. MySQL allocates up to 512M of system RAM when it needs to.

The fourth line, Buffer used, is supposed to indicate the maximum amount of key buffer MySQL has ever used at once. In actuality, MySQL may currently being using less or, strangely, more. MySQL calls this a "high water mark." This line is usually indicative of whether or not the key_buffer_size system variable is sufficiently large. If this line indicates that MySQL has used upwards of 80% to 90% of the key buffer, then key_buffer_size should be increased. Note, however, that this line will probably never indicate above 95% used because, as the MySQL documentation states, some of the shared key buffer is used for internal data structures which mysqlreport cannot account for. Therefore, 95% used is practically 100% used.

In this example, the MySQL server has used 380k of 512M, or 0.07%, so the key buffer is plenty large. However, the next line indicates something different.

Current:* *Line 5

This line only appears for MySQL servers version 4.1.2 and newer because the Key_blocks_unused status value was added in version 4.1.2. This line indicates how much key buffer MySQL is actually using right now. If the previous line is truly a high water mark, then this line should always be less than or equal to it, but this is not always the case. Whether this is a bug in MySQL or not is unknown. Regardless, this line in combination with the preceding line gives a good indication of whether or not key_buffer_size is set sufficiently large.

In this example, the MySQL server is using about 60M of the key buffer (12%), which is good because it is nowhere near full capacity.

Write hit:* *Line 6

Indexes (keys) are inherently RAM-based. Their usefulness is due in part to the fact that they exist in RAM which is very quick to access instead of existing only on a hard disk which is very slow to access. However, MySQL inevitably must write and read indexes to and from a hard disk at times.

This line, Write hit, indicates the effectiveness of key writes. (Technically, it is the ratio of key writes to hard disk to key writes to RAM expressed as a percentage.) There is no standard value for key write hit. Key write hit will depend on what kind queries the MySQL server primarily executes. If MySQL primarily executes updates, inserts, etc., then the key write hit may be near 0% and this is acceptable. If MySQL primarily executes selects, then the key write hit may be 90% or more and this is acceptable too. However, a negative key write hit (i.e., less than zero percent) indicates that MySQL is more often writing keys to hard disk than RAM which is usually slow, undesirable, and unacceptable.

To best interpret the key write hit effectiveness, it is necessary to know how the MySQL server is primarily used. The **DMS sub-report** can help determine this.

Read hit:* *Line 7

More important than the key write hit is the key read hit. This line indicates the effectiveness of key reads. (Technically, it is the ratio of key reads from hard disk to key reads from RAM.) Key read hit should be no less than 99%. A lower percentage may indicate a problem. A low key hit percentage is usually caused by the key buffer being too small (indicated in the Key Report section above) which prevents MySQL from loading more indexes into RAM. When this happens, MySQL must revert to reading indexes from the hard disk which is terribly slow and completely negates the point of indexes.

It is common, however, for this value to be less than 99% within the first hour or two of starting (or restarting) MySQL. After an hour or two it should definitely be at least 99%.

Questions Report: Lines 9 - 26

The second major report section, Questions, is the second most important because it shows a lot about what MySQL is busy doing and how busy it is doing all that it is doing. Questions includes SQL queries and MySQL protocol communications. A common concern is how many queries per second the MySQL server is executing, but this metric is actually very arbitrary when considered in the larger context. The larger context is all the other questions that MySQL is handling. This report provides the larger context.

Total:* *Line 10

The first line is simply the total number of all questions that MySQL has answered† (first column) and the rate of those answers over the MySQL server uptime (second column). The rate is what is commonly referred to when people make statements such as, "My MySQL server averages one hundred queries a second." However, the real question is: of those one hundred, how many are really accomplishing something? mysqlreport answers this question in the following lines.

(† A clarification on terms: questions are answered and queries are executed. mysqlreport makes the distinction between questions and queries, especially in the Questions Report. Questions are every and any kind of request made to the MySQL server. This includes SQL queries but also MySQL-specific commands and protocol communications. Queries are only SQL queries: SELECT, UPDATE, etc.)

