



# Extracting Performance and Scalability Metrics From TCP

Baron Schwartz  
Percona Live  
May 26th, 2011



PERCONA

**Consulting  
Support  
Training  
Development**

**For MySQL**

# Diamond Sponsors



# Agenda

- Capturing TCP Traffic
- Time-Series Plotting
- Stall Detection
- Detecting Performance Problems
- Modeling Scalability
- Forecasting Performance
- Validating Input
- When Is This Useful?

# Capturing TCP Traffic

```
tcpdump -s 384 -i any -nnq -tttt  
'tcp port 3306 and (((ip[2:2] - ((ip[0]&0xf)<<2))  
- ((tcp[12]&0xf0)>>2)) != 0)' > tcp-file.txt
```

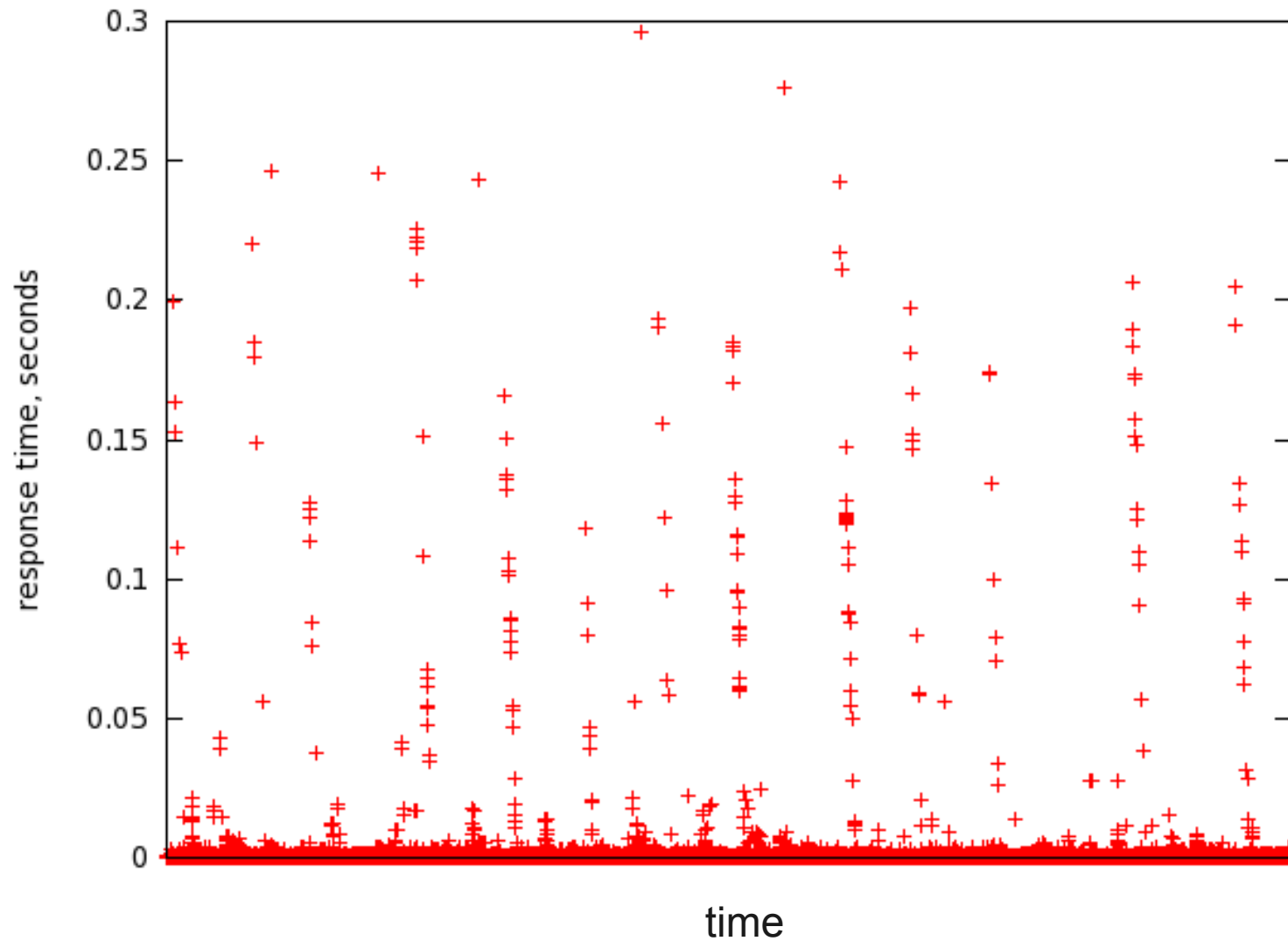
```
2011-05-05 10:47:17.810932 IP 10.220.146.79.35805 > 10.119.42.41.3306: tcp 83  
2011-05-05 10:47:17.811021 IP 10.119.42.41.3306 > 10.220.146.79.35805: tcp 64  
2011-05-05 10:47:17.811545 IP 10.250.95.31.45400 > 10.119.42.41.3306: tcp 82
```

