

Diagnosing and Fixing MySQL Performance Problems

Percona, Inc.

<http://www.percona.com/>

Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

Introduction

- ★ Welcome Everybody!
- ★ Your hosts for the next 3 1/2 hours are **@jswanhart** and **@RoomieGunns** from **@percona**.

House Keeping

- ★ **Restrooms?** -> Leave if you need to.
- ★ **Questions?** -> Ask at any time.
- ★ **Heckling?** -> There will be a prize for the best heckle.

- ★ I hope you enjoyed lunch. We'll be taking one scheduled break in the middle of the tutorial..

Wifi / Power?!

- ★ Everyone connected, are the wifi instructions posted?
- ★ Be kind to your neighbour - share the power strips in the room.



Where does this talk come from?

- ★ If you're curious – some of this presentation comes from the slide decks we use at our training.
- ★ However, we've prepared this material for this conference.

Can we get a show of hands:

- ★ MySQL 5 / MySQL 5.1 / MySQL 5.5 / Anything else?
- ★ Linux / Windows / Solaris / BSD / Something else?
- ★ Oracle / SQL Server experience?
- ★ What are you interested in knowing -
 - ◆ Problems you are currently having?
 - ◆ Weird things that happened you could never explain?
 - ◆ Something else?

Can you have this slide deck?

- ★ We'll give you the URL at the very end of the day.
- ★ Be careful to write it down!
 - ◆ O'Reilly doesn't normally upload tutorial slides the same way they do for conference sessions.

Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

What *about* Performance?

- ★ Performance optimization should **start** with a goal.
- ★ That goal can be to -
 - ◆ Improve time for users?
 - ◆ Reduce load?
 - ◆ Grow the service?

What performance is not:

- ★ CPU Usage
- ★ Memory Usage
- ★ Load Average

Performance is best measured as:

- ★ The response (R) time for a user's task.
- ★ What is a task?
 - ◆ a unit of work
 - ◆ a business unit of work
 - ◆ something a user wants to do
 - ◆ some function of the application

Related Concepts

Load	Utilization	Scalability
Throughput	Concurrency	Capacity

Related Concepts (cont.)

Load: how much work is incoming? or, how big is the backlog?	Utilization	Scalability
Throughput	Concurrency	Capacity

Related Concepts (cont.)

Load: how much work is incoming? or, how big is the backlog?	Utilization: how much of a system's resources are used?	Scalability
Throughput	Concurrency	Capacity

Related Concepts (cont.)

<p>Load:how much work is incoming? or, how big is the backlog?</p>	<p>Utilization: how much of a system's resources are used?</p>	<p>Scalability: what is the relationship between utilization and R?</p>
<p>Throughput</p>	<p>Concurrency</p>	<p>Capacity</p>

Related Concepts (cont.)

<p>Load: how much work is incoming? or, how big is the backlog?</p>	<p>Utilization: how much of a system's resources are used?</p>	<p>Scalability: what is the relationship between utilization and R?</p>
<p>Throughput: X - how many tasks can be done per unit of time?</p>	<p>Concurrency</p>	<p>Capacity</p>

Related Concepts (cont.)

<p>Load: how much work is incoming? or, how big is the backlog?</p>	<p>Utilization: how much of a system's resources are used?</p>	<p>Scalability: what is the relationship between utilization and R?</p>
<p>Throughput: X - how many tasks can be done per unit of time?</p>	<p>Concurrency: how many tasks can we do at once?</p>	<p>Capacity</p>

Related Concepts (cont.)

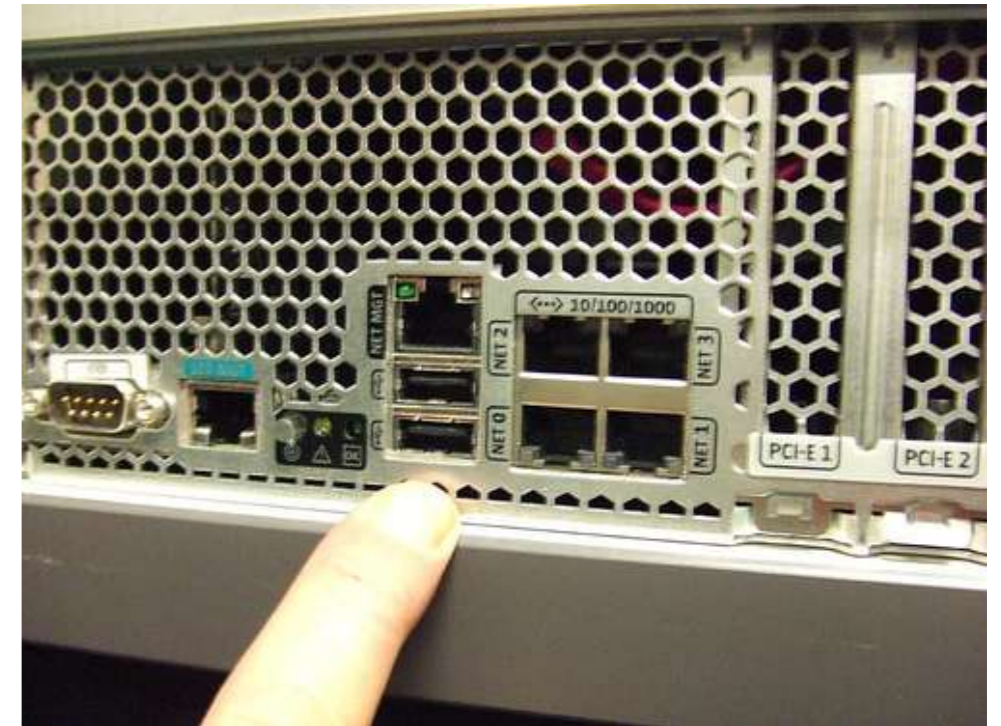
<p>Load: how much work is incoming? or, how big is the backlog?</p>	<p>Utilization: how much of a system's resources are used?</p>	<p>Scalability: what is the relationship between utilization and R?</p>
<p>Throughput: X - how many tasks can be done per unit of time?</p>	<p>Concurrency: how many tasks can we do at once?</p>	<p>Capacity: how big can X go without making other things unacceptable?</p>

Throughput \neq Performance

$$R = \text{Time} / \text{Task}$$
$$X = \text{Tasks} / \text{Time}$$

Why Throughput != Performance

- ★ Big Niagara machines with many cores, each of them very slow.
- ◆ Also not reciprocal of performance, because....



Sun T2000. Photo Credit:
<http://www.flickr.com/photos/clone/milkmen/310565533/>

Throughput != Performance (cont.)

★ Parallelism

- ♦ 10 tasks at once take 10 seconds.
- ♦ $X = 10/10 = 1$, $R = 10$

★ Non-uniform distribution

- ♦ 100 tasks take 100 seconds, what is R ? Don't know, it could be 99 fast and 1 slow task.

What is important...

- ★ Is the relationship between throughput, utilization, response time and capacity.
- ★ Queuing may occur:
 - ◆ R is the combination of service time and wait time.

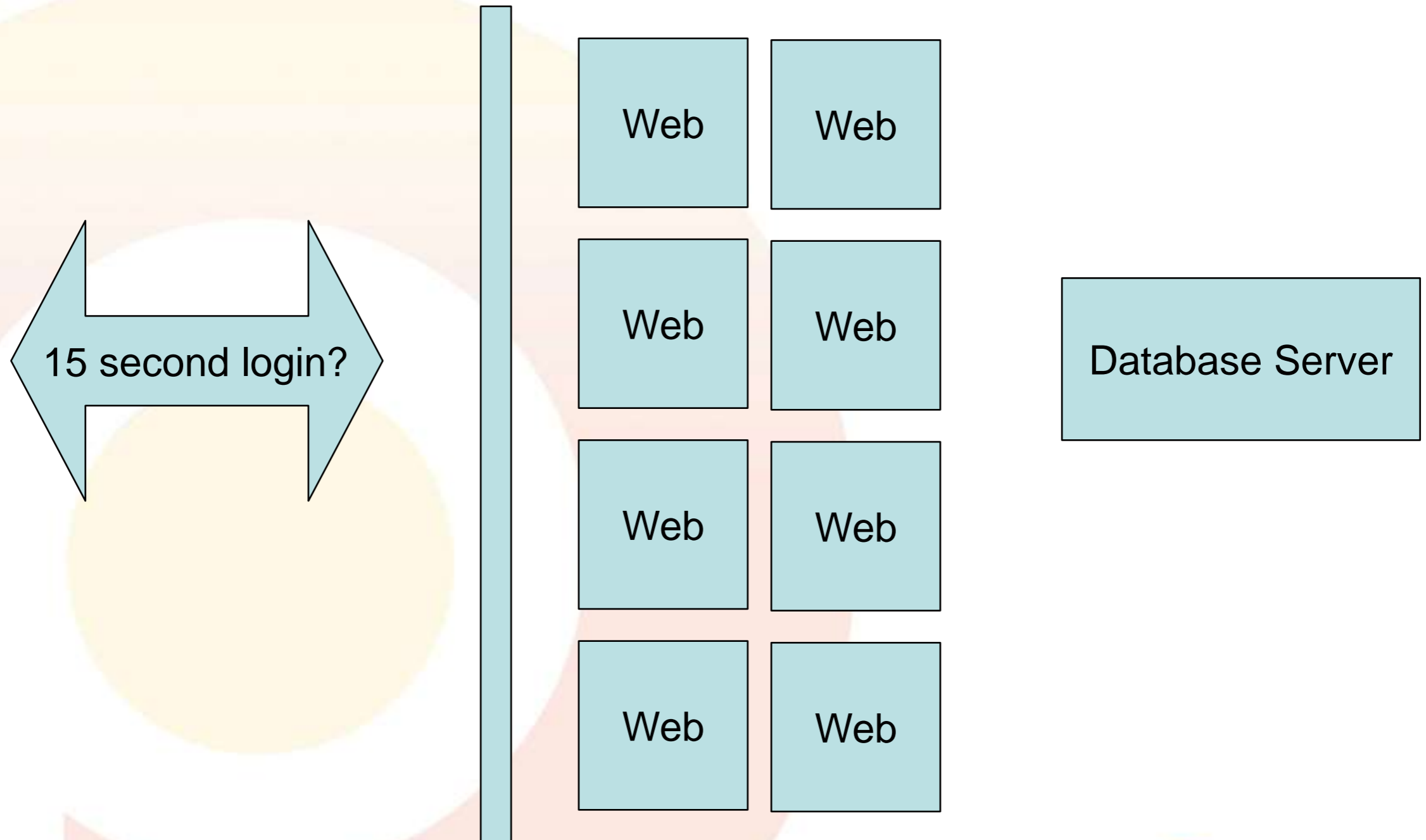
What to take away:

- ★ Above all, focus on time.
 - ◆ For performance optimization, reduce time spent waiting for response.
 - ◆ For load reduction, reduce total time consumed by the task.

Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

What is the performance problem?



If you said that...

★ The Database

- ♦ You'll be right most of the time - but you're not being 100% honest with yourself.
- ♦ The database has more scalability challenges than the other components. For the most part we can just add web servers.

However;

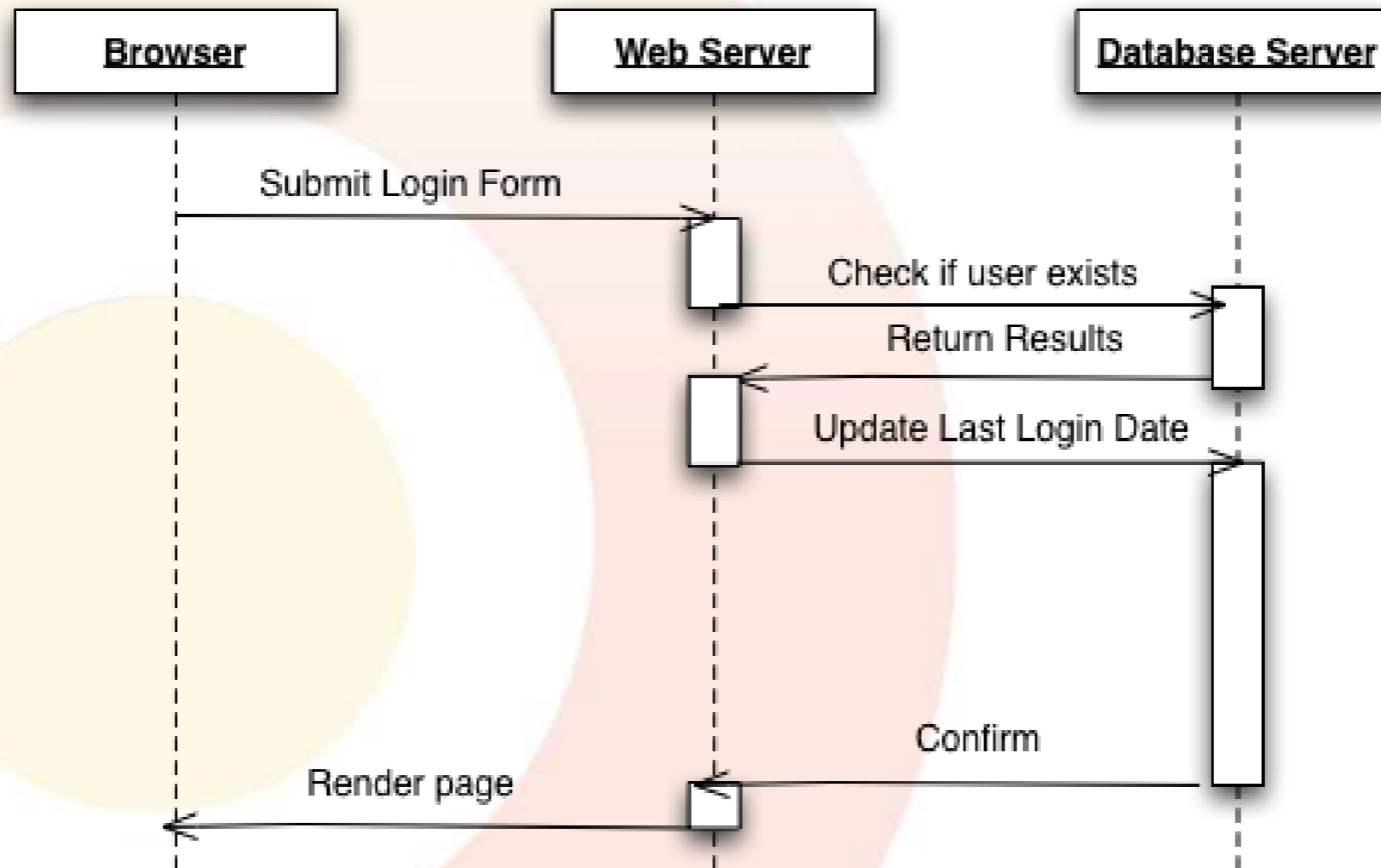
- ★ We can lead ourselves into a real trap by guessing based on previous experience.
- ★ Proving is probably *a lot more important* than knowing.

What's interesting...

- ★ What's more interesting than drawing the stack is drawing **the flow of information** between each component of the stack.
- ★ It's important to be able to do this while users execute **tasks**.

Following the Flow:

- ★ For a given task, measure the breakdown in time:



Wait, what.!?

- ★ Updating the **last_login_date** takes a sizeable amount of time?
- ★ For the value that it provides, why are we spending so long on that sub-task?