Distribution of Total Queries (DTQ):* *Lines 11 - 15

All questions can largely be divided into five categories: **Data Manipulation Statements** (DMS), query cache hits (QC Hits), COM_QUIT, all other Com_ commands, and Unknown. These five categories are indicated in the five lines 11 through 15 but their order is dynamic: mysqlreport sorts them in descending order based on total number (first column).

This sub-report quickly indicates what MySQL is most busy doing. Ideally, MySQL should be most busy with DMS or QC Hits because these are the categories of questions that are really accomplishing something. COM_QUIT, Com_, and Unknown are necessary but should play only a minor role.

Before explaining each category further, it will be helpful to mention that the third column for this and other sub-reports in the Questions report shows the percentage of that line's total value to all questions (Total, line 10).

In this example, DMS questions account for 82.84% of all questions that the MySQL server has answered which is a really good percentage.

Data manipulation statements include: SELECT, INSERT, REPLACE, UPDATE, and DELETE. (Technically, there are others but mysqlreport uses only these.) Basically, DMS is what one thinks of when thinking of MySQL doing something useful. Hence, DMS should be what MySQL is most busy doing. This category is expanded in more detail in the DMS sub-report later, lines 17 through 22.

QC Hits is the number of queries that MySQL has executed by retrieving the result set from the query cache instead of actually executing the query. Having a high percentage of QC Hits is coveted because returning result sets from the QC is very fast. However, it can be difficult to achieve a very effective QC cache for reasons explained in the Insrt:Prune and Hit:Insert Ratios section of the Query Cache Report.

In this example, QC Hits account for 16.91% of all questions which is pretty good. However, don't be mislead by this: the Query Cache Report (lines 38 through 45) can tell a very different story. Whereas QC Hits seem pretty good here, this server's query cache is actually not that spectacular as will be seen later.

COM_QUIT is a category which is written about in the article **COM_QUIT and Questions**. It is an unimportant

category which can be ignored. It is included in mysqlreport for completeness.

Com_ represents all the various commands that MySQL handles, usually protocol related. Ideally, this category should be low because when it is high it is like MySQL is spinning its wheels really fast but going nowhere. A high value for this category can indicate some weird problems which are discussed later in the Com_ sub-report (lines 23 through 26 usually).

Unknown is an inferred category. Ideally, the sum total of the preceding four categories should equal total questions, but they usually do not. This is because there are a few questions that MySQL handles and increments the total questions counter for but does not otherwise maintain a separate status value for.

This line is dynamic in that it can read "+Unknown" or "-Unknown." +Unknown means there are more total questions than mysqlreport can account for. -Unknown means mysqlreport counted less questions than total questions.

This category can vary greatly. On some servers it is near the top, but on most it is at the very bottom. It is better for it to be at the very bottom. Eventually, the nature of these unknowns will be discovered and mysqlreport will account for them correctly.

Slow: Line 16

Line 16 is very important: it indicates the number of slow queries that MySQL has executed. What constitutes "slow" is set by the **system variable** long_query_time which is 10 (seconds) by default. Since many people consider 10 seconds to be an eternity in database time, long_query_time is best set to 1 or less for newer versions of MySQL that support microsecond resolution.

This value, long_query_time, is the number that appears just after Slow. As of **mysqlreport** v3.5, the **resolution** of this value is given: s (seconds), ms (milliseconds), or µ (microseconds). In some cases, the resolution may not be shown due to the width limit of this field in the report which is 8 characters. For example, a long_query_time value of '999.999 ms' is truncated to '999.999 ', or '10.000100 s' is truncated to '10.0001 '.

Ideally, there should be zero slow queries, but usually there are a few. Generally, Slow as a percentage of total questions (third column) should be 0.05 or less. There can be a lot of slow queries (first column), but it is the percentage of all them compared to total that indicates a problem. This line also adds a fourth column: percentage of DMS questions. For Slow, zero is best, but this column is more useful in the DMS sub-report.