# Process with mk-tcp-model

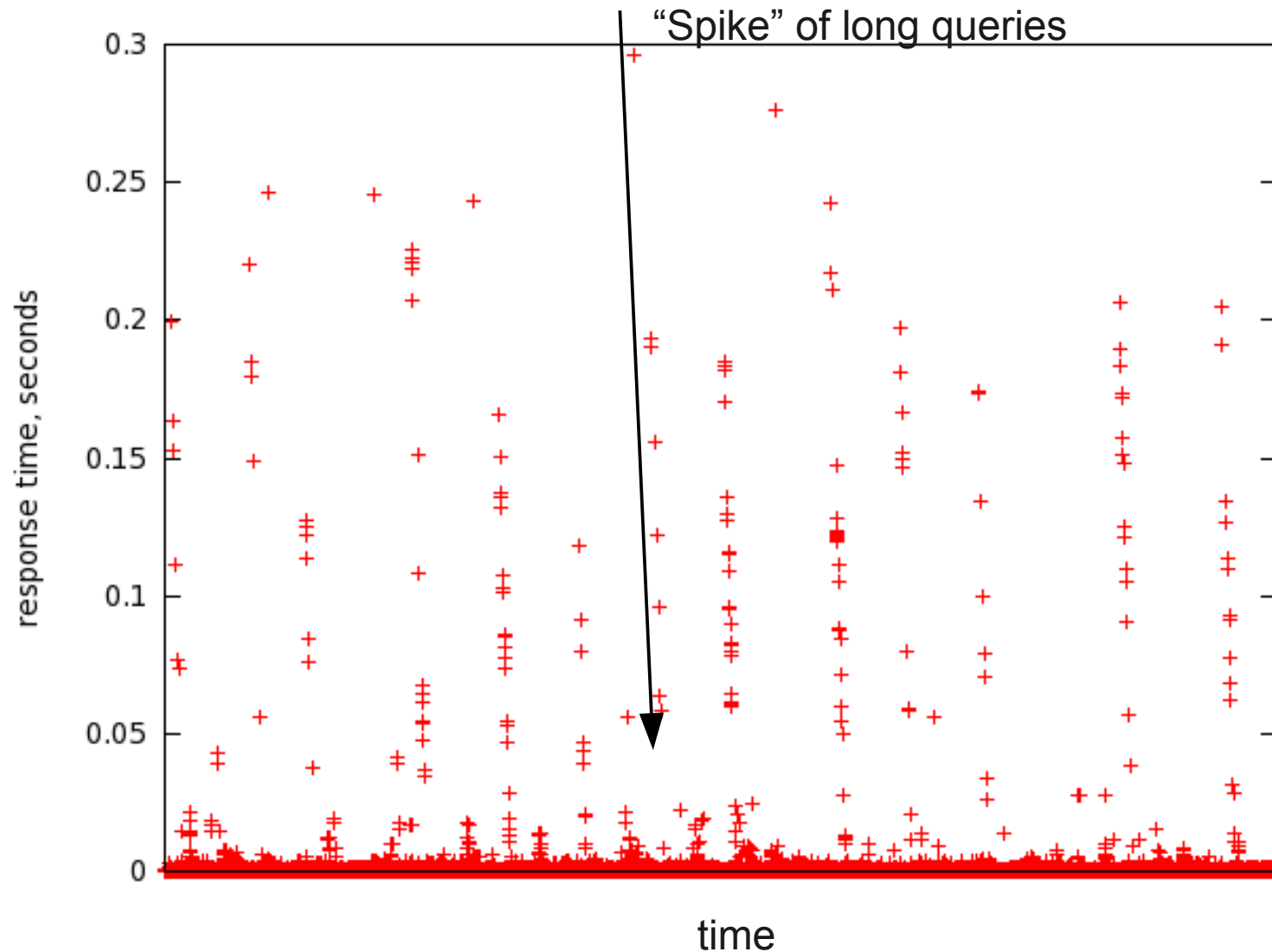
```
mk-tcp-model tcp-file.txt > requests.txt
```

#	start	end	elapsed	host:port
=	=====	=====	=====	=====
0	1304606837.810932	1304606837.811021	0.000089	10.220.146.79:35805
1	1304606837.811545	1304606837.811778	0.000233	10.250.95.31:45400
2	1304606837.811669	1304606837.811971	0.000302	10.243.78.239:45612
3	1304606837.811893	1304606837.812073	0.000180	10.222.110.47:44024
4	1304606837.813067	1304606837.813312	0.000245	10.220.146.79:35805

