



Example Scenario One



DB Tuna

Current

Activity

Statistics

Variables

Time:

Last Hour

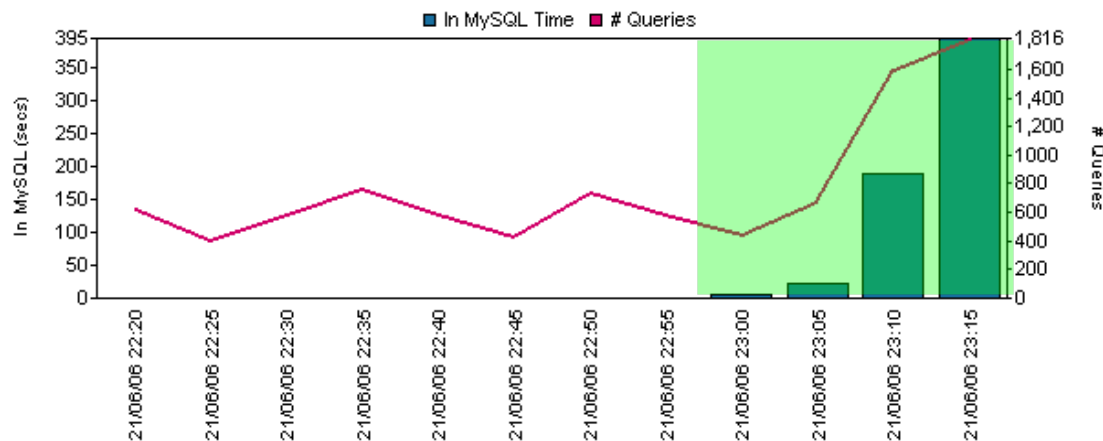
View:

SQL Grouped

Server:

KENNY

In MySQL Overtime



SQL Wait States



Entity	Resource Consumption	Weighted
251147298 select * from petshop where pet_type = :s	00:09:52	96.26%

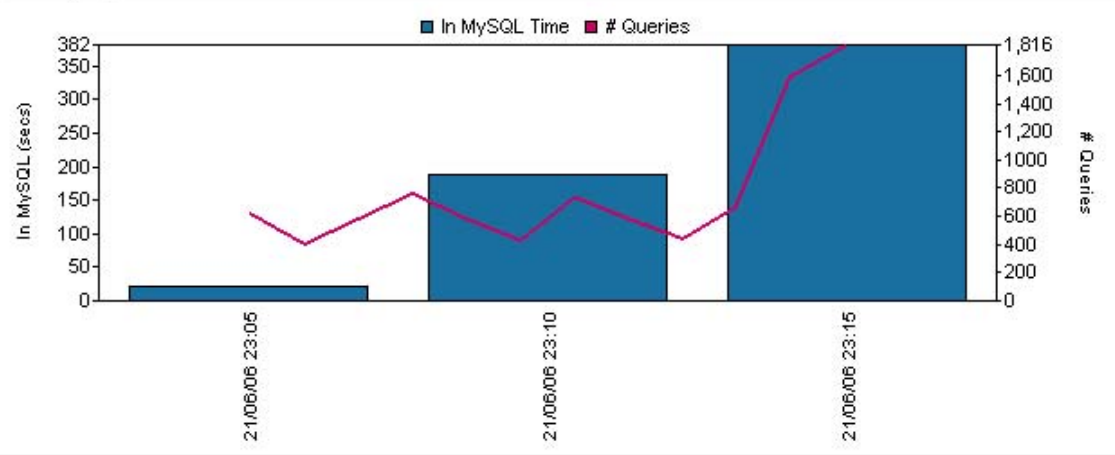
- DBTuna clearly displays an increase in database resource consumption directly correlated with an increase in the number of SQL statements
- The SQL Grouped view shows that 96.26% of this resource consumption is because of an individual statement i.e. "select * from petshop where pet_type = :s"



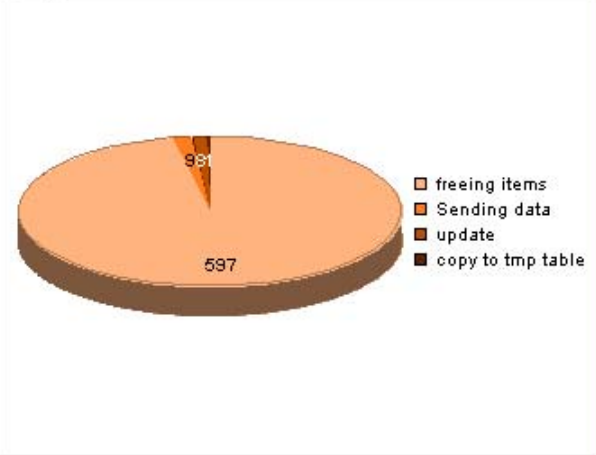
- Current
- Activity
- Statistics
- Variables

Time: Last Hour | View: SQL | Server: KENNY

In MySQL Overtime



SQL Wait States



Filter: ss.sql_collapsed = '251147298'

Entity	Resource Consumption	Weighted
select * from petshop where pet_type = 'dog'	00:04:19	42.11%
select * from petshop where pet_type = 'snake'	00:02:08	20.81%
select * from petshop where pet_type = 'bird'	00:01:41	16.42%
select * from petshop where pet_type = 'dog'	00:01:15	12.20%

■ Drilling down into the grouped statement reveals the real literals in the statement. We can now see the resource consumption difference between a search for a dog and one for a snake!