My analysis:

★ Query is:

- ◆ UPDATE users SET last_login_date=NOW()
WHERE id = N;

★ Schema is:

- ◆ CREATE TABLE users (
id INT NOT NULL PRIMARY KEY auto_increment,
username CHAR(32) NOT NULL,
..
last_login_date DATETIME,
UNIQUE (username)
) ENGINE=MyISAM;

Id	User	Host	db	Command	Time	State	Info
1	root	localhost	myapp_production	Query	0	NULL	show processlist
9688	root	localhost	myapp_production	Query	2	Sending Data	SELECT COUNT(*) from users
9689	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 986755
9690	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 607334
9691	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1802251
9692	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1076084
9693	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 141037
9694	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1418038
9695	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1156819
9696	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 165878
9697	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1345988
9698	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1783549
9699	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 665358
9700	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 168566
9701	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1531867
9702	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 931161
9703	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 342250
9704	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 437672
9705	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 976963
9706	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 615735
9707	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1152889
9708	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1748237
9709	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 652162
9710	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1067106
9711	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1920992
9712	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1698141
9713	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1649822
9714	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 94358
9715	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 983337
9716	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1091145
9717	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 255341
9718	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 25397
9719	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1223432
9720	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1001712
9721	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1995106
9722	root	localhost	myapp_production	Query	2	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 508775
9723	root	localhost	myapp_production	Query	1	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1121464
9724	root	localhost	myapp_production	Query	1	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 946613
9725	root	localhost	myapp_production	Query	1	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1153740
9726	root	localhost	myapp_production	Query	1	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1656344
9727	root	localhost	myapp_production	Query	1	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 102982
9728	root	localhost	myapp_production	Query	1	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 1672517
9729	root	localhost	myapp_production	Query	1	Locked	UPDATE users SET last_login_date=NOW() WHERE id = 350907
...							

101 rows in set (0.00 sec)

```
$ uptime
```

```
15:00 up 11 days, 16:58, 5 users, load averages: 0.88 0.61 0.44
```

```
$ uptime
```

```
15:00 up 11 days, 16:58, 5 users, load averages: 0.88 0.61 0.44
```

```
mysql> show global status like 'slow_queries%';
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Slow_queries  | 3     |
+-----+-----+
1 row in set (0.00 sec)
```

```
$ uptime
```

```
15:00 up 11 days, 16:58, 5 users, load averages: 0.88 0.61 0.44
```

```
mysql> show global status like 'slow_queries%';
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Slow_queries  | 3     |
+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> show global status like 'table_lock%';
```

```
+-----+-----+
| Variable_name          | Value |
+-----+-----+
| Table_locks_immediate  | 2267  |
| Table_locks_waited    | 15640 |
+-----+-----+
2 rows in set (0.00 sec)
```

Why do I show you this example?

- ★ A lock is a form of queue.
 - ◆ $R = \text{Queue} + \text{Service Time}$
- ★ Some queuing shows up as “Load”.
 - ◆ CPUs have a run queue.
 - ◆ IO requests are queued before execution.

However;

- ★ Not all queuing shows up in the Operating System - such as internal locking contention.

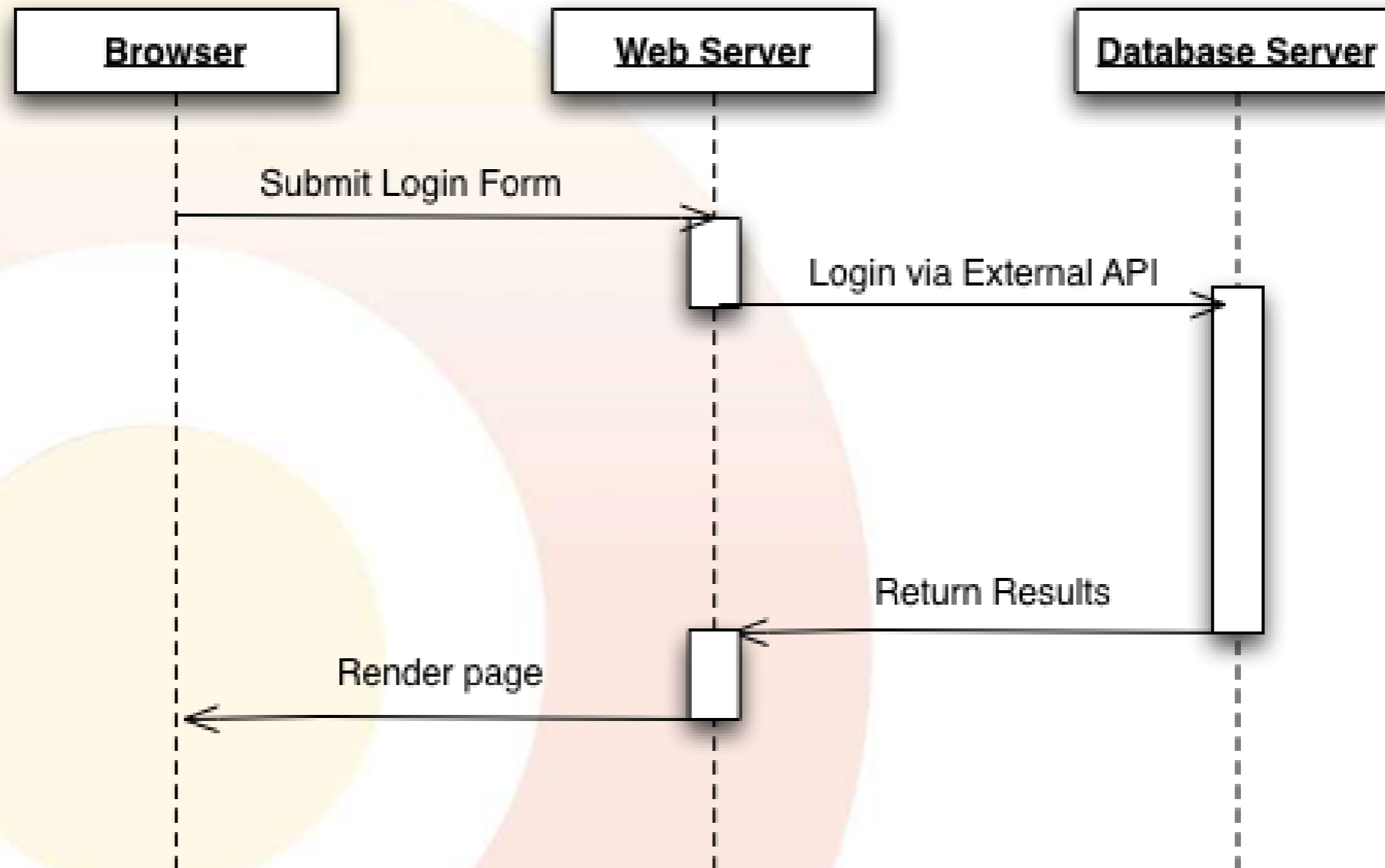
Cause: More people request a resource than current capacity that can be offered.	Possible Effect: System may look busy (CPU or Disks).
--	---

- ★ It's always better to instrument for the cause - as it may still be there without any 'effect' present.

Back to the Sequence Diagram

- ★ You want to *find what doesn't look right*. Then take those quick wins.
- ★ This sounds trivial - but without instrumentation your guesses are often not correct.

Another Sequence Diagram:



The Lesson

- ★ In this case, the external API looks like a great piece to attack first.
- ★ How do we improve it?
 - ◆ Maybe a developer can help here..
 - ◆ Can we cache the credentials?

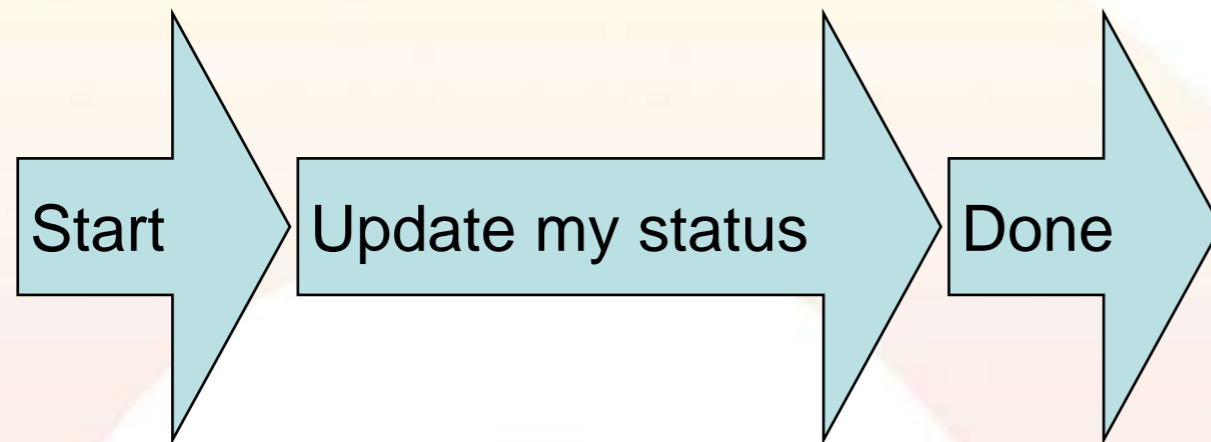
The Important Lesson (cont.)

- ★ We had a customer exactly like this:
 - ◆ They looked at their webservers.
 - ◆ They weren't loaded.
 - ◆ They looked at their database servers.
 - ◆ They weren't loaded either.
 - ◆ They tried to optimize their database servers because 'it was a database problem last time'.
 - ◆ Performance wasn't much better.

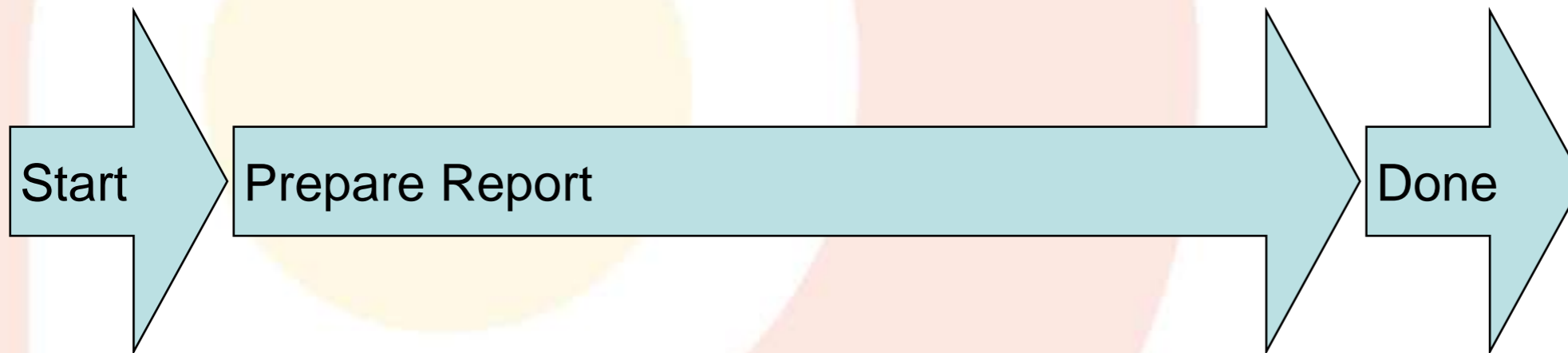
Why do we set different goals for
each task?

Two different Tasks:

- ★ Sequence #1: (Total time 0.01 seconds)



- ★ Sequence #2: (Total time 60 seconds)

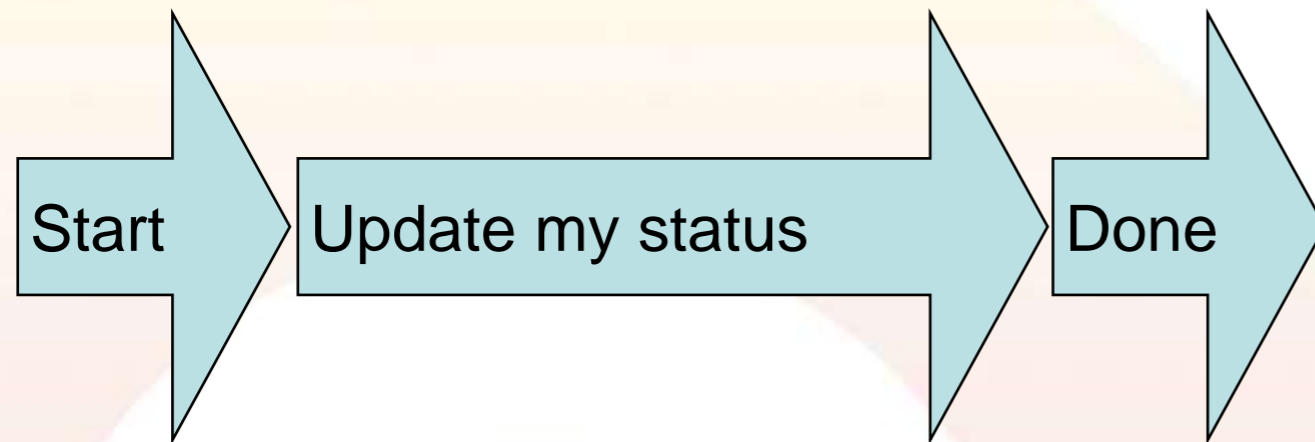


The “Optimization”

- ★ The customer was worried about a 60 second long query that ran at 3am every night.
- ★ Oh no, let's optimize it by adding an index!

The outcome:

- ★ Sequence #1: (Total time 0.02 seconds)



- ★ Sequence #2: (Total time 10 seconds)



Optimize what matters:

- ★ The (ideally user facing) tasks matter.
- ★ Write instrumentation so that you can think like a sequence diagram does.

Limits of Sequence Diagrams:

- ★ Why you might not see everyone using a sequence diagram in production -
 - ◆ It breaks down when there's too many sub-tasks involved in execution.

Introducing the Profile

Profile for the "Product Search" Task

Rank	Sub-Task	R	R%	Calls	R/Call
1	DB::fetch_items	8.98	48.5%	1	8.98
2	App::Render_item	6.95	37.5%	50	0.14
3	Sphinx::search	1.43	7.7%	1	1.43
4	[Missing Time]	1.01	5.5%	-	-
5	App::Show_header	0.11	0.6%	1	0.11
6	App::Paginate	0.03	0.2%	12	0.00

Introducing the Profile (cont.)

- ★ The profile is related to the sequence diagram;
 - ◆ What we do in a profile is aggregate similar sub-tasks into one entry.
 - ◆ Then we order the profile from total R of a given task.

Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

The Method for Isolation

- ★ In general:
 - ◆ Start at the user, measure time taken from the front to the backend.
- ★ The absolutely certain method:
 - ◆ Plod through the sequence diagram, measure each step.

The sequence diagram

★ Advantage:

- ◆ You will inevitably find the solution and be able to prove it.

★ Disadvantages:

- ◆ might take too much time
- ◆ things might be too hard or impossible to measure
- ◆ there might be too much information

Method for Isolation (cont.)

- ★ Slightly more practical preferred way:
 - ◆ 1. Use a tool that makes it easy
 - ◆ 2. If that's not possible, guess and measure.
 - ◆ 3. If you can't measure anything, reason from logic.