The last column, Log, indicates if slow query logging is turned ON or OFF (set by the log_slow_queries system variable). Slow query logging should always be ON.

DMS: Lines 17 - 22

The DMS sub-report, like the DTQ sub-report, is sorted in descending order of value (first column). Its 6 lines, 17 through 22, represent the data manipulation statements mentioned earlier (SELECT, INSERT, etc.). The first line (17) is the total of all these again (identical to line 11 in the DTQ sub-report).

This sub-report indicates what "kind" of MySQL server this is: is it SELECT heavy, or INSERT heavy, etc. MySQL servers tend to be SELECT heavy. Knowing what kind of MySQL server a server is helps orient one's thoughts and understanding about other the other values. For example, an INSERT heavy server should have a write ratio near 1.0. It will probably have high values regarding table locks. It would also be a candidate for InnoDB tables. A SELECT heavy server had better have a read ratio of zero and a very low table lock values. It maybe be using query caching. It will probably use MyISAM.

In this example, the server is SELECT heavy: 65.72% of all questions are SELECTs (third column), and 79.33% of all DMS questions are SELECTs (fourth column). Clearly, this server is oriented towards SELECT statements. Knowing that shapes how one approaches all aspects of optimization.

Com_: Lines 23 - (26)

The Com_ sub-report like other sub-report so-far is sorted. The contents of this sub-report vary from server to server because each line (default 3; more can be specified like --com 10) represents some Com_ status value which, in turn, usually represent some COM_ command in the **MySQL protocol**. Most of the names are intuitive, like Com_change_db.

This sub-report matters when Com_ in the DTQ sub-report is near the top because it indicates MySQL is busy doing "program things" instead of answering SQL queries. As an example, there was a server that had a very high number of Com_rollback. A rollback occurs when a transaction fails and this is usually not a good thing. The server was failing nearly every transaction so clearly something was wrong. Without mysqlreport, the DTQ sub-report, and this sub-report it was practically impossible to otherwise tell that the server had any problem.

For most servers, the Com_ sub-report indicates nothing weird, but it is good to check it from time to time.

SELECT and Sort Report: Lines 28 - 36

The SELECT and Sort Report details the various Select_ status value which are described in the article **MySQL Select and Sort Status Variables**.

The most important lines are 29 and 31: Scan and Full join. Scan indicates how many SELECT statements resulted in a full table scan which is a slow process. Full join is like Scan except that it applies to tables being joined in a multi-table query. Such tables are joined by process of a full table scan, but in the context of a join, a table scan is even slower. Therefore, these two values should be as low as possible, but there is no real standard for "low" here. Some servers which are running really well have a relatively high percentage of Scan to all SELECT statements (third column).

Query Cache Report: Lines 38 - 45

The Query Cache report only appears if, one, the MySQL server version supports query cache and, two, query cache is enabled.

Memory usage: Line 39

This first line of this report indicates how much of the query cache memory is being used. If it is at

max capacity, this will probably also be reflected in the Prunes value below since queries in the QC are pruned when memory is low.

Block Fragmnt: Line 40

Line 40, Block Fragmnt (Fragmentation), indicates a condition particular to the way the query cache functions. Quoting from the MySQL manual section **5.14.3. Query Cache Configuration**:

> The default value of query_cache_min_res_unit is 4KB. This should be adequate for most cases. ... If you have a lot of queries with small results, the default block size may lead to memory fragmentation, as indicated by a large number of free blocks. Fragmentation can force the query cache to prune (delete) queries from the cache due to lack of memory. In this case, you should decrease the value of query_cache_min_res_unit. The number of free blocks and queries removed due to pruning are given by the values of the Qcache_free_blocks and Qcache_lowmem_prunes status variables.

This value is a percentage of free QC blocks to total blocks. The higher the percentage, the more the QC memory is fragmented. 10% to 20% is about average.