# Plot as Time-Series and Inspect



# Plot as Time-Series and Inspect



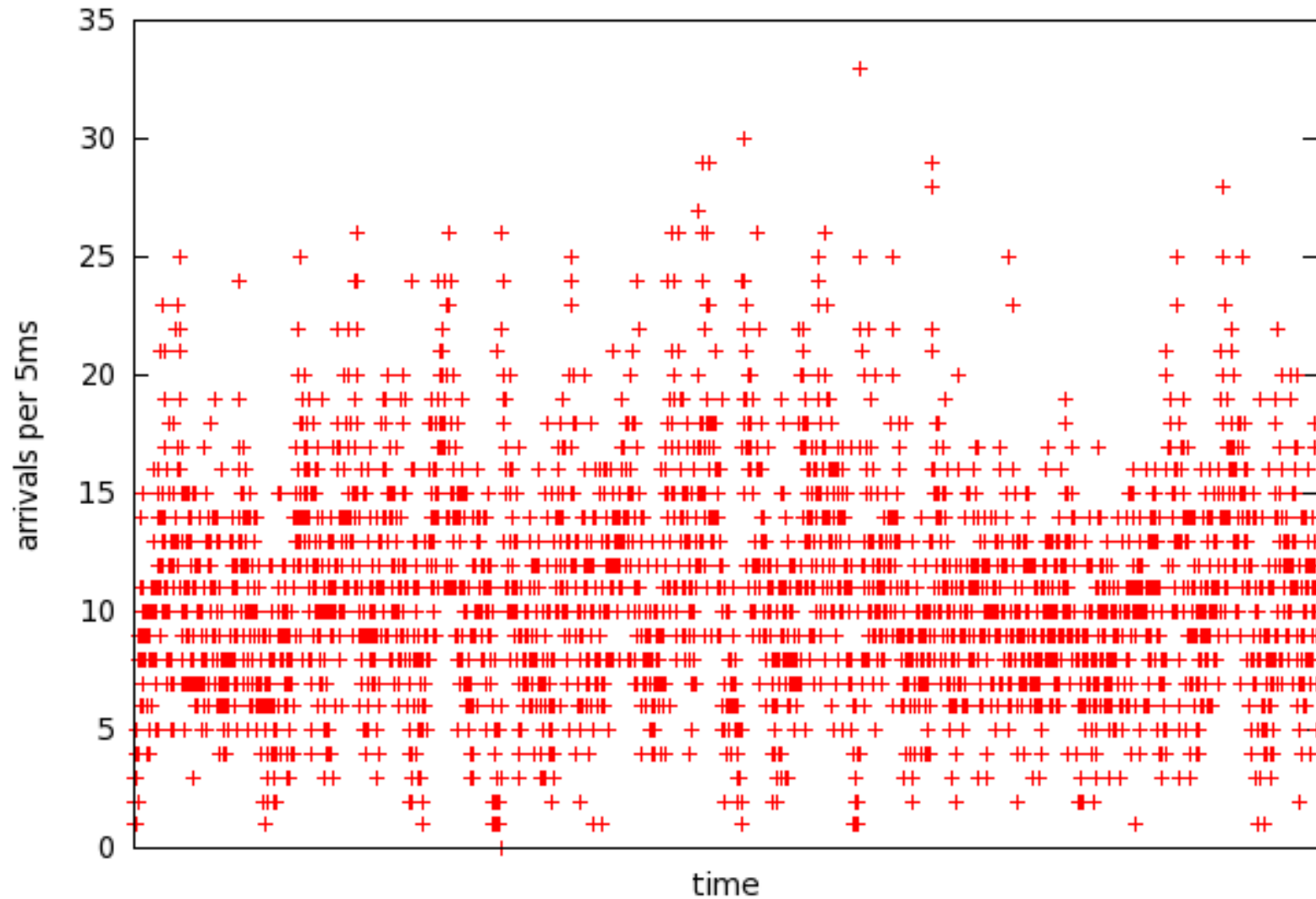
# Stall Detection

- What's happening there?
- Each query completes when the previous one releases the resource.
- A long query makes incoming queries queue.

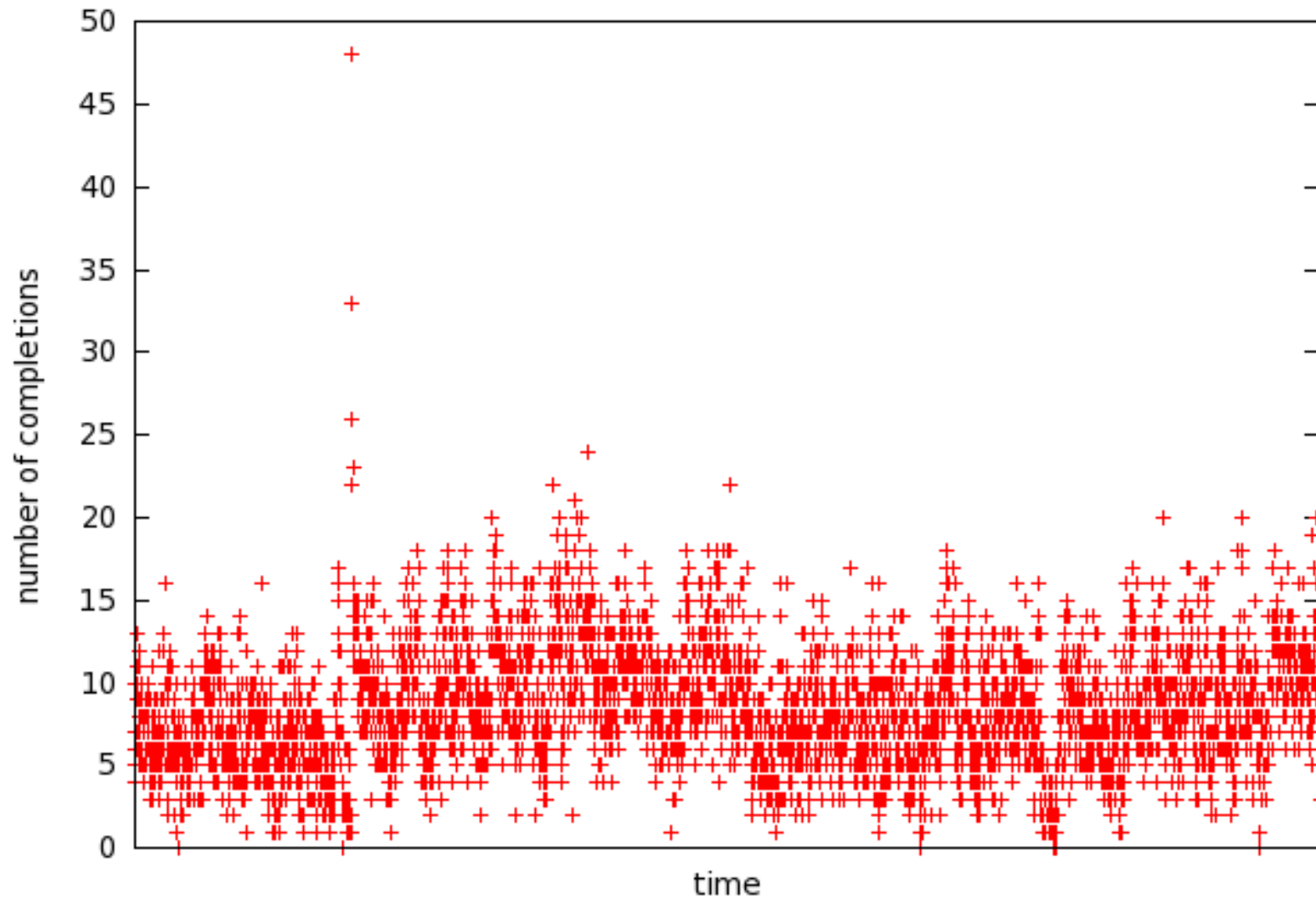
*(I happen to know it's `SELECT FOR UPDATE`).*