Method #1: Instrument by time

- ★ Our favourite - and where you should start.
- ★ Only Con:
 - ◆ Not always that easy to do in a complicated system.
 - ◆ May not always be feasible to install instrumentation

Method #2: Guess and Measure

- ★ Look for a part of the system that “looks bad”
- ★ How much time does it consume?
- ★ Example: If logging in takes 2.1 seconds and DNS takes 2 seconds, you found the problem.

Method #3 - Reason from Logic

- ★ Really hard, really slow.
- ★ If you can't do any of those, you can
 - ◆ ask for help
 - ◆ trial-and-error (more on this later)
- ★ We suggest asking for help.

What not to do.

Work on Unimportant Things

- ★ **[DO]** Apply Amdahl's Law: a 5% time consumer can't benefit you more than 5%.
- ★ **[DON'T]** Apply "best practices"
 - ◆ These can have unexpected side effects.
 - ◆ Example: I see many `sort_merge_passes`, maybe I need to increase `sort_buffer_size`.
 - ◆ Example: I've heard that I should put the logs and data on different disks.

Guess-and-don't-measure

- ★ If you skip measuring, then you cause problems.
 - ◆ This is also known as trial-and-error.
- ★ This method has 3 interesting properties:
 - ◆ It sometimes works.
 - ◆ It often wastes a lot of time and makes things worse.
 - ◆ It can cost you your job.

Guess and Feel-Good-Measure

★ Apply BS-Benchmarks:

- ◆ Time how long it takes to create 10,000 tables and conclude InnoDB is slower.
- ◆ Write an application benchmark that doesn't mimic that of your application (not enough data, single threaded only).

The best tool to make things easy
is a profiler.

System-wide

- ★ System-wide:
 - ◆ oprofile, strace -c
- ★ Perl: Devel::NYTProf
- ★ Ruby: ruby -r profile (or just NewRelic)
- ★ PHP:
 - ◆ Instrumentation-for-PHP - <http://code.google.com/p/instrumentation-for-php>
 - ◆ Xdebug + KCachegrind -- but not in production
 - ◆ XHPProf <http://mirror.facebook.net/facebook/xhprof/>
 - ◆ New Relic - <http://www.newrelic.com>

Instrumentation Demo

- ★ We've got it running here:
 - ◆ <http://204.236.183.243/my-movies>

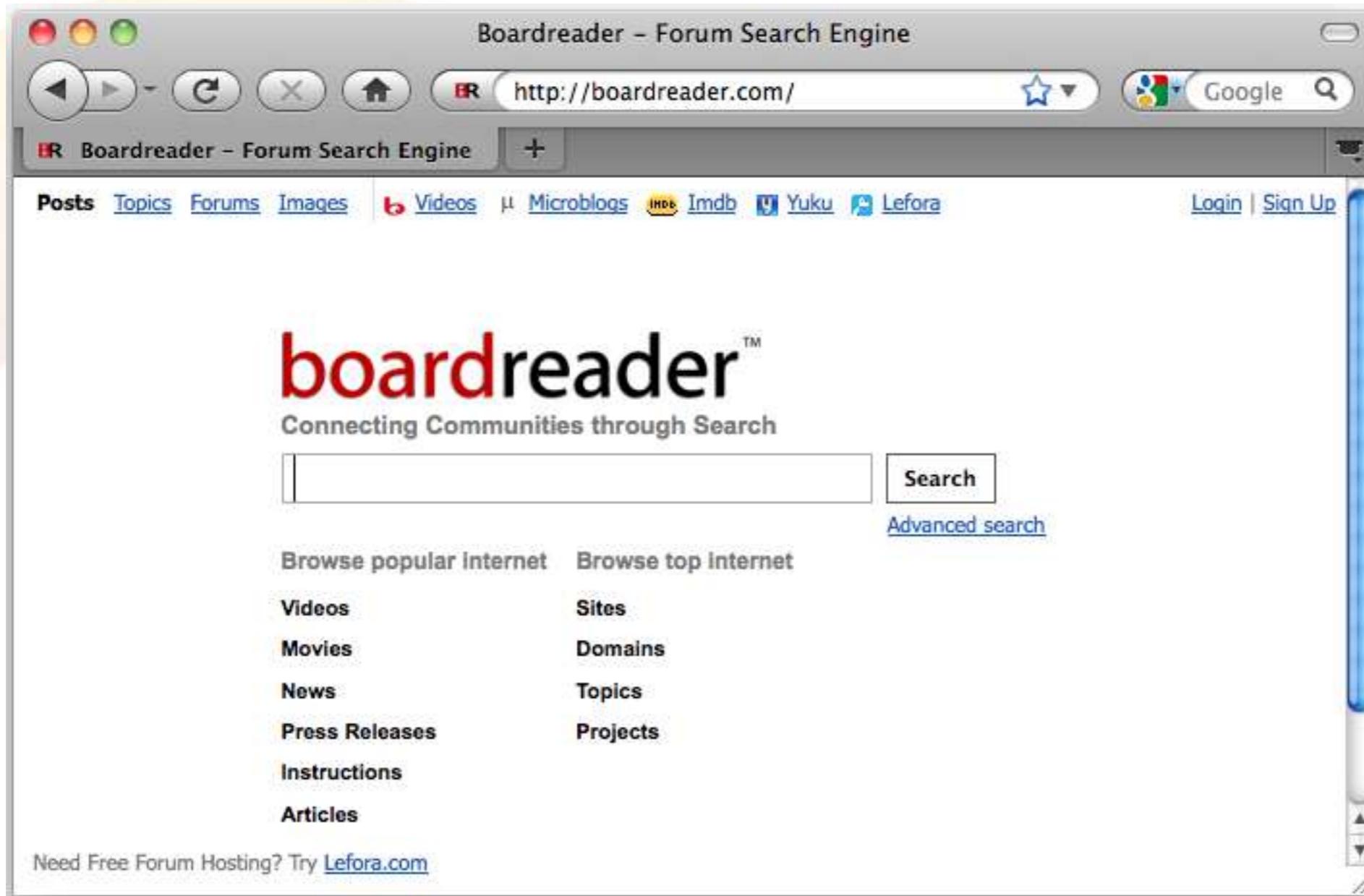
Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

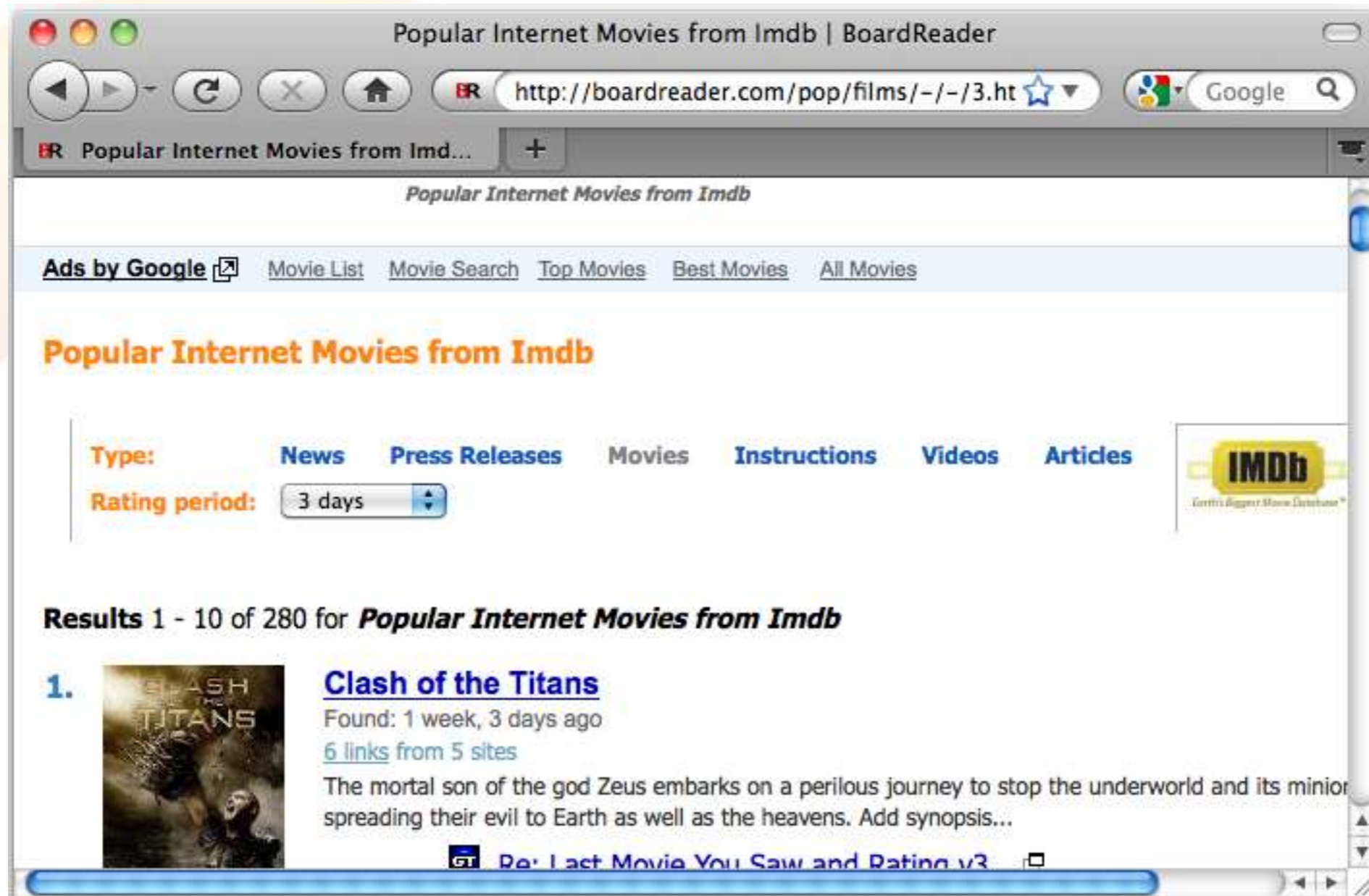
Real Life.

- ★ Custom written profiling code from boardreader.com.
 - ◆ Works similar to what you can achieve with NewRelic and XHProf or Instrumentation-for-PHP

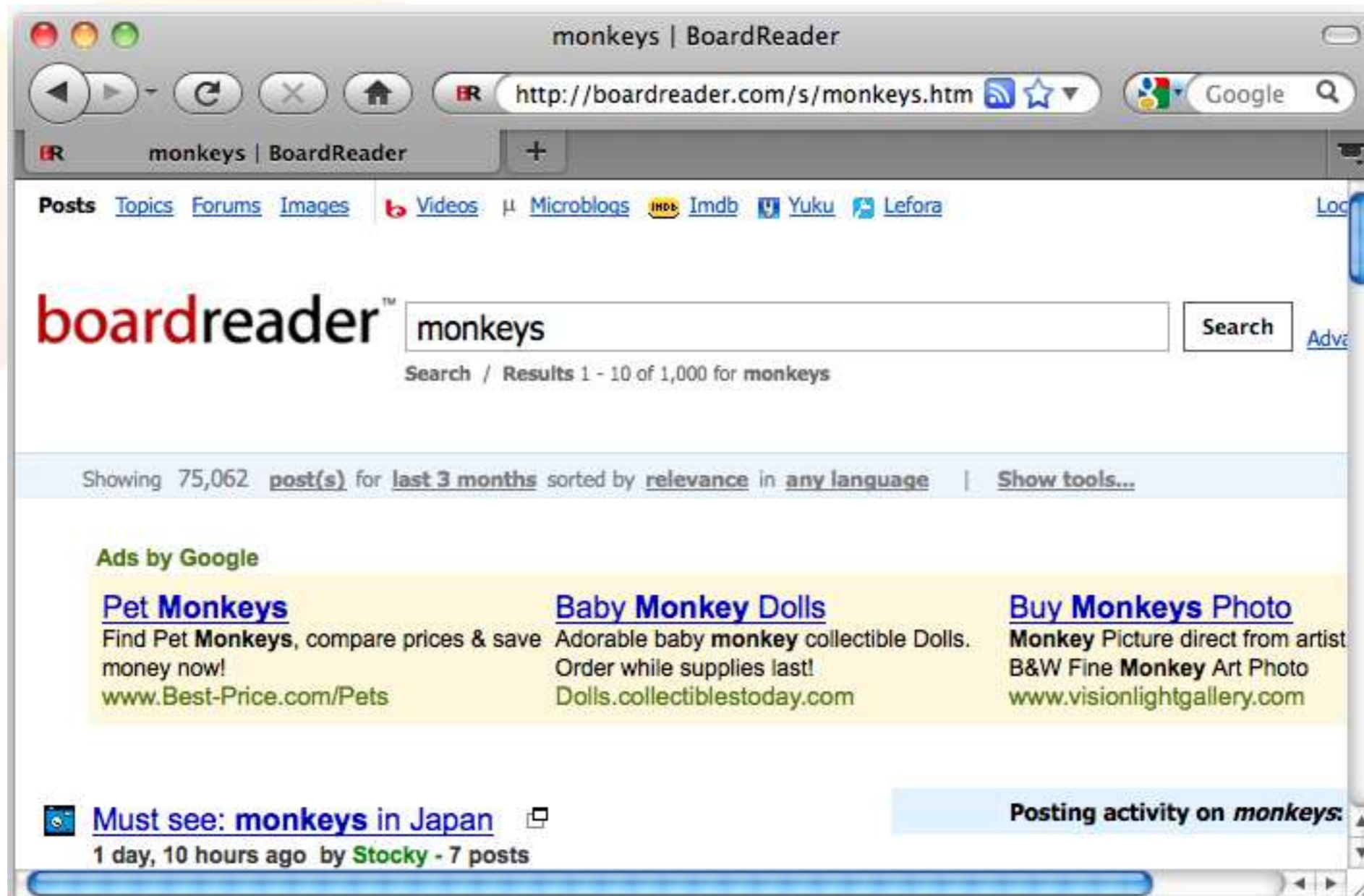
A Screenshot:



Another Screenshot:



Another Screenshot:



User Tasks

- ★ If I had to guess it - I would say that they have a few different tasks that people do:
 - ◆ Loading Static Pages (like the home page)
 - ◆ Browse by a popular topic.
 - ◆ Returning results for a custom search.

Task Goals

- ★ Each of these items (in my mind) has a different response time goal. Let's make some up for context:
 - ◆ Loading the homepage 100ms.
 - ◆ Browsing by a popular topic 500ms
 - ◆ Search requests 2000ms (2 seconds)

Measuring Goals

★ Here's how boardreader.com stores the information:

```
CREATE TABLE `performance_log_090721` (  
  `ip` varchar(15) NOT NULL,  
  `server_ip` varchar(25) NOT NULL,  
  `page` varchar(3000) NOT NULL,  
  `utime` float NOT NULL,  
  `stime` float NOT NULL,  
  `wtime` float NOT NULL,  
  `mysql_time` float NOT NULL,  
  `sphinx_time` float NOT NULL,  
  `mysql_count_queries` int(11) NOT NULL,  
  `mysql_queries` text NOT NULL,  
  `sphinx_count_queries` int(11) NOT NULL,  
  `sphinx_queries` text NOT NULL,  
  ..  
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Measuring Goals (cont.)

- ★ Here's an aggregation of a day's worth of search responses:

```
mysql> select avg(wtime) request, avg(stime+utime)/avg(wtime) cpu
,avg(mysql_time)/avg(wtime) mysql, avg(sphinx_time)/avg(wtime)
sphinx, avg(wtime-stime-utime-sphinx_time-mysql_time)/avg(wtime)
rest from performance_log_090721 where page_type='search' \G
***** 1. row *****
request: 1.2175869055517
      cpu: 0.16983144536072
      mysql: 0.1544487152423
      sphinx: 0.61537297006254
      rest: 0.060346869334443
1 row in set (4.16 sec)
```

Measuring Goals (cont.)