- Current
- Activity
- Variables
- Statistics
- SQL**

SQL

```
SELECT *  
FROM petshop  
WHERE pet_type = 'dog'
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	petshop	ALL					732059	Using where

petshop

Field	Type	Null	Key	Default	Extra
pet_type	varchar(20)	YES			
stat_code	varchar(12)	NO			
stat_type	char(1)	NO			
yearmonth	int(4)	NO		0	
count	int(11)	NO		0	
mean_price	int(11)	YES			
median_price	int(11)	YES			

petshop Indexes

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment
-------	------------	----------	--------------	-------------	-----------	-------------	----------	--------	------	------------	---------

- Clicking on the SQL text takes you through to the SQL view
- The MySQL explain plan and information on the referenced objects are displayed
- We can see that the statement is accessing the full table because there are no indexes to use



- Current
- Activity
- Variables
- Statistics
- SQL**

SQL

```
SELECT *  
FROM petshop  
WHERE pet_type = 'dog'
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	petshop	ref	pet_index	pet_index	23	const	1	Using where

petshop

Field	Type	Null	Key	Default	Extra
pet_type	varchar(20)	YES	MUL		
stat_code	varchar(12)	NO			
stat_type	char(1)	NO			
yearmonth	int(4)	NO		0	
count	int(11)	NO		0	
mean_price	int(11)	YES			
median_price	int(11)	YES			

petshop Indexes

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment
petshop	1	pet_index	1	pet_type	A	2882			YES	BTREE	

- The solution in this example is easy. The DBA needs to create an index on pet_type in order to make the query more selective. This was done immediately, and then checked in the DBTuna SQL view



Current

Activity

Statistics

Variables

Time:

Last Hour

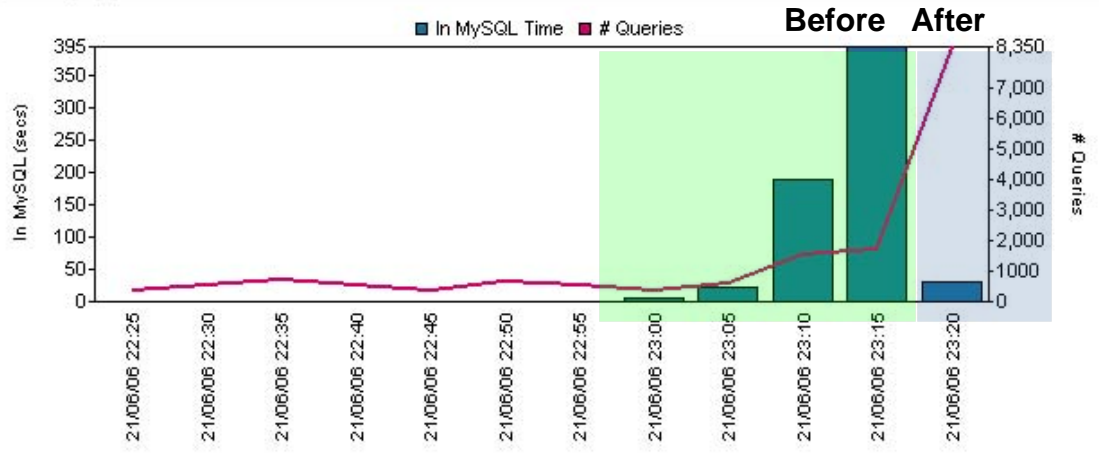
View:

SQL

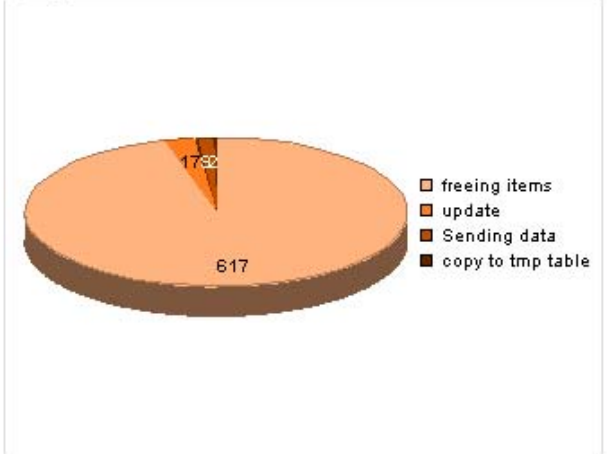
Server:

KENNY

In MySQL Overtime



SQL Wait States



Entity	Resource Consumption	Weighted
<code>select * from petshop where pet_type = 'dog'</code>	00:04:37	42.95%
<code>select * from petshop where pet_type = 'snake'</code>	00:02:08	19.84%
<code>select * from petshop where pet_type = 'bird'</code>	00:01:41	15.86%
<code>select * from petshop where pet_type = 'dog'</code>	00:01:15	11.63%
<code>select * from petshop where pet_type = 'cat'</code>	00:00:29	4.50%
<code>create index pet_index on petshop(pet_type)</code>	00:00:05	0.78%

- Following the new index we can see that resource consumption has dropped substantially, and throughput (# queries/second) has increased by at least 4x