In this example, block fragmentation is 13.05%. This is acceptable, but it might be helpful to play around with query_cache_min_res_unit to see if it could be lowered.

Hits, Inserts, Prunes: Lines 41 - 43

Query cache Hits, Inserts, and Prunes are indicated on lines 41, 42, and 43. Hits is the most important because it indicates how many SELECT statements were served from the cache, so the more the better. Inserts and Prunes are better understood in terms of the ratio on line 44. Although, as mentioned earlier, a high rate of Prunes can be indicative of the QC size being too small, but not always.

In this example, only 55% of the QC is in use, with relatively low fragmentation, yet prunes are pretty high; prunes are occurring at the rate of 16/s, double the rate of QC hits. In a sense, this server's QC is like an apple tree where the limbs are being cut off faster than the apples are being picked.

Insrt:Prune and Hit:Insert Ratios: Lines 44 - 45

The QC Insert:Prune ratio on line 44 is an indicator of QC volatility. In a highly stable QC, more queries will be inserted than are pruned. In a volatile QC, this ratio will be one-to-one or heavy on the prune side, indicating a kind of evacuation of queries from the QC. A stable QC is desirable because a stable QC implies that the cached queries are being used often. A volatile QC can indicate two things: one, the QC size is too small so MySQL has to keep pruning and inserting queries, or two, MySQL is trying to cache everything to a self-defeating end.

In the first case, simply increasing the QC size may help. This type of volatility may be further indicated by high block fragmentation and QC memory usage.

The second type of volatility is more common because MySQL does try to cache nearly everything it can when the QC is enabled with the default type 1. Type 1 means (quoting the MySQL manual): "Cache all query results except for those that begin with SELECT SQL_NO_CACHE." It seems, however, that SQL_NO_CACHE is rarely used. A better way to enable the QC is with type 2 "DEMAND": "Cache results only for queries that begin with SELECT SQL_CACHE." Demand caching requires more work for developers because they have to explicitly add SQL_CACHE to the queries that they want MySQL to cache, but the advantage is that they probably know what queries are good, stable cache candidates.

The other ratio is Hit:Insert. This ratio indicates QC effectiveness. Ideally, the MySQL server should insert a bunch of stable queries into the QC, then get a lot more hits on them. Therefore, this ratio should be heavy on the hit side if the QC is effective. If it is heavy on the insert side, then the QC is not really helping much and it is probably too volatile. Consider a Hit:Insert ratio of 1:1. This practically means that a cached result is only used once before it is replaced. This completely defeats the idea of a query cache. A worse ratio, like 0.34:1, indicates that some results are not even hit before they are pruned or replaced.

In this example, as mentioned previously in the DTQ sub-report, even though QC Hits account for a good percentage of total questions, the actual QC effectiveness is really low as indicated by the Hit:Insert ratio being terribly Insert heavy. This MySQL server would benefit from demand caching since QC memory usage and fragmentation are not bad. Chances are MySQL is just defeating itself trying to cache everything.

Table Locks Report: Lines 47 - 49

The Table Locks report consists of two lines: the first, Waited, shows the number of table locks that MySQL had to wait to obtain, and the second, Immediate, shows the number of table locks MySQL obtained immediately. Waiting is almost always a bad thing in database terms, therefore, table locks waited should be as low as possible. What is most indicative of table locking problems is the third column of table locks waited: %Total of all table locks. The percentage of table locks that had to wait should be 10% or less. Higher percentage can indicate poor table/query indexing or slow queries.

Tables Report: Lines 51 - 53

The Tables report is also two lines: the first, Open, shows how many tables are open right now (first column), of how many total possible (table cache; second column), and the percentage of table cache usage (third column). The second line, Opened, shows the total number of tables MySQL has ever opened and this value over the MySQL server uptime (second column).