```
★ mysql> select date_format(logged,"%H") h ,round(avg(wtime),3) r,  
round(avg(stime+utime)/avg(wtime),2) cpup ,round(avg(mysql_time)/avg(wtime),2) mp,  
round(avg(sphinx_time)/avg(wtime),2) sp, round(avg(wtime-stime-utime-sphinx_time-  
mysql_time)/avg(wtime),2) rst from performance_log_090721 where page_type='search' group by h;
```

h	r	cpup	mp	sp	rst
00	1.816	0.11	0.14	0.70	0.05
01	1.480	0.17	0.18	0.59	0.06
02	1.394	0.16	0.22	0.53	0.09
...					
08	1.384	0.13	0.09	0.74	0.04
09	1.315	0.17	0.11	0.67	0.04
10	0.950	0.20	0.15	0.60	0.05
11	0.874	0.21	0.16	0.57	0.06
12	1.139	0.17	0.13	0.65	0.05
13	1.191	0.16	0.14	0.65	0.05
14	1.349	0.16	0.19	0.58	0.06
15	1.076	0.20	0.21	0.53	0.06
16	1.526	0.14	0.14	0.58	0.13
17	0.853	0.24	0.19	0.50	0.07
18	0.978	0.25	0.23	0.43	0.09
19	0.924	0.23	0.17	0.54	0.06
20	1.310	0.18	0.26	0.47	0.09
21	1.211	0.17	0.24	0.51	0.08
22	1.538	0.14	0.19	0.59	0.08
23	1.450	0.15	0.18	0.60	0.06

```
24 rows in set (4.33 sec)
```

Interpreting Results

- ★ Always interesting to see the difference in timing on a production system with real load!
- ★ 1 hour as I've got on my slides is probably not the best aggregate - some 'blips' in service may not show up.

Interpreting Results (cont.)

- ★ Average (mean) is easy to do in MySQL - but 95th percentile is probably better.
- ★ What is the problem with min/max/mean?

Actioning Results

- ★ It is only when you've got these metrics in place that you can start to look at where to fix the problem.
- ★ This is where the toolchest comes in.

Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

Quick Glossary of Tools

Top

- ★ Best use it to confirm what is *really* running.
 - ◆ On a DB server mysqld should be at the top.
 - ◆ Most of the time there should only be one mysqld process.
- ★ Example usage:
 - ◆ `top -bn1`

ps

- ★ Quick Trick - verify that the sum of VSZ adds up to roughly the amount of memory used by the system:
 - ◆ `ps -e -o vsz | awk '{size += $1}END{print(size)}'`
- ★ Check for all running servers:
 - ◆ `ps aux | grep mysqld`

free

- ★ Doesn't really show anything vmstat won't.
- ◆ But very handy one line math to show caches. Example:

```
$ free -m
```

	total	used	free	shared	buffers	cached
Mem:	32177	30446	1730	0	368	16649
-/+ buffers/cache:		13428	18748			
Swap:	4095	2	4093			

vmstat

- ★ Best run when server is busy. You can watch what is happening right now. Example:
 - ◆ `vmstat 5`
- ★ General-purpose, but a good way to sense what the system as a whole is doing.

vmstat (cont.)

★ **Most important bits:**

- ◆ si/so should be zero
- ◆ bi/bo are blocks read and written, so you can see IO
- ◆ What does 12% cpu usage mean?

vmstat (cont)

```
$ vmstat 5 5
```

```
procs -----memory----- ---swap-- -----io----- --system-- -----cpu-----  
r  b   swpd   free   buff  cache   si   so    bi    bo    in   cs  us  sy  id  wa  st  
0  3     0  44604  15980 1585532   0   0     3    281    6    9  0  0  97  3  0  
0  3     0  44232  16072 1586572   0   0     1  17718  110  138  0  5  0  94  1  
0  4     0  46960  16152 1581112   0   0    30  18514  108  152  0  5  0  94  1  
0  3     0  46712  16284 1581196   0   0    29  26166  159  201  0  7  0  91  2  
0  3     0  47332  16388 1581220   0   0     1  18990  114  145  0  5  0  94  1
```

mpstat

- ★ Installed as part of sysstat.
- ★ More useful than vmstat because it shows individual CPUs. Example:
 - ◆ `mpstat -P ALL 5`

mpstat (cont)

```
# mpstat -P ALL 5
```

```
Linux 2.6.18-238.el5 (livecd.localdomain)
```

```
03/10/2011
```

11:01:49 PM	CPU	%user	%nice	%sys	%iowait	%irq	%soft	%steal	%idle	intr/s
11:01:54 PM	all	10.31	0.00	4.64	0.00	0.00	0.17	0.00	84.89	1004.00
11:01:54 PM	0	0.20	0.00	0.00	0.00	0.00	0.20	0.00	99.60	1000.20
11:01:54 PM	1	0.00	0.00	0.00	0.00	0.00	0.40	0.00	99.60	0.20
11:01:54 PM	2	15.40	0.00	7.40	0.00	0.00	0.20	0.00	77.00	0.00
11:01:54 PM	3	8.40	0.00	3.60	0.00	0.00	0.20	0.00	87.80	0.00
11:01:54 PM	4	31.60	0.00	14.40	0.00	0.00	0.20	0.00	53.80	0.00
11:01:54 PM	5	6.20	0.00	2.40	0.00	0.00	0.20	0.00	91.20	3.60

netstat (cont.)

★ Show count of states:

- ◆ `netstat -antp | awk '{print $6}' | sort | uniq -c | sort -rn`

★ Show count of peers:

- ◆ `netstat -antp | awk '{print $5}' | cut -d: -f1 | sort | uniq -c | sort -rn`

netstat

★ netstat -antp

- ◆ "ss -ant" works acceptably as well.

```
$ sudo netstat -antp
```

```
Active Internet connections (servers and established)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State	PID/Program name
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN	943/sshd
tcp	0	0	127.0.0.1:25	0.0.0.0:*	LISTEN	965/sendmail: accep
tcp	0	0	0.0.0.0:3306	0.0.0.0:*	LISTEN	20237/mysqld
tcp	0	52	10.170.182.171:22	24.6.173.25:52944	ESTABLISHED	20470/sshd: ec2-use
tcp	0	0	:::22	:::*	LISTEN	943/sshd

netstat (cont.)

- ★ Things to inspect:
 - ◆ Are there a lot of sockets in TIME_WAIT?
 - ◆ A lot going to port 53 (DNS) ?
 - ◆ Who's connecting to mysqld?
- ★ You need to know the TCP handshake process:
 - ◆ SYN, SYN-ACK, ACK, ESTABLISHED.

ping

- ★ useful for checking for packet loss.
- ★ let it run, watch the rtt, cancel,
- ★ look for missing packets.
- ★ icmp packets may be deprioritized

iostat

- ★ **Much** better IO statistics than what vmstat provides.
Example usage:
 - ◆ `iostat -dx 5` (need x for extended statistics)
 - ◆ `iostat -kx 5` (show CPU stats at the same time)
- ★ Main problem is that it lumps reads and writes together.
 - ◆ Look for what the disks are doing and ask if this is reasonable.

iostat (cont.)

- ★ Main items to look at:
 - ◆ What is the queue length?
 - ◆ How much is being read and written?
 - ◆ What is the average wait, and what is the service time?

iostat (cont)

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.11    0.00    5.13   93.91    0.85    0.00
```

```
Device:            rrqm/s   wrqm/s     r/s     w/s    rkB/s    kB/s avgrq-sz avgqu-sz   await  svctm   %util
xvdap1              0.00    35.15     0.00   14.85     0.00   200.00   26.94     1.97     2.49   1.70    2.52
xvdap2              0.00  2819.23     0.00  302.24     0.00 32481.20   82.59    74.39    23.71   1.76   53.29
xvdap3              0.00     0.00     0.00    0.00     0.00    0.00    0.00     0.00     0.00   0.00    0.00
```

Aspersa toolkit

Open source tools which improve upon the toolchest

- ◆ Disk statistics
 - **diskstat** – does not lump reads/writes, interactive
 - **ioprofile** – shows exactly where a process is using IO
- ◆ Diagnostics
 - **stalk / collect** – get well scoped diagnostic data
 - **pmp** – diagnose mutex contention
 - **sift** – quickly filter information from **collect**
 - **summary / mysql-summary** – important info at a glance

Case Study #1

Scope of Problem:

- ★ Overnight the query performance went from <1ms to 50x worse.
- ★ Nothing changed in terms of server configuration, schema, etc.
- ★ Tried throttling the server to 1/2 of its workload
 - ◆ from 20k QPS to 10k QPS
 - ◆ no improvement.

Train of thought

- ★ Change in config client doesn't know about?
- ★ Hardware problem such as a failing disk?
- ★ Load increase: data growth or QPS crossed a "tipping point"?
- ★ Schema changes client doesn't know about (missing index?)
- ★ Network component such as DNS?

Elimination of easy possibilities:

- ★ ALL queries are found to be slower in slow-query-log
 - ◆ eliminates DNS as a possibility.
- ★ Queries are slow when run via Unix socket
 - ◆ eliminates network.
- ★ No errors in dmesg or RAID controller
 - ◆ suggests (doesn't eliminate) that hardware is not the problem.

Easy Elimination (cont.)

- ★ Detailed historical metrics show no change in Handler_ graphs
 - ◆ suggests (doesn't eliminate) that indexing is not the problem.
 - ◆ Also, combined with the fact that ALL queries are 50x slower, very strong reason to believe indexing is not the problem.

Investigation of the obvious:

- ★ Aggregation of SHOW PROCESSLIST shows queries are not in Locked status.
- ★ Investigating SHOW INNODB STATUS shows no problems with semaphores, transaction states such as "commit", main thread, or other likely culprits.

Investigation (cont.)

- ★ However, SHOW INNODB STATUS shows many queries in "" status, as here:
 - ◆ ---TRANSACTION 4 3879540100, ACTIVE 0 sec, process no 26028, OS thread id 1344928080
MySQL thread id 344746, query id 1046183178
10.16.221.148 webuser
SELECT
- ★ All such queries are simple and well-optimized according to EXPLAIN.

Investigation (cont.)

- ★ The system has 8 CPUs, Intel(R) Xeon(R) CPU E5450 @ 3.00GHz
- ★ The system has a RAID controller with 8 Intel XE-25 SSD drives behind it, with BBU and WriteBack caching.

vmstat 5

```
r b  swpd  free  buff  cache si so bi bo  in  cs us sy id wa
4 0 875356 1052616 372540 8784584 0 0 13 3320 13162 49545 18 7 75 0
4 0 875356 1070604 372540 8785072 0 0 29 4145 12995 47492 18 7 75 0
3 0 875356 1051384 372544 8785652 0 0 38 5011 13612 55506 22 7 71 0
```

iostat -dx 5

```
Device: rrqm/s  wrqm/s  r/s    w/s  rkB/s   kB/s  avgrq-sz  avgqu-sz  await  svctm  %util
sda      0.00   61.20  1.20  329.20 15.20  4111.20   24.98    0.03   0.09   0.09   3.04
dm-0     0.00    0.00  0.80  390.60 12.80  4112.00   21.08    0.03   0.08   0.07   2.88
```

```
Device: rrqm/s  wrqm/s  r/s    w/s  rkB/s   kB/s  avgrq-sz  avgqu-sz  await  svctm  %util
sda      0.00   65.80  0.60  346.40  9.60  4974.40   28.73    0.04   0.11   0.09   3.20
dm-0     0.00    0.00  0.60  410.80  9.60  4968.80   24.20    0.04   0.10   0.08   3.28
```

```
Device: rrqm/s  wrqm/s  r/s    w/s  rkB/s   kB/s  avgrq-sz  avgqu-sz  await  svctm  %util
sda      0.40   58.20  1.00  308.80 16.00  3320.80   21.54    0.03   0.11   0.10   3.04
dm-0     0.00    0.00  1.40  362.00 16.00  3300.80   18.25    0.04   0.11   0.08   3.04
```

mpstat 5

```
10:36:12 PM CPU %user %nice %sys %iowait %irq %soft %steal %idle intr/s
10:36:17 PM all 18.81 0.05 3.22 0.22 0.24 2.71 0.00 74.75 13247.40
10:36:17 PM 0 19.57 0.00 3.52 0.98 0.20 2.74 0.00 72.99 1939.00
10:36:17 PM 1 18.27 0.00 3.08 0.38 0.19 2.50 0.00 75.58 1615.40
10:36:17 PM 2 19.09 0.20 3.35 0.20 0.39 1.97 0.00 74.80 1615.60
10:36:17 PM 3 17.73 0.00 3.47 0.39 0.39 3.08 0.00 74.95 1615.40
10:36:17 PM 4 18.15 0.00 2.70 0.00 0.39 2.70 0.00 76.06 1615.60
10:36:17 PM 5 19.38 0.00 3.10 0.19 0.39 2.52 0.00 74.42 1615.40
10:36:17 PM 6 18.39 0.00 3.45 0.00 0.19 2.49 0.00 75.48 1615.40
10:36:17 PM 7 19.96 0.20 2.94 0.00 0.00 3.33 0.00 73.58 1615.40
10:36:17 PM 8 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

Premature Conclusion

- ★ As a result of all the above, I conclude that
 - ◆ nothing external to the database is obviously the problem
 - ◆ The system is not virtualized
 - ◆ I expect the database to be able to perform normally.
- ★ What to do next?

What next?

- ★ Try to use a tool to make things easy.
- ★ **Solution:**
 - ◆ use oprofile.

```

root@dbserver:~/percona# oprofile --demangle=smart --symbols --merge tgid `which mysqld`
samples %      image name          app name          symbol name
893793  31.1273  /no-vmlinux          /no-vmlinux      (no symbols)
325733  11.3440  mysqld mysqld Query_cache::free_memory_block(Query_cache_block*)
117732  4.1001  libc                  libc              (no symbols)
102349  3.5644  mysqld mysqld my_hash_sort_bin
76977   2.6808  mysqld mysqld MySQLparse(void*)
71599   2.4935  libpthread            libpthread        pthread_mutex_trylock
52203   1.8180  mysqld mysqld read_view_open_now
46516   1.6200  mysqld mysqld Query_cache::invalidate_query_block_list(THD*, Query_cache_block_table*)
42153   1.4680  mysqld mysqld Query_cache::write_result_data()
37359   1.3011  mysqld mysqld MySQLlex(void*, void*)
35917   1.2508  libpthread            libpthread        __pthread_mutex_unlock_usercnt
34248   1.1927  mysqld mysqld __intel_new_memcpy
33825   1.1780  mysqld mysqld rec_get_offsets_func
25713   0.8955  mysqld mysqld my_pthread_fastmutex_lock
22541   0.7850  mysqld mysqld page_rec_get_n_recs_before
20322   0.7077  mysqld mysqld buf_page_get_gen
19037   0.6630  mysqld mysqld lex_start(THD*)
17818   0.6205  mysqld mysqld Query_cache::free_query(Query_cache_block*)
17509   0.6098  mysqld mysqld btr_search_guess_on_hash
17495   0.6093  mysqld mysqld find_field_in_table_ref(
14224   0.4954  mysqld mysqld build_template(row_prebuilt_struct*, THD*, st_table*, unsigned int)
13575   0.4728  mysqld mysqld query_cache_query_get_key
13308   0.4635  /usr/bin/oprofiled    /usr/bin/oprofiled    (no symbols)
13072   0.4552  mysqld mysqld Protocol::send_fields(List<Item>*, unsigned int)
12615   0.4393  /usr/lib/libperl.so.5.8.8 /usr/lib/libperl.so.5.8.8 (no symbols)
12242   0.4263  mysqld mysqld btr_cur_search_to_nth_level
11880   0.4137  mysqld mysqld page_cur_search_with_match
11343   0.3950  mysqld mysqld my_hash_search

```

..