If completions are what's affected,  
maybe we can see the evidence.

# Arrivals per 5ms



# Completions Per 5ms



# Hmmmm

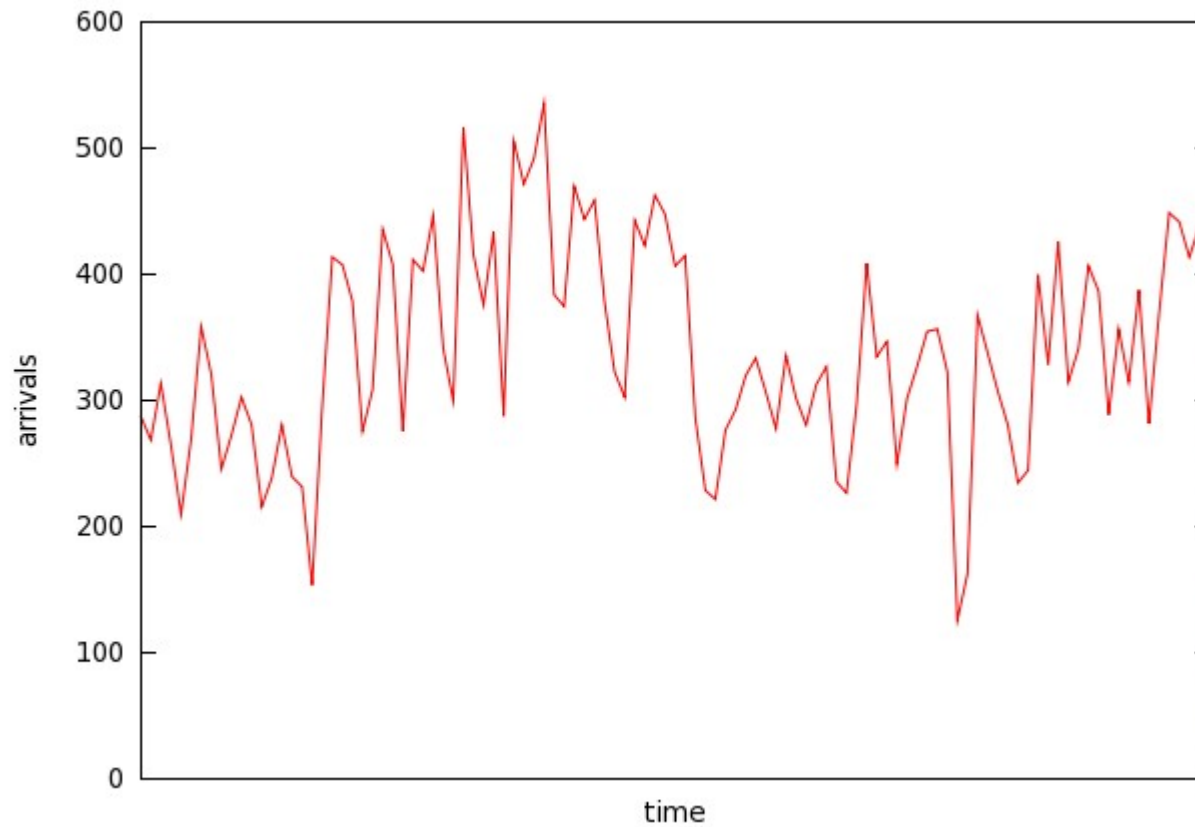
It's kind of hard to see what's going on in that graph. Is 5 ms too fine granularity?