Two things are important here: first is the table cache usage. It is not bad to have 100% table cache usage, but if it is close to 100%, then it may be beneficial to increase the table_cache system variable. Second, the rate of opening tables can also help determine if table_cache is too low. Generally, it is nice to have this value less than 1/s. However, a busy and well running MySQL server can, for example, be opening 7 tables/s and running at 100% table cache.

Connections Report: Lines 55 - 57

Another two line report, the Connection report is practically identical to the Tables report and so it will not be explained again. If the max number of connections used is approaching 100% (first line, third column), the max_connections system variable might need to be increased. However, this is often misleading. One often sees MySQL servers with very high max_connection for no good reason. The default value is 100 and this works for even extremely busy, well-optimized servers. A connection to MySQL should last a fraction of a second, so even 100 connections goes a long way. If max connections is very high or slowly rising over time, the problem might be elsewhere, like slow queries, poor indexing, or even **slow DNS resolution**. Before setting max_connections above 100, discover the fundamental reason why 100 connections at once is not enough and verify that it is a

legitimate need and not actually another problem that manifests itself as too few connections.

Regarding the number of connections per second, this value can be rather high. In fact, if it is high and everything else is working well, it is usually a good indication. Upwards of 10 connections/s is possible, but most server's connections/s are well under 5/s.

Created Temp Report: Lines 59 - 62

MySQL can create temporary tables on hard disk, in RAM, and in temporary files. Each of these three corresponds to the three lines of the Created Temp report. These value are relative; there is no standard for them. Since temporary tables on hard disk are the slowest (indicated by the first line, Disk table), it is best if this value is the lowest of the three. A temporary table is created on hard disk only if it cannot fit into a temporary table in RAM which is limited by the system variable tmp_table_size. The value of tmp_table_size is indicated by Size on the second line, third column. Temporary tables in RAM (indicated by the second line, Table) and temporary files are so common that these value are completely relative to one's MySQL server.

Threads, Aborted, Bytes Reports: Lines 64 - 76

These three reports are the least important. Therefore, they are not discussed in detail and they are mostly self-evident.

There is one line of particular interest: line 66 of the Threads report, Cache and specifically %Hit. Every connection to MySQL is handled by a separate thread. At startup, MySQL creates a few threads and keeps a few in a thread cache so that it does not have to constantly keep killing and creating threads. Although threads are not expensive to make, it is not good to "thread thrash." When the number of connections/s to MySQL exceeds the thread cache (set by the system variable thread_cache_size) MySQL starts to thread thrash: it goes crazy creating threads to keep up with the demand for new connections. When this happens, the thread cache hit rate drops.

Does thread thrashing matter? Yes it does: **Jeremy Zawondy once blogged**:

> So the moral of the story is this: If you have a busy server that's getting a lot of quick connections, set your thread cache high enough that the Threads_created value in SHOW STATUS stops increasing. Your CPU will thank you. ... Thread caching really wasn't the worst of our problems. But it became the worst after we had fixed all the bigger ones.

Therefore, if the MySQL server is thread thrashing (indicated by a low Threads Cached %Hit), increase thread_cache_size.

In this example, Thread Cache %Hit is a very poor 0.05% which means nearly every new connection causes MySQL to create a new thread. It is easy to see why: the first column of the same line (66) says there are zero threads left in the cache. Therefore, thread_cache_size should be increased. Also notice the correlation between line 67, threads created, and the earlier line 57, total connections: 201 threads created, 202 total connections. Hence, the near-zero thread cache hit rate.

InnoDB Buffer Pool Report: Lines 78 - 92

The InnoDB reports that follow were added in mysqlreport v3.0. The Innodb_ status values (in SHOW STATUS;) are only available in MySQL v5.0.2 and later. Therefore, mysqlreport may not show these InnoDB reports even if the MySQL is running the InnoDB storage engine. In short: mysqlreport's InnoDB reports only work with MySQL v5.0.2 and later.

A central feature of the InnoDB storage engine is the buffer pool in which InnoDB caches table data and indexes. Internally, the buffer pool is composed of 16Kb pages which contain different types of data. The InnoDB Buffer Pool Report contains values pertaining to the pages in the buffer pool.