Solution:

- ★ Start innotop (just to have a realtime monitor)
- ★ Disable query cache.
- ★ Watch QPS change in innotop.

Additional Confirmation

★ The slow query log also confirms queries back to normal

```
tail -f /var/log/slow.log | perl mk-query-digest --run-time 30s --report-format=profile
```

```
# Profile
# Rank Query ID           Response time      Calls R/Call  Item
# =====
# 1 0x5CE5EC5A7CA344DD     2.3601 15.9% 12773 0.0002 SELECT team_member
# 2 0xE1D373DA4E0F4D7A     2.3244 15.6%  9488 0.0002 SELECT tg_user
# 3 0x950A5CF5173D3022     1.9800 13.3%  5693 0.0003 SELECT namespace_member
# 4 0x02B7087599A7C6BB     1.7745 11.9%  5662 0.0003 SELECT namespace_p?p_key
# 5 0x6D26A1663AE2F07A     1.6751 11.3%  7266 0.0002 SELECT host
# 6 0x75960C3BD6637C00     1.1919  8.0%  5318 0.0002 SELECT host
# 7 0x813031B8BBC3B329     1.1193  7.5%  8545 0.0001 COMMIT
# 8 0x0262228C76E3BDFD     0.9228  6.2%  5408 0.0002 SELECT pref
# 9 0x5B0232CD0D7A122F     0.3382  2.3%  1879 0.0002 SELECT namespace_member
# 10 0xFB44D5AA1D96A090     0.1700  1.1%  1142 0.0001 SELECT namespace_member
# 11 0xC83E431FCADB7E4B     0.1539  1.0%   850 0.0002 SELECT team_member
# 12 0x19C8068B5C1997CD     0.1464  1.0%  9637 0.0000 ROLLBACK
# 13 0x46ED81A7F2B93617     0.1381  0.9%   690 0.0002 UPDATE tg_user
# 14 0x010D1348A9CC32EC     0.1373  0.9%   846 0.0002 SELECT namespace
# 15 0xC5FF324E9F0795CB     0.1195  0.8%   544 0.0002 SELECT namespace_member
# 16 0xCCE9F94F19CB7DA2     0.1144  0.8%   673 0.0002 SELECT namespace
# 17 0xB269C2A859F7F1AE     0.1074  0.7%   561 0.0002 SELECT namespace
# 18 0x943798A09019B333     0.0984  0.7%  5315 0.0000 SHOW WARNINGS
```

Case Study #2

Information Provided

- ★ About 4PM on Saturday, queries suddenly began taking insanely long to complete
 - ◆ From sub-ms to many minutes.
 - ◆ As far as the customer knew, nothing had changed.
 - ◆ Nobody was at work.
 - ◆ They had disabled selected apps where possible to reduce load.

Overview

- ★ They are running 5.0.77-percona-highperf-b13.
- ★ The server has an EMC SAN
 - ◆ with a RAID5 array of 5 disks, and LVM on top of that
 - ◆ Server has 2 quad-core CPUSXeon L5420 @ 2.50GHz.
 - ◆ No virtualization.
- ★ They tried restarting mysqld
 - ◆ It has 64GB of RAM, so it's not warm yet.

Train of thought

- ★ The performance drop is way too sudden and large.
 - ◆ On a weekend, when no one is working on the system.
 - ◆ Something is seriously wrong.
 - ◆ Look for things wrong first.

Elimination of easy possibilities:

- ★ First, confirm that queries are actually taking a long time to complete.
 - ◆ They all are, as seen in processlist.
- ★ Check the SAN status.
 - ◆ They checked and reported that it's not showing any errors or failed disks.

Investigation of the obvious:

- ★ Server's incremental status variables don't look amiss
- ★ 150+ queries in commit status.
- ★ Many transactions are waiting for locks inside InnoDB
 - ◆ But no semaphore waits, and main thread seems OK.
- ★ iostat and vmstat at 5-second intervals:
 - ◆ Suspicious IO performance and a lot of iowait
 - ◆ But virtually no work being done.

iostat

```
Device: rrqm/s  wrqm/s    r/s    w/s  rsec/s  wsec/s  avgrq-sz  avgqu-sz  await  svctm   %util
sda      0.00    0.00    0.00   0.00   0.00    0.00    0.00     0.00    0.00   0.00    0.00
sdb      0.00   49.00  10.00 104.00 320.00 8472.00  77.12     2.29   20.15  8.78 100.10
sdb1     0.00   49.00  10.00 104.00 320.00 8472.00  77.12     2.29   20.15  8.78 100.10
sdc      0.00   17.00   0.00   6.00   0.00  184.00  30.67     0.00    0.00   0.00   0.00
sdc1     0.00    0.00   0.00   0.00   0.00   0.00   0.00     0.00    0.00   0.00   0.00
sdc2     0.00    0.00   0.00   0.00   0.00   0.00   0.00     0.00    0.00   0.00   0.00
sdc3     0.00    0.00   0.00   0.00   0.00   0.00   0.00     0.00    0.00   0.00   0.00
sdc4     0.00    0.00   0.00   0.00   0.00   0.00   0.00     0.00    0.00   0.00   0.00
sdc5     0.00   17.00   0.00   6.00   0.00  184.00  30.67     0.00    0.00   0.00   0.00
dm-0     0.00    0.00   0.00  23.00   0.00  184.00   8.00     0.00    0.00   0.00   0.00
dm-1     0.00    0.00   0.00   0.00   0.00   0.00   0.00     0.00    0.00   0.00   0.00
dm-2     0.00    0.00   9.00 152.00 288.00 7920.00 50.98     3.47   21.61  6.21 100.00
```

vmstat

```
r b swpd      free      buff      cache si so  bi   bo   in    cs us sy id wa st
5 1  176 35607308 738468 19478720 0 0   48  351    0     0  1  0 96  3  0
0 1  176 35605912 738472 19478820 0 0  560  848 2019 2132  4  1 83 13  0
0 2  176 35605788 738480 19479048 0 0  608  872 2395 2231  0  1 85 14  0
0 1  176 35604664 738484 19479128 0 0  688 1692 2082 1785  0  0 85 15  0
1 2  176 35604540 738496 19479436 0 0  528  876 2513 2311  0  0 84 15  0
1 2  176 35604076 738500 19479484 0 0  480 1092 1962 1684  0  0 84 16  0
1 1  176 35603084 738500 19479572 0 0  624  808 1888 1635  0  0 84 16  0
1 2  176 35602348 738500 19479608 0 0  704  792 2014 1729  1  0 84 15  0
1 1  176 35601604 738504 19479704 0 0  496 1116 2140 1910  0  0 85 15  0
1 1  176 35601140 738508 19479736 0 0  464  896 2116 1927  0  0 85 14  0
1 3  176 35599900 738508 19479908 0 0 1328 1020 2083 1869  0  1 83 17  0
1 3  176 35596660 738508 19479944 0 0 1792  696 1855 1754  1  1 81 17  0
1 3  176 35594496 738512 19480028 0 0 1732  776 2016 1848  1  0 81 18  0
```

From vmstat/iostat:

- ★ It looks like something is blocking commits
- ★ Likely to be either a serious bug (a transaction that has gotten the commit mutex and is hung?) or a hardware problem.
- ★ IO unreasonably slow, so that is probably the problem.

Analysis

- ★ Because the system is not "doing anything,"
 - ◆ profiling where CPU time is spent is probably useless.
 - ◆ We already know that it's spent waiting on mutexes in the commit problem, so oprofile will probably show nothing.
 - ◆ Other options that come to mind:
 - profile IO calls with strace -c
 - benchmark the IO system, since it seems to be suspicious.
- ★ But first, a bit more investigation.

Stack Dump

```
[root@db203 ~]# pmp stacktrace
```

```
154 threads with the following stack trace:
```

```
#0 0x000000359920ce74 in __lll_lock_wait () from /lib64/libpthread.so.0
#1 0x00000035992088e0 in _L_lock_1167 () from /lib64/libpthread.so.0
#2 0x0000003599208839 in pthread_mutex_lock () from /lib64/libpthread.so.0
#3 0x0000000000062d9ed in innobase_xa_prepare (thd=0x2ab438cf5960, all=true)
at ha_innodb.cc:7577
#4 0x0000000000061d64e in ha_commit_trans (thd=0x2ab438cf5960, all=true) at
handler.cc:706
```

```
1 threads with the following stack trace:
```

```
#0 0x000000359920dde8 in pread64 () from /lib64/libpthread.so.0
```

```
1 threads with the following stack trace:
```

```
#0 0x00000035986cc837 in fdatsync () from /lib64/libc.so.6
#1 0x000000000007d5a5f in my_sync (fd=12, my_flags=16) at my_sync.c:52
#2 0x000000000005e401e in MYSQL_LOG::flush_and_sync (this=<value optimized
out>) at log.cc:1819
```

Oprofile

★ As expected: nothing useful in oprofile

samples	%	symbol name
6331	15.3942	buf_calc_page_new_checksum
2008	5.1573	sync_array_print_long_waits
2004	4.8728	MYSQLparse(void*)
1724	4.1920	srv_lock_timeout_and_monitor_thread
1441	3.5039	rec_get_offsets_func
1098	2.6698	my_utf8_uni
780	1.8966	mem_pool_fill_free_list
762	1.8528	my_strnncollsp_utf8
682	1.6583	buf_page_get_gen
650	1.5805	MYSQLlex(void*, void*)
604	1.4687	btr_search_guess_on_hash
566	1.3763	read_view_open_now

strace -c

- ★ Nothing relevant after 30 seconds or so.

```
[root@db203 ~]# strace -cp 24078
Process 24078 attached - interrupt to quit
Process 24078 detached%
time          seconds          usecs/call         calls      errors  syscall
100.00        0.098978          14140              7           0       select
 0.00         0.000000           0                  7           0       accept
 0.00         0.000000           0                  7           0       getsockname
 0.00         0.000000           0                 14           0       setsockopt
 0.00         0.000000           0                   2           0       clone
 0.00         0.000000           0                 35           0       fcntl
 0.00         0.000000           0                 10           0       futex
```

Examine history

- ★ Look at 'sar' for historical reference.
- ★ Ask the client to look at their graphs to see if there are obvious changes around 4PM.

SAR around 4pm

04:00:01	PM	CPU	%user	%nice	%system	%iowait	%steal	%idle
04:00:01	PM	all	0.73	0.00	0.43	5.33	0.00	93.51
04:10:01	PM	all	0.71	0.00	0.41	5.03	0.00	93.85
04:20:01	PM	all	0.68	0.00	0.39	4.76	0.00	94.17
04:30:01	PM	all	0.71	0.00	0.39	6.51	0.00	92.39
04:40:01	PM	all	0.42	0.00	0.22	16.44	0.00	82.92
04:50:01	PM	all	0.45	0.00	0.24	15.87	0.00	83.45
05:00:01	PM	all	0.49	0.00	0.25	15.81	0.00	83.45
05:10:01	PM	all	0.47	0.00	0.25	15.90	0.00	83.38
05:20:01	PM	all	0.46	0.00	0.24	15.77	0.00	83.53
05:30:01	PM	all	0.45	0.00	0.24	16.02	0.00	83.29

04:00:01	PM	tps	rtps	wtps	bread/s	bwrtn/s
04:00:01	PM	1211.86	101.74	1110.12	4137.71	28573.15
04:10:01	PM	1143.72	96.40	1047.33	3838.95	27059.94
04:20:01	PM	1088.95	92.68	996.27	3817.55	25423.51
04:30:01	PM	1081.20	91.65	989.55	3752.29	25487.12
04:40:01	PM	452.65	54.85	397.80	2633.19	8366.46
04:50:01	PM	511.75	52.75	459.00	2494.71	12460.27
05:00:01	PM	516.54	53.59	462.95	2515.42	10101.05
05:10:01	PM	517.63	54.63	463.01	2553.41	10248.53
05:20:01	PM	509.73	53.60	456.13	2568.57	11770.04
05:30:01	PM	515.03	58.53	456.50	2799.31	10294.01

Observations

- ★ writes dropped dramatically around 4:40
- ★ at the same time iowait increased a lot
- ★ corroborated by the client's graphs
- ★ points to decreased performance of the IO subsystem
- ★ SAN attached by fibre channel, so it could be
 - ◆ this server
 - ◆ the SAN
 - ◆ the connection
 - ◆ the specific device on the SAN.

Elimination of Options:

- ★ Benchmark /dev/sdb1 and see if it looks reasonable.
- ★ This box or the SAN?
 - ◆ check the same thing from another server.
- ★ Tool: use iotop with the -I flag (O_DIRECT).
- ★ The result was 54 writes per second on the first iteration
 - ◆ canceled it after that because that took so long.

Pay Dirt!

- ★ Before I could repeat, customer said RAID failed after all
- ★ Moral of the story: information != facts
- ★ Customer's web browser had cached SAN status page!

Case Study #3

Information from the start:

- ★ Sometimes (once every day or two) the server starts to reject connections with a max_connections error.
- ★ This lasts from 10 seconds to a couple of minutes and is sporadic.
- ★ Server specs:
 - ◆ 16 cores
 - ◆ 12GB of RAM, 900MB data
 - ◆ Data on Intel XE-25 SSD
 - ◆ Running MySQL 5.1 with InnoDB Plugin

Train of Thought

- ★ Pile-ups cause long queue waits?
 - ◆ thus incoming new connections exceed max_connections?
- ★ Pile-ups can be
 - ◆ the query cache
 - ◆ InnoDB mutexes
 - ◆ et cetera...

Elimination

- ★ There are no easy possibilities.
- ★ We'd previously worked with this client and the DB wasn't the problem then.
- ★ Queries aren't perfect, but are still running in less than 10ms normally.

Investigation

- ★ Nothing is obviously wrong.
- ★ Server looks fine in normal circumstances.

Analysis

- ★ We are going to have to capture server activity when the problem happens.
- ★ We can't do anything without good diagnostic data.
- ★ Decision: install 'collect' (from Aspersa) and wait.

Analysis (cont.)

- ★ After several pile-ups nothing very helpful was gathered
 - ◆ But then we got a good one
 - ◆ This took days/a week
- ★ Result of diagnostics data: too much information!

During the Freeze

- ★ Connections increased from normal 5-15 to over 300.
- ★ QPS was about 1-10k.
 - ◆ Lots of Com_admin_commands.
 - ◆ Vast majority of "real" queries are Com_select (300-2000 per second)
 - ◆ There are only 5 or so Com_update, other Com_ are zero.

During the Freeze (cont.)

- ★ No table locking.
- ★ Lots of query cache activity, but normal-looking.
 - ◆ no lowmem_prunes.
- ★ 20 to 100 sorts per second
 - ◆ between 1k and 12k rows sorted per second.