Is there really a problem?

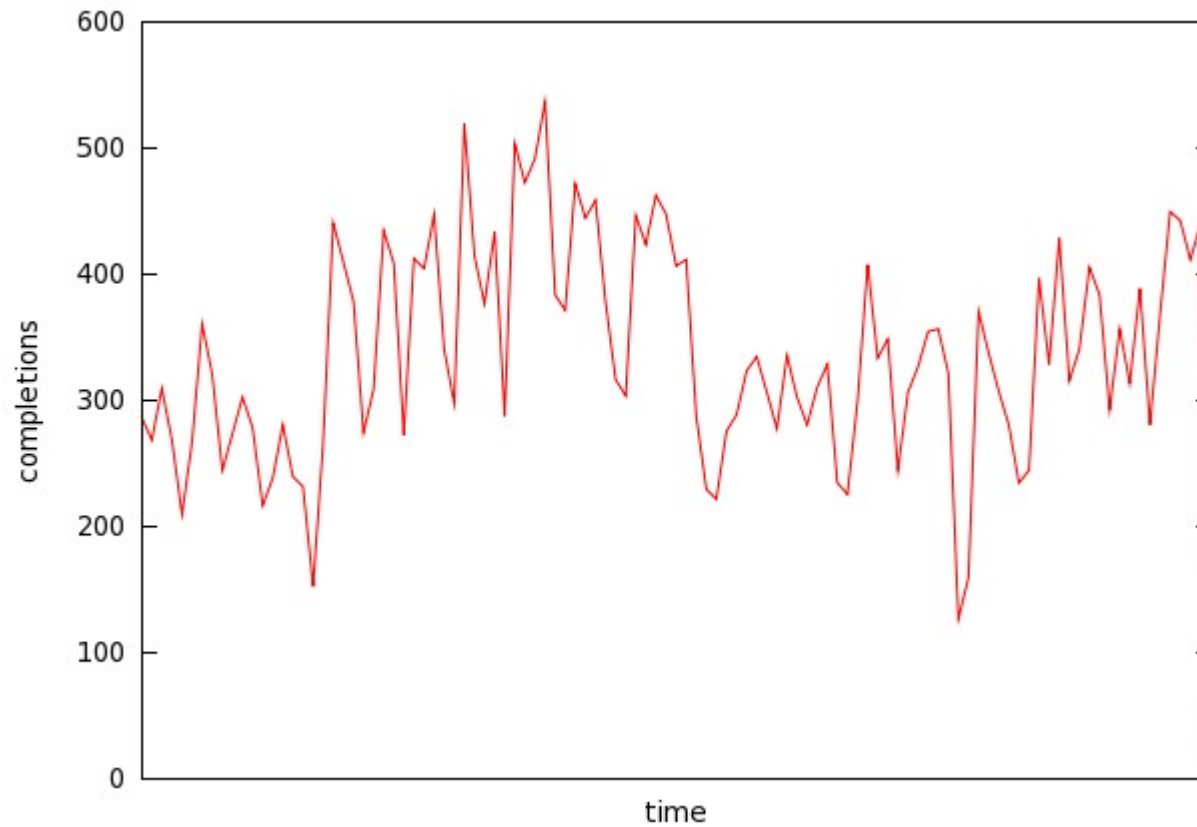
If arrivals are fairly uniform, but completions cluster together, can we see it?