NOTE: I have not seen enough mysqlreports from MySQL servers that are both version 5.0.2 or newer and relying heavily on InnoDB. Therefore, the guide covering the InnoDB reports may seem less thorough than the guide covering the previous reports. Although MySQL v5 has been the GA release for awhile now, it is amazing how common v4.0 and v4.1 still are.

Usage: Line 79

Line 79, InnoDB Buffer Pool Usage is similar to line 4, Key Buffer used. However, the MyISAM engine stores only indexes in its key buffer (hence the name), but the InnoDB engine stores indexes and other data in the buffer pool. Therefore, this line shows how much of the buffer pool is being used but it does not show what accounts for the usage. To get an idea of what accounts for the buffer pool usage, one must look at lines 81 through 85.

Obviously, one must avoid running out of buffer pool space. With the MyISAM engine, running out of key buffer space may only cause performance problems (because of the adverse effect on table indexes). With the InnoDB engine, running out of buffer pool space can cause many more problems because nearly everything relies on the buffer pool. It is possible to **configure an auto-extending buffer pool**.

Read hit: Line 80

The InnoDB Buffer Pool Read hit is very similar to Key Read hit on line 7. However, given that InnoDB stores more than just indexes in the buffer pool, this value is more general then Key Read hit.

InnoDB Buffer Pool Read hit indicates the percentage of buffer pool page reads (hits) from RAM (verses from hard disk). Therefore, this percentage should be very near 100.00%. In most cases, this percentage is > 99.98%.

Pages: Lines 81 - 85

These lines are a very broad look into the substance of the buffer pool. Each line (82 - 85) corresponds to a different kind of buffer pool page: free pages (line 82), data pages (83), miscellaneous pages (84), and "latched" pages (85).

Free pages are self-describing. The far-right column, "%Total:", indicates what percentage of all buffer pool pages are free (or data, misc, and latched correspondingly). This line is the opposite of line 79: instead of saying how much of the buffer pool is used (line 79), this line says how much of the buffer pool is free.

Data pages are also self-describing. Currently there is no way to know the kinds of data that these pages comprise. This line has one extra column: %Drty (%Dirty). This column indicates what percentage of data pages have been modified (are dirty) but have not been flushed/saved back to hard-disk.

Not much can be said about the remaining two kinds of pages: misc and latched. Regarding miscellaneous

pages, the MySQL manual simply says: "The number of pages that are busy because they have been allocated for administrative overhead such as row locks or the adaptive hash index." And regarding latched pages: "These are pages currently being read or written or that cannot be flushed or removed for some other reason."

Reads: Lines 86 - 89

The following four lines give some indication of InnoDB's buffer pool reading activity. The first line, line 86, is simply a metric of the number of buffer pool reads from RAM. On busy servers using InnoDB, this value should be high because InnoDB should be reading most of its pages from the buffer pool in RAM. This metric can been considered a measure of InnoDB buffer pool throughput. Since almost everything InnoDB needs is kept in and retrieved from its buffer pool, buffer pool reads should be as fast as possible. For example, an InnoDB Buffer Pool Reads rate (second column) of over 200k/s is not impossible.

Line 87, however, should be much smaller in value. Line 87 lists "The number of logical reads that InnoDB could not satisfy from the buffer pool and had to do a single-page read." In other words, how many buffer pool page reads from hard disk.

Line 88, Ahead Rnd (Random), lists "The number of random read-aheads initiated by InnoDB. This happens when a query scans a large portion of a table but in random order."

Line 89, Ahead Sql (Sequential), lists "The number of sequential read-aheads initiated by InnoDB. This happens when InnoDB does a sequential full table scan." As always, full table scans are usually a bad thing and should be minimized.

Writes: Line 90

Like line 86, InnoDB Buffer Pool Writes can been considered a measure of InnoDB buffer pool throughput. This line lists the number and rate of writes to the buffer pool. This value will probably be high if the server does a lot of UPDATEs or INSERTs.