During the Freeze (cont.)

- ★ Between 12 and 90 temp tables created per second
 - ◆ about 3 to 5 of them created on disk.
- ★ Most queries doing index scans or range scans - not full table scans or cross joins.
- ★ InnoDB operations are just reads, no writes.
- ★ InnoDB doesn't write much log or anything.

During the Freeze (cont.)

★ InnoDB status:

- ◆ InnoDB main thread was in "flushing buffer pool pages" and there were basically no dirty pages.
- ◆ Most transactions were waiting in the InnoDB queue.
"12 queries inside InnoDB, 495 queries in queue"
- ◆ The log flush process was caught up.
- ◆ The InnoDB buffer pool wasn't even close to being full (much bigger than the data size).

During the Freeze (cont.)

- ★ There were mostly 2 types of queries in SHOW PROCESSLIST, most of them in the following states:
 - ♦ `$ grep State: status-file | sort | uniq -c | sort -nr`
 - 161 State: Copying to tmp table
 - 156 State: Sorting result
 - 136 State: statistics

iostat

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	avgrq-sz	avgqu-sz	await	svctm	%util
sda3	0.04	493.63	0.65	15.49	142.18	4073.09	261.18	0.17	10.68	1.02	1.65
sda3	0.00	8833.00	1.00	500.00	8.00	86216.00	172.10	5.05	11.95	0.59	29.40
sda3	0.00	33557.00	0.00	451.00	0.00	206248.00	457.31	123.25	238.00	1.90	85.90
sda3	0.00	33911.00	0.00	565.00	0.00	269792.00	477.51	143.80	245.43	1.77	100.00
sda3	0.00	38258.00	0.00	649.00	0.00	309248.00	476.50	143.01	231.30	1.54	100.10
sda3	0.00	34237.00	0.00	589.00	0.00	281784.00	478.41	142.58	232.15	1.70	100.00
sda3	0.00	11029.00	0.00	384.00	0.00	162008.00	421.90	71.80	238.39	1.73	66.60
sda3	0.00	36.00	0.00	14.00	0.00	400.00	28.57	0.01	0.93	0.36	0.50
sda3	0.00	18.00	0.00	13.00	0.00	248.00	19.08	0.01	0.92	0.23	0.30
sda3	0.00	38.00	0.00	13.00	0.00	408.00	31.38	0.01	0.92	0.23	0.30
sda3	0.00	15.00	0.00	13.00	0.00	224.00	17.23	0.00	0.15	0.15	0.20
sda3	0.00	40.00	0.00	16.00	0.00	448.00	28.00	0.01	0.50	0.19	0.30
sda3	0.00	19.00	0.00	12.00	0.00	248.00	20.67	0.01	0.42	0.17	0.20
sda3	0.00	17.00	0.00	14.00	0.00	248.00	17.71	0.01	0.36	0.21	0.30
sda3	0.00	22.00	0.00	12.00	0.00	272.00	22.67	0.00	0.17	0.17	0.20
sda3	0.00	734.00	0.00	131.00	0.00	6920.00	52.82	0.43	3.31	0.21	2.70
sda3	0.00	30.00	0.00	16.00	0.00	368.00	23.00	0.01	0.50	0.12	0.20
sda3	0.00	18.00	0.00	12.00	0.00	240.00	20.00	0.01	0.83	0.17	0.20
sda3	0.00	35.00	0.00	15.00	0.00	400.00	26.67	0.01	0.93	0.20	0.30
sda3	0.00	11.00	0.00	11.00	0.00	176.00	16.00	0.00	0.27	0.09	0.10
sda3	0.00	22.00	0.00	14.00	0.00	288.00	20.57	0.00	0.21	0.21	0.30
sda3	0.00	146.00	0.00	405.00	0.00	4408.00	10.88	1.71	4.22	0.08	3.30
sda3	0.00	20.00	0.00	13.00	0.00	264.00	20.31	0.01	0.54	0.15	0.20
sda3	0.00	13418.00	0.00	108.00	0.00	45576.00	422.00	23.98	70.35	1.59	17.20
sda3	0.00	31233.00	0.00	513.00	0.00	238480.00	464.87	125.17	219.29	1.95	100.00
sda3	0.00	19725.00	0.00	483.00	0.00	239784.00	496.45	124.55	318.01	2.03	98.10
sda3	0.00	62.00	0.00	19.00	0.00	648.00	34.11	0.02	1.00	0.16	0.30

vmstat

```
 r b  swpd    free    buff    cache  si  so  bi    bo    in    cs  us  sy  id  wa  st
50 2  86064 1186648 3087764 4475244  0  0  5     138     0     0  1  1 98  0  0
13 0  86064 1922060 3088700 4099104  0  0  4    37240 312832 50367 25 39 34  2  0
 2 5  86064 2676932 3088812 3190344  0  0  0   136604 116527 30905  9 12 71  9  0
 1 4  86064 2782040 3088812 3087336  0  0  0   153564  34739 10988  2  3 86  9  0
 0 4  86064 2871880 3088812 2999636  0  0  0   163176  22950  6083  2  2 89  8  0
 0 4  86064 3002924 3088812 2870352  0  0  0   131532  32138  9234  3  2 87  7  0
 0 0  86064 3253988 3088836 2794932  0  0  0    29664  34756 11057  3  4 91  3  0
 0 0  86064 3254104 3088860 2794604  0  0  0     200  24995  9419  1  1 97  0  0
 0 0  86064 3255184 3088900 2794772  0  0  0     124  29767 10042  3  2 95  0  0
 2 0  86064 3254660 3088900 2794840  0  0  0     204  12570  4181  2  1 98  0  0
 1 0  86064 3254692 3088900 2794856  0  0  0     112  12447  3374  1  1 98  0  0
 0 0  86064 3254556 3088912 2794876  0  0  0     224  22128  7584  2  2 97  0  0
 1 0  86064 3255020 3088912 2794920  0  0  0     124  12875  3422  1  1 98  0  0
 0 0  86064 3254952 3088912 2794936  0  0  0     124  15209  4333  1  1 98  0  0
 0 0  86064 3255100 3088912 2794960  0  0  0     136  13568  4351  1  1 98  0  0
 0 0  86064 3255120 3088912 2794980  0  0  0    3460  19657  5690  2  1 97  0  0
 1 0  86064 3254488 3088912 2794996  0  0  0     184  31300  7393  5  2 94  0  0
 0 0  86064 3255488 3088912 2795116  0  0  0     120  22892  6468  3  1 96  0  0
 0 0  86064 3255080 3088936 2795136  0  0  0     200  21948  6303  3  1 96  0  0
 2 0  86064 3255204 3088936 2795160  0  0  0      88  15222  4805  2  1 98  0  0
 1 0  86064 3255896 3088936 2795176  0  0  0     144  20555  5956  2  1 97  0  0
 0 0  86064 3254596 3088936 2795188  0  0  0    2204  18818  5079  2  1 95  2  0
 4 0  86064 3255560 3088936 2795228  0  0  0     132  24550  6266  3  2 95  0  0
 1 4  86064 3011800 3088952 3029380  0  0  0    70528  38483 10295  4  4 89  3  0
 0 2  86064 3169196 3088956 2877628  0  0  0   143468  49020  9422  4  3 83  9  0
 2 0  86064 3254888 3089028 2795476  0  0  0    47924  29703  7856  2  2 90  6  0
 2 0  86064 3254912 3089028 2795512  0  0  0     324  27352  7536  3  2 95  0  0
```

lostats, formatted incrementally:

m	m	dev	reads	rd_mrg	rd_sectors	ms_reading	writes	wr_mrg	wr_sectors	ms_writing	cur_ios	ms_doing_io	ms_wghdt
0	0	sda3	1	0	8	51	498	8833	85192	5871	-28	292	4993
0	0	sda3	0	0	0	0	658	44808	304056	144370	130	1232	176432
0	0	sda3	0	0	0	0	569	34133	269472	155917	13	1005	144815
0	0	sda3	0	0	0	0	725	42361	349696	146777	-6	1004	143371
0	0	sda3	0	0	0	0	518	29256	239008	139677	-8	1005	145328
0	0	sda3	0	0	0	0	168	434	66848	37659	-129	280	14491
0	0	sda3	0	0	0	0	14	36	400	13	0	5	13
0	0	sda3	0	0	0	0	13	18	248	12	0	3	12
0	0	sda3	0	0	0	0	13	38	408	12	0	3	12
0	0	sda3	0	0	0	0	13	15	224	2	0	2	2
0	0	sda3	0	0	0	0	16	40	448	8	0	3	8
0	0	sda3	0	0	0	0	12	19	248	5	0	2	5
0	0	sda3	0	0	0	0	14	17	248	5	0	3	5
0	0	sda3	0	0	0	0	12	22	272	2	0	2	2
0	0	sda3	0	0	0	0	131	734	6920	434	0	27	434
0	0	sda3	0	0	0	0	16	30	368	8	0	2	8
0	0	sda3	0	0	0	0	12	18	240	10	0	2	10
0	0	sda3	0	0	0	0	15	35	400	14	0	3	14
0	0	sda3	0	0	0	0	11	11	176	3	0	1	3
0	0	sda3	0	0	0	0	398	143	4328	1703	0	34	1703
0	0	sda3	0	0	0	0	21	25	368	8	0	2	8
0	0	sda3	0	0	0	0	13	20	264	7	0	2	7
0	0	sda3	0	0	0	0	430	26860	194664	89081	48	766	99648
0	0	sda3	0	0	0	0	582	37453	284544	159783	41	1264	167989
0	0	sda3	0	0	0	0	92	63	44632	24832	-89	123	6059
0	0	sda3	0	0	0	0	19	62	648	19	0	3	19
0	0	sda3	0	0	0	0	96	510	4848	182	0	21	182
0	0	sda3	0	0	0	0	13	19	256	12	0	2	12
0	0	sda3	0	0	0	0	16	21	296	15	0	2	15

Oprofile

samples	%	image name	app name	symbol name
473653	63.5323	no-vmlinux	no-vmlinux	/no-vmlinux
95164	12.7646	mysqld	mysqld	/usr/libexec/mysqld
53107	7.1234	libc-2.10.1.so	libc-2.10.1.so	memcpy
13698	1.8373	ha_innodb.so	ha_innodb.so	build_template()
13059	1.7516	ha_innodb.so	ha_innodb.so	btr_search_guess_on_hash
11724	1.5726	ha_innodb.so	ha_innodb.so	row_sel_store_mysql_rec
8872	1.1900	ha_innodb.so	ha_innodb.so	rec_init_offsets_comp_ordinary
7577	1.0163	ha_innodb.so	ha_innodb.so	row_search_for_mysql
6030	0.8088	ha_innodb.so	ha_innodb.so	rec_get_offsets_func
5268	0.7066	ha_innodb.so	ha_innodb.so	cmp_dtuple_rec_with_match

Analysis:

- ★ There is a lot of data here
- ★ most of it points to nothing in particular except "need more research."
 - ◆ For example, in oprofile, what does build_template() do in InnoDB?
 - ◆ Why is memcpy() such a big consumer of time?
 - ◆ What is hidden within the 'mysqld' image/symbol?
- ★ We could spend a lot of time on these things.

Analysis (cont.)

- ★ In looking for things that just don't make sense, the iostat data is very strange.
- ★ We can see hundreds of MB per second written to disk for sustained periods
- ★ but there isn't even that much data in the whole database.
- ★ So clearly this can't simply be InnoDB's "furious flushing" problem

Analysis (cont.)

- ★ Virtually no reading from disk is happening in this period of time.
- ★ Raw disk stats show that all the time is consumed in writes.
- ★ There is an enormous queue on the disk.

Analysis (cont.)

- ★ There was no swap activity, and 'ps' (not shown) confirmed that nothing else significant was happening.
- ★ 'df -h' and 'lsdf' (not shown) showed that:
 - ◆ mysqld's temp files became large
 - ◆ disk free space was noticeably changed while this pattern happened.
- ★ So mysqld was writing GB to disk in short bursts.

Analysis (cont.)

- ★ Although this is not fully instrumented inside of MySQL, we know that
 - ◆ MySQL only writes data, logs, sort, and temp tables to disk.
 - ◆ Thus, we can eliminate data and logs.
- ★ Discussion with developers revealed that some kinds of caches could expire and cause a stampede on the database.

Conclusion

- ★ Based on reasoning and knowledge of internals: it is likely that poorly optimized queries are causing a storm of very large temp tables on disk.

Plan of Attack

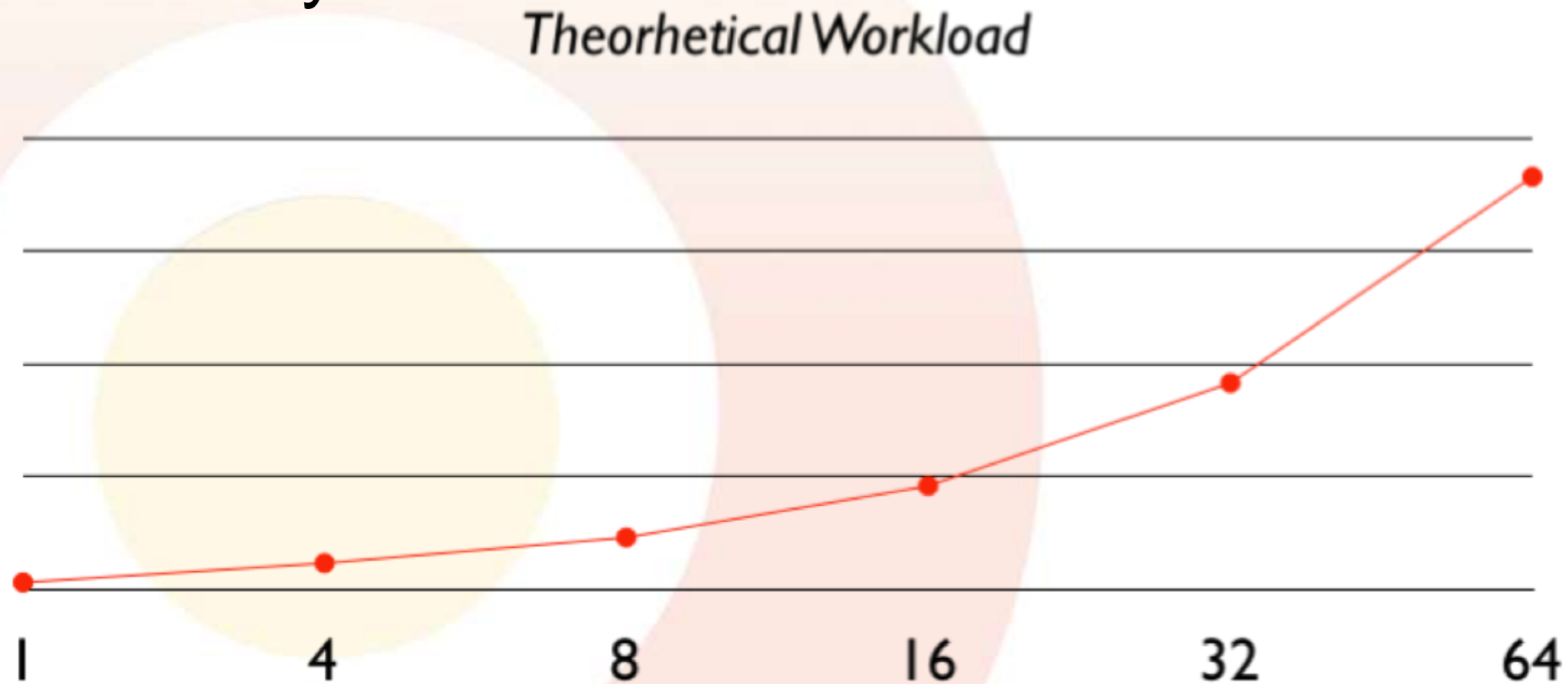
- ★ Optimize the 2 major kinds of queries found in SHOW PROCESSLIST so they don't use temp tables on disk.
- ★ These queries are fine in isolation, but when there is a rush on the database, can pile up.
- ★ Problem resolved after removing temporary tables on disk

Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

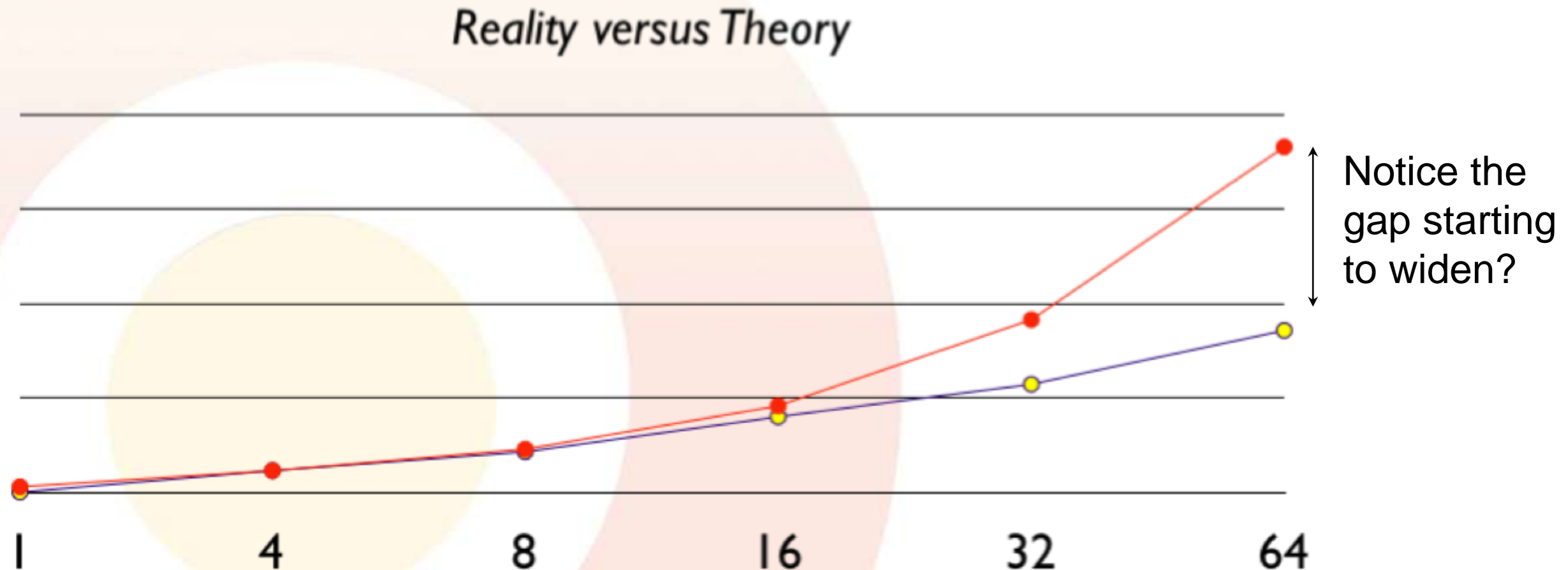
CPU Scalability

- ★ **[Perfect World]** As we add CPUs we get a linear throughput increase, provided we have sufficient concurrency:



CPU Scalability

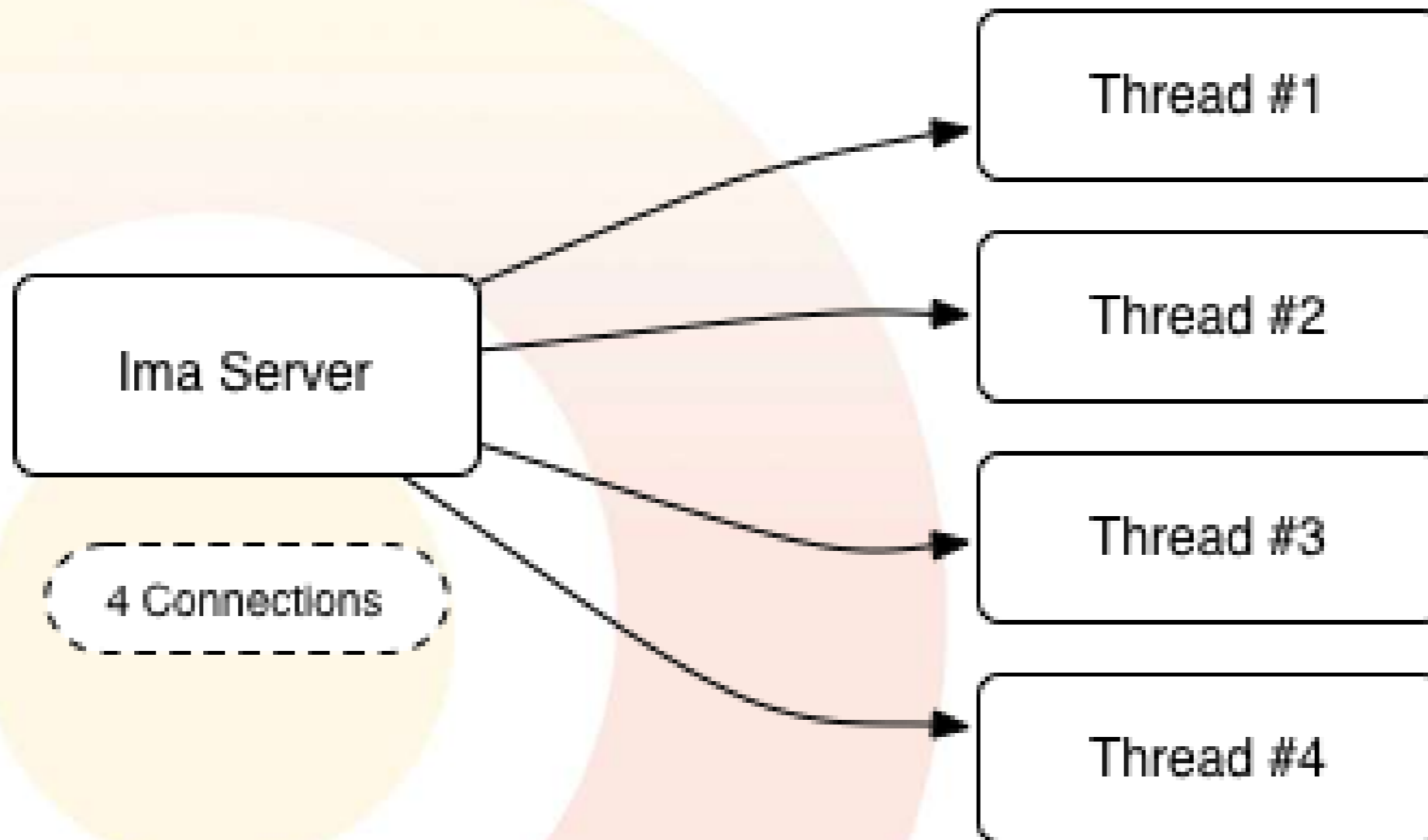
★ **[Reality]** We never quite follow the theoretical curve:



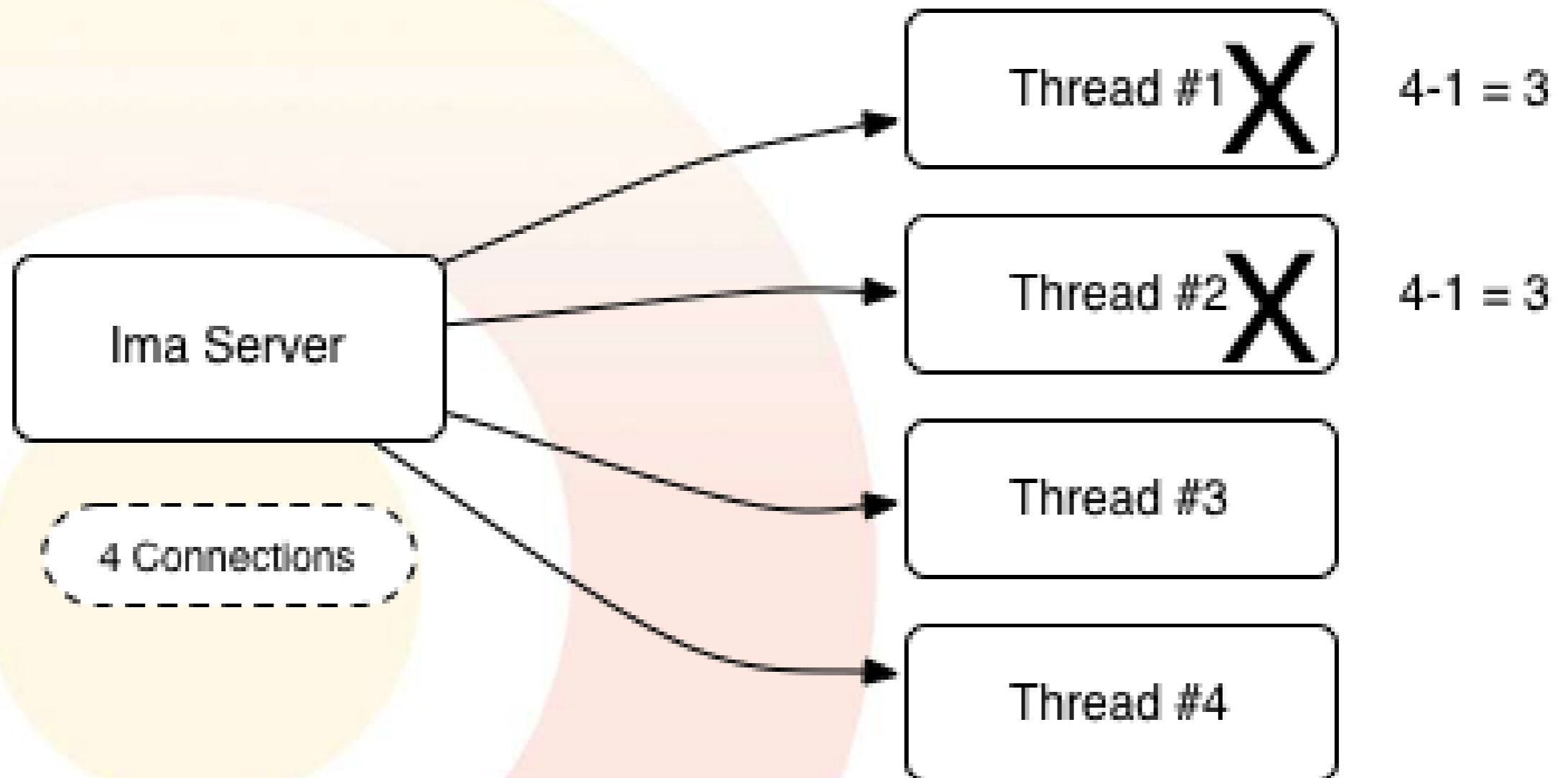
Mutex Contention.

This is the *most* likely reason.

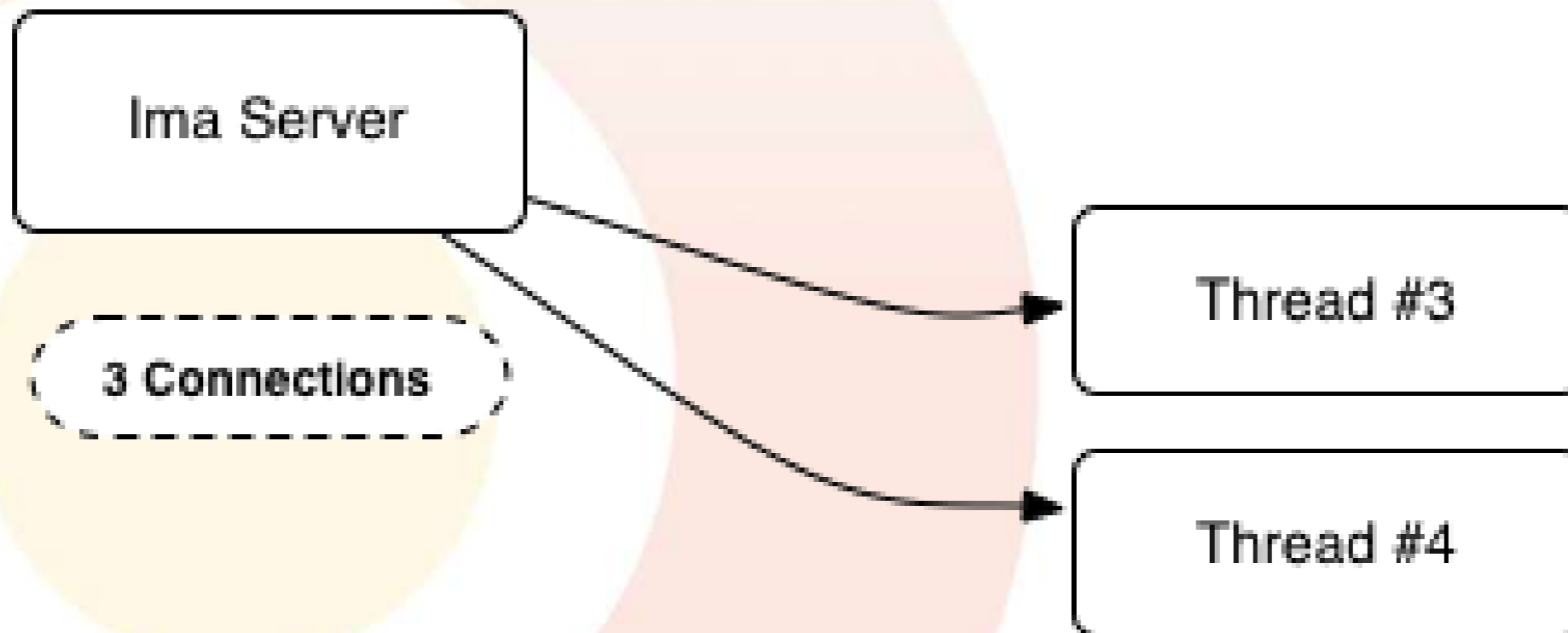
What's a Mutex?



What's a Mutex? (cont.)