*The following graphs are per 2-tenths of a sec.*

# Arrivals



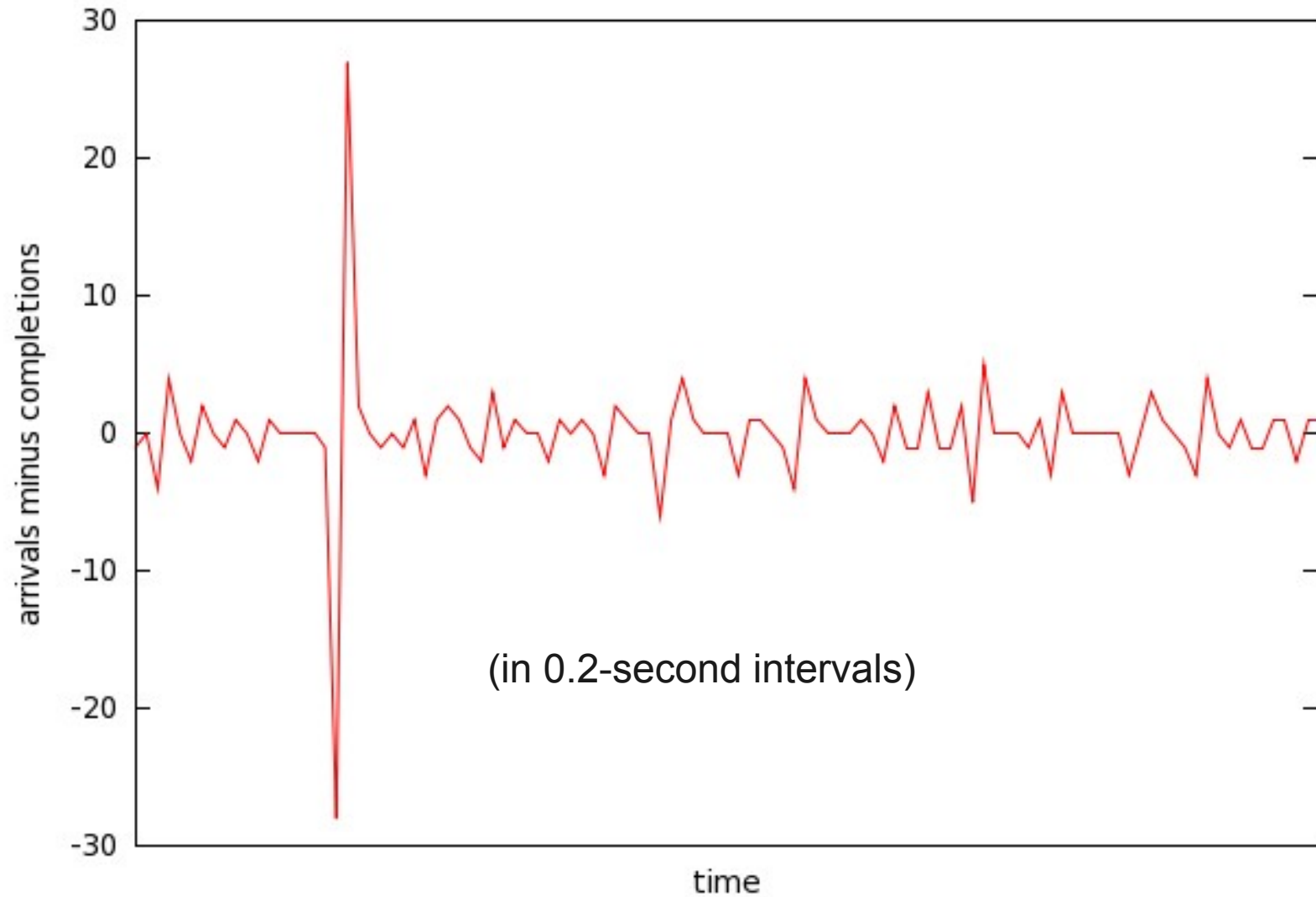
# Completions



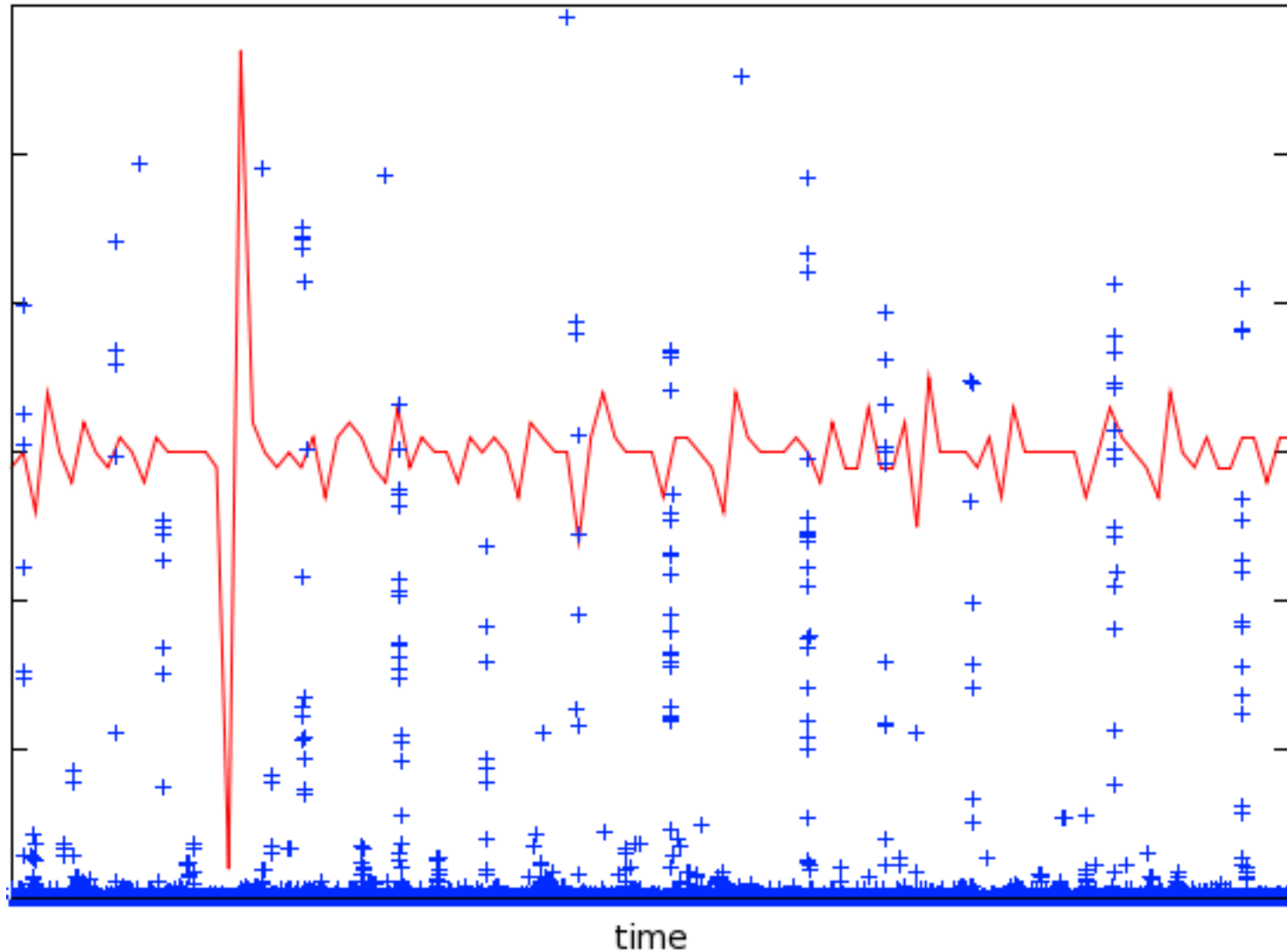
# Hmmmmmmmmmmmmmmmmmmmm

- Looks almost identical. Is my theory wrong?
- What if we plot completions minus arrivals?

# Completions Minus Arrivals



# Seeing Hidden Patterns



# Pile-Up Detection Algorithm

- Plot completions minus arrivals