Flushes: Line 91

This line is simply the number of buffer pool page-flush requests.

Wait Free: Line 92

Quoting from the MySQL manual regarding this status value:

> Normally, writes to the InnoDB buffer pool happen in the background. However, if it is necessary to read or create a page and no clean pages are available, it is also necessary to wait for pages to be flushed first. This counter counts instances of these waits. If the buffer pool size has been set properly, this value should be small.

InnoDB Lock Report: Lines 94 - 100

InnoDB row lock status values were added in MySQL 5.0.3. MyISAM is a table-level locking storage engine, but InnoDB is a row-level locking store engine. Therefore, these values are important to consider when using the InnoDB engine.

Waits: Line 95

This line shows "The number of times a row lock had to be waited for." Zero is best.

Current: Line 96

This line shows "The number of row locks currently being waited for." Zero is best.

Time acquiring: Lines 97 - 100

Lines 98, 99, and 100 show millisecond times corresponding to the Total number of milliseconds that row locks had to wait (line 98), the Average wait time (99), and the Maximum wait time (100). For all three metrics, zero is best.

InnoDB Data, Pages, Rows Report: Lines 102 - 121

The InnoDB Data, Pages, Rows Report is very general and, as such, is probably only useful as general indicator of InnoDB engine throughput. The three sections, Data, Pages, and Rows, give a very broad look into InnoDB's activity.

Data: Lines 103 - 110

The first section, Data, lists four categories corresponding to InnoDB data: Reads, Writes, fsync, Pending. The first category, Reads, refers to the total number of data reads done by the InnoDB engine. This does not mean the total number of data bytes read. It only means how many times InnoDB has read data; it does not indicate what kind of data or how much data was read.

The second category, Writes, is just like Reads: it refers to the total number of data writes done by InnoDB, but it does not indicate what kind of data or how much data was written.

The third category, fsync, refers to the total number of file system syncs. In other words: how many times InnoDB has saved data from RAM back to hard-disk. This value will tend to be lower than Reads or Writes.

Pending, the final category, is further divided into 3 lines (lines 108, 109, 110): Reads, Writes, fsync. Correspondingly, these lines refer to the current number of Data Reads, Writes, and fsyncs that are pending (waiting). Zero is best.

Pages: Lines 112 - 115

The Pages section has three categories: Created, Read, Written. Each category is self-describing and all three refer to pages in the InnoDB buffer pool. These values indicate the number and rate at which pages in the buffer pool have been created, read, and written. However, none of the values indicate what kind of pages. Therefore, these values are also only useful as general indicators of InnoDB engine throughput.

Rows: Lines 117 - 121

The final section, Rows, is last because it is the most general. Each of the four categories in this

section (Deleted, Inserted, Read, Updated) refer to rows in InnoDB tables. Therefore, these values tend to be very large and, while their counts (first column) may indicate little, their rates (second column) are another indicator of InnoDB engine throughput.

Conclusion

Now that we have read the entire mysqlreport report and considered it all, we can make a general assessment of this example server.

In general, the server is running very well according to a number of big indicators: key buffer usage is only at 12%, key ratios are good, DMS and QC Hits account for over 99% of all questions, no weird Com_ problems, table locks are good, table cache is only at 10% usage, and relatively low and slow number of connections.

Concerning the InnoDB engine, it appears to be in use, but not heavily. As best as the InnoDB status values can inform us, nothing is out of the ordinary in this case.

Things we could work on include, first and foremost, the query cache because it is too volatile, and secondly we must set thread_cache_size higher until the thread cache hit rate comes back up.

That is all there is too it. If you have further questions you can **contact** me. And, if you did not notice, there are a number of other example mysqlreport reports on the **mysqlreport web page**. Although the example reports are from varying older versions of mysqlreport, the format is still similar.

This guide was last updated on April 16, 2008 for mysqlreport v3.5.