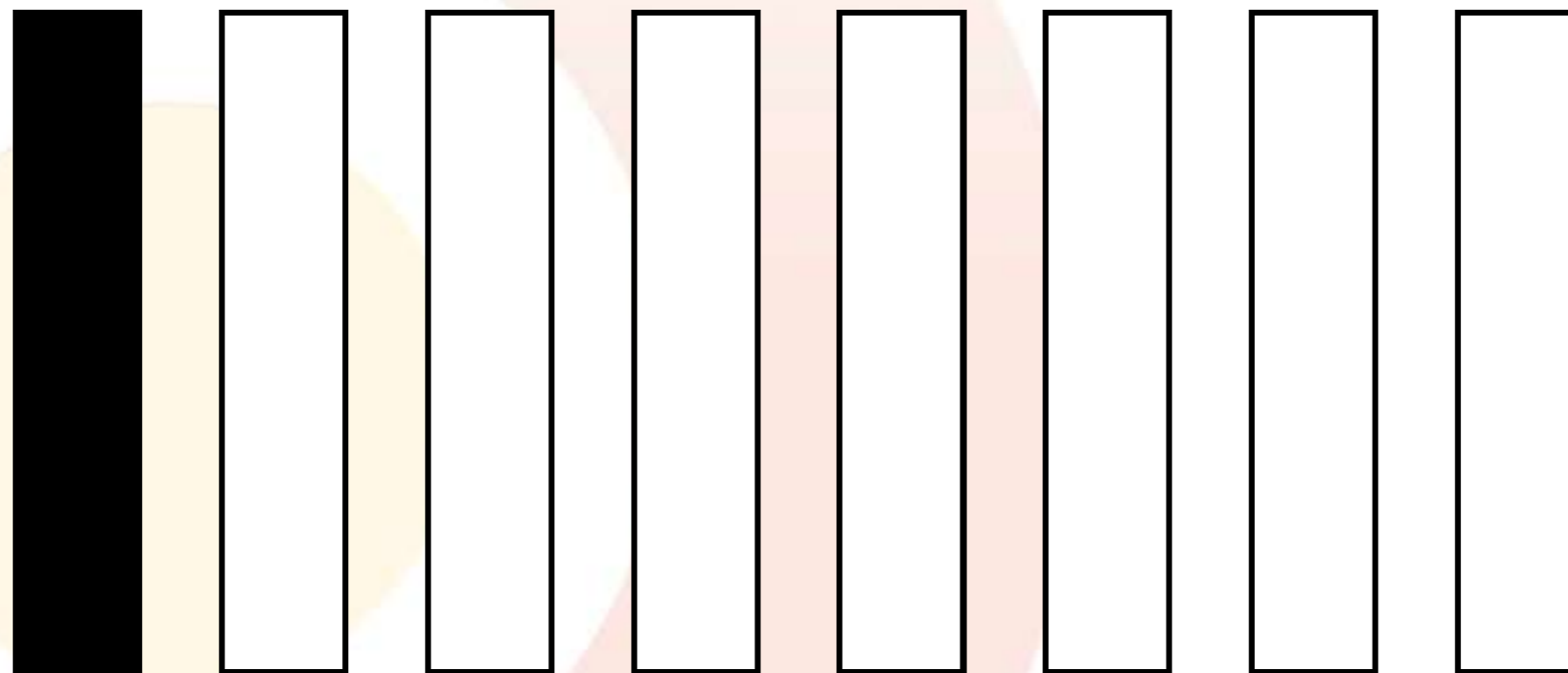


What's a Mutex? (cont.)



Mutexes become hotspots

- ★ The longer the mutex is held, the more likely you can hold up other tasks - and reduce CPU scalability:



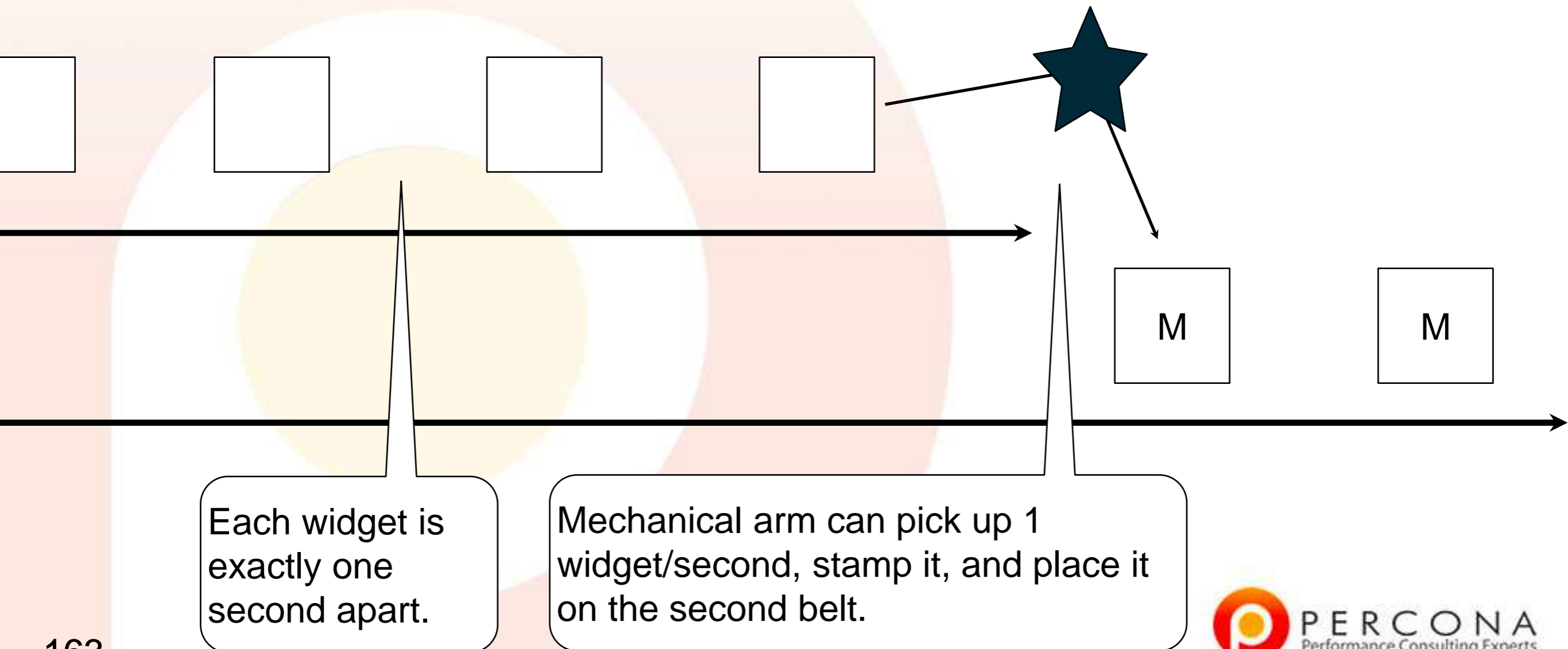
CPU's in use

Mutex contention might look like:

```
10:36:12 PM CPU %user %nice %sys %iowait %irq %soft %steal %idle intr/s
10:36:17 PM all 18.81 0.05 3.22 0.22 0.24 2.71 0.00 74.75 13247.40
10:36:17 PM 0 19.57 0.00 3.52 0.98 0.20 2.74 0.00 72.99 1939.00
10:36:17 PM 1 18.27 0.00 3.08 0.38 0.19 2.50 0.00 75.58 1615.40
10:36:17 PM 2 19.09 0.20 3.35 0.20 0.39 1.97 0.00 74.80 1615.60
10:36:17 PM 3 17.73 0.00 3.47 0.39 0.39 3.08 0.00 74.95 1615.40
10:36:17 PM 4 18.15 0.00 2.70 0.00 0.39 2.70 0.00 76.06 1615.60
10:36:17 PM 5 19.38 0.00 3.10 0.19 0.39 2.52 0.00 74.42 1615.40
10:36:17 PM 6 18.39 0.00 3.45 0.00 0.19 2.49 0.00 75.48 1615.40
10:36:17 PM 7 19.96 0.20 2.94 0.00 0.00 3.33 0.00 73.58 1615.40
10:36:17 PM 8 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
```

Second thing to know about tasks:

- ★ Not all tasks arrive on time. Take the following example of a manufacturing process:



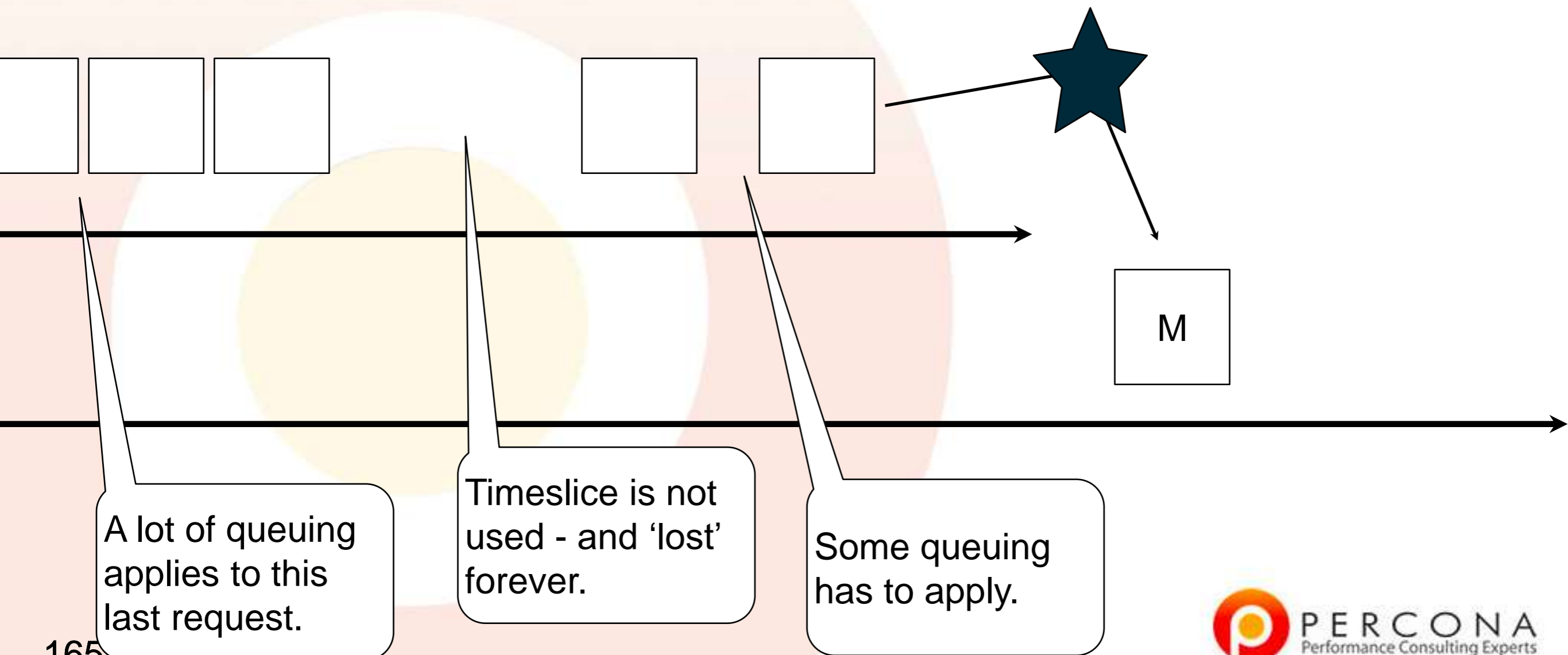
Throughput Question:

- ★ There is only one mechanical arm - no parallelism is possible.
- ★ Service time of the mechanical arm is 1 second.
- ★ Maximum capacity is 60 boxes/minute.
- ★ Can we have a throughput of 60 boxes/minute and a response time of 1 second?

In this example we can. But only because we control the arrival rate of the widgets.

Important Real-Life Difference

- ★ The arrival rate of requests is not evenly distributed:



So there are some lessons:

- ★ If you have random arrivals - you may not be able to reach capacity **and** have an acceptable response time.
- ★ All CPUs hitting 100% may never happen.
- ★ Just because you don't see CPUs hitting 100% it does not mean that you do not have a problem.
 - ◆ There may still be a response time impact.

Tip: Never max out

- ★ Keep some capacity free.
- ★ Our experience:
 - ◆ You'll have trouble exceeding ~75% of CPU usage and getting respectable response.

Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

Utilization

- ★ Utilization means how much of the time the resource is busy - it has a ceiling of 100%.

Let me repeat again:
Utilization has a ceiling of 100%.

Utilization (cont.)

- ★ This makes it unreliable to evaluate *demands* on a system - because load can increase to infinity - utilization is fixed at 100%.
- ★ When utilization is at 100%, a backlog is started.

Load avg

- ★ Unix load average is an example of a backlog:
 - ◆ shell> uptime
 - ◆ 17:22 up 11 days, 19:20, 6 users, load averages: 0.31
0.32 0.33
- ★ The value here is simply the number of runnable tasks that are waiting to be serviced. It's a weighted moving average. It can go to infinity.
- ★ On Linux also includes processed blocked on IO

iostat

- ★ Has both utilization and average queue length.
- ★ The queue length is an interesting measurement of backlog.
- ★ It's normally perfectly safe if we have 100% utilization.
 - ◆ A large backlog is a different question.

Backlog implies *queuing*

- ★ Which means that it has a measurable effect on response time.

Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

Resources that matter:

- ★ Network, storage, cpu, memory, load balancer, connection pool.
- ★ In all cases, think about whether the resource is dedicated or shared/virtualized.

Network

- ★ Packet loss.
- ★ Faulty device (shows up in ifconfig).
- ★ Poor bandwidth.
- ★ High latency.
- ★ Congestion.

Filesystem

- ★ **Serialization.**
- ★ **Maintenance Operations (fsck speed?)**
- ★ **Mount options.**

Disks & Raid

- ★ Speed, size: how fast should this disk be able to work?
- ★ Configuration: RAID5, RAID10?
- ★ Raid controller: BBU? WriteBack policy? Good quality, or generic 3ware junk?

SAN and NFS

- ★ SAN: over NFS? Benchmark with iotop -l to measure real performance, don't be surprised to see terrible random-read performance.
- ★ Is the NFS over a dedicated network?
- ★ SSD: not a magic bullet.

CPUs

- ★ What model? How fast should it be? Generally if the CPU has any problem, *nothing* works
- ★ New server slower than the old server?
 - ◆ Check cache size and cache alignment (/proc/cpuinfo).

CPUs (cont.)

- ★ Performance is unpredictable?
 - ◆ We've found most virtualized technologies have high burstable speed, and many people under-provision not understanding the unpredictable minimums.

Memory

- ★ Can use memtest86+ to test and benchmark it.
- ★ Easy to misconfigure what your vendor needs you to do for best performance.
 - ◆ Best to test before production.

Load Balancer

- ★ It adds more latency, measure with tcpdump on both sides of the load balancer to see if there is a difference.
- ◆ how does the app use it? could it be misconfigured, holding connections open for too long?

Table Of Contents

0. Welcome	5. Your Toolchest
1. Defining Performance	6. CPUs and Tasks
2. The Stack at 10000 Feet	7. Utilization <i>versus</i> Backlog
3. Isolating Problems	8. External-to-the-database Problems
4. Approaching the Stack	9. Conclusion

Topics we didn't cover

(and
shameless
plugs)

Summary tables and materialized views with Flexviews

<http://en.oreilly.com/mysql2011/public/schedule/detail/17146>

Achieving PCI compliance with MySQL

<http://en.oreilly.com/mysql2011/public/schedule/detail/17141>

Diagnosing replication failures

<http://en.oreilly.com/mysql2011/public/schedule/detail/17242>

The End

- ★ Here is the URL for our slides:
 - ◆ <http://slideshare.net/MySQLGeek>
- ★ Our details again are:
 - ◆ Justin Swanhart justin.swanhart@percona.com
 - ◆ Ryan Lowe ryan.a.lowe@percona.com
 - ◆ Baron Schwartz baron@percona.com
- ★ Stop by our booth this week to say hello