- Try finer and finer granularities
- Is a 0.2-second pile-up acceptable?

# Detecting Performance Problems

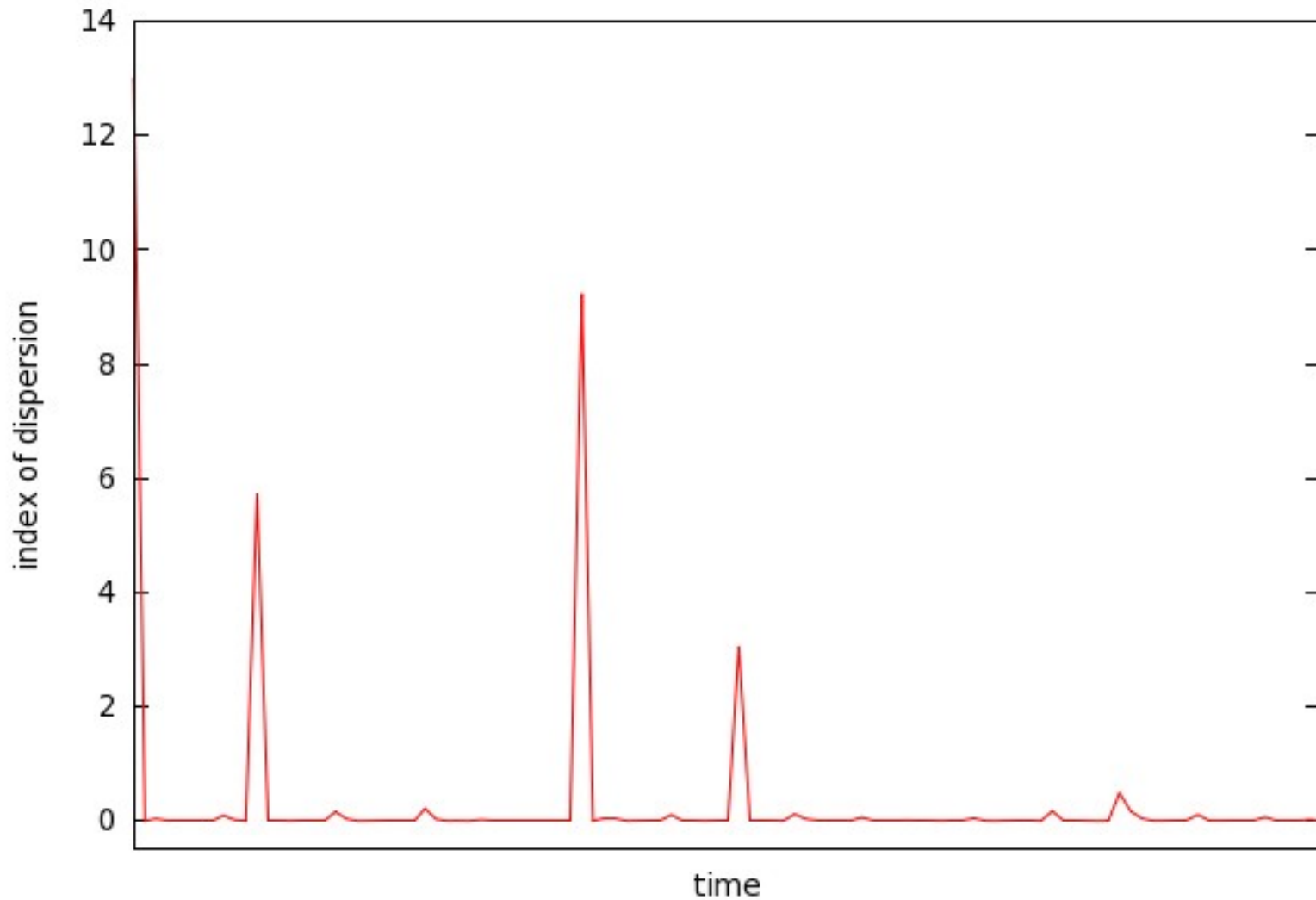
Another good statistic is the index of dispersion of response times.

$$\frac{\text{Variance}}{\text{Mean}}$$

# Index of Dispersion

- Normalized relative to average response time.
- Lets you compare different workloads.

# Index of Dispersion

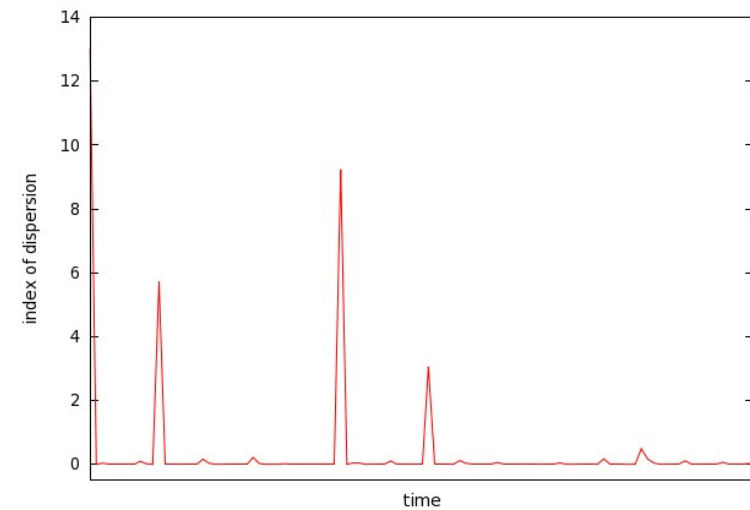
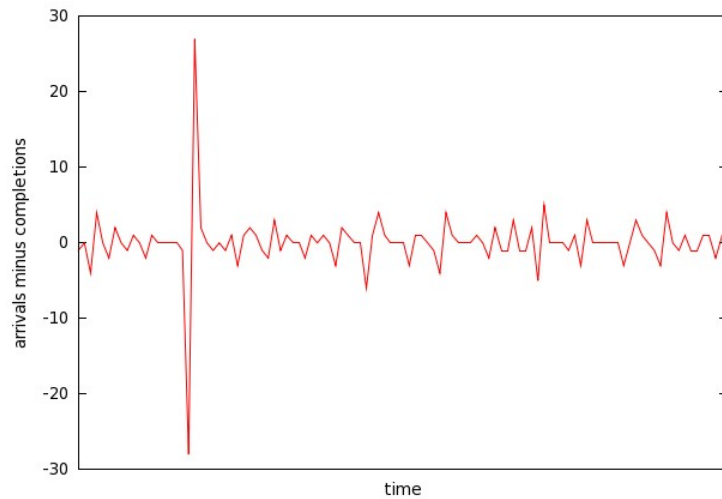
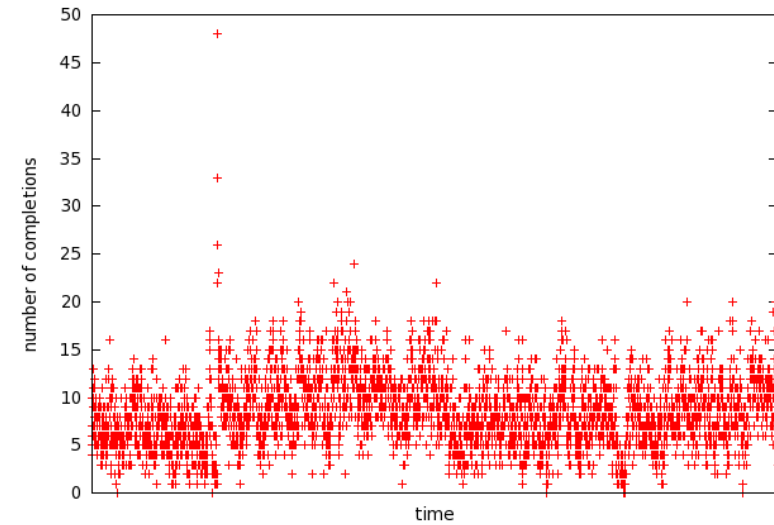
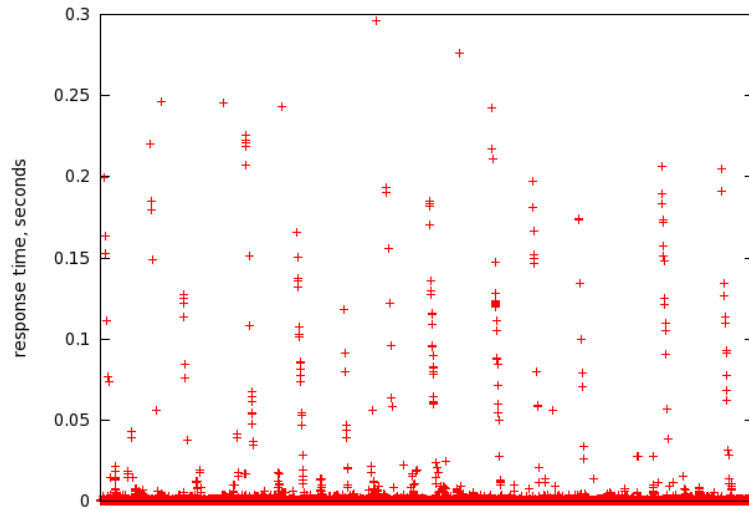


# Interpreting Index of Dispersion

- What does it MEAN when there's a spike?
- “This time period is highly variable.”
- “Highly variable performance is bad.”

Highly variable == highly optimizable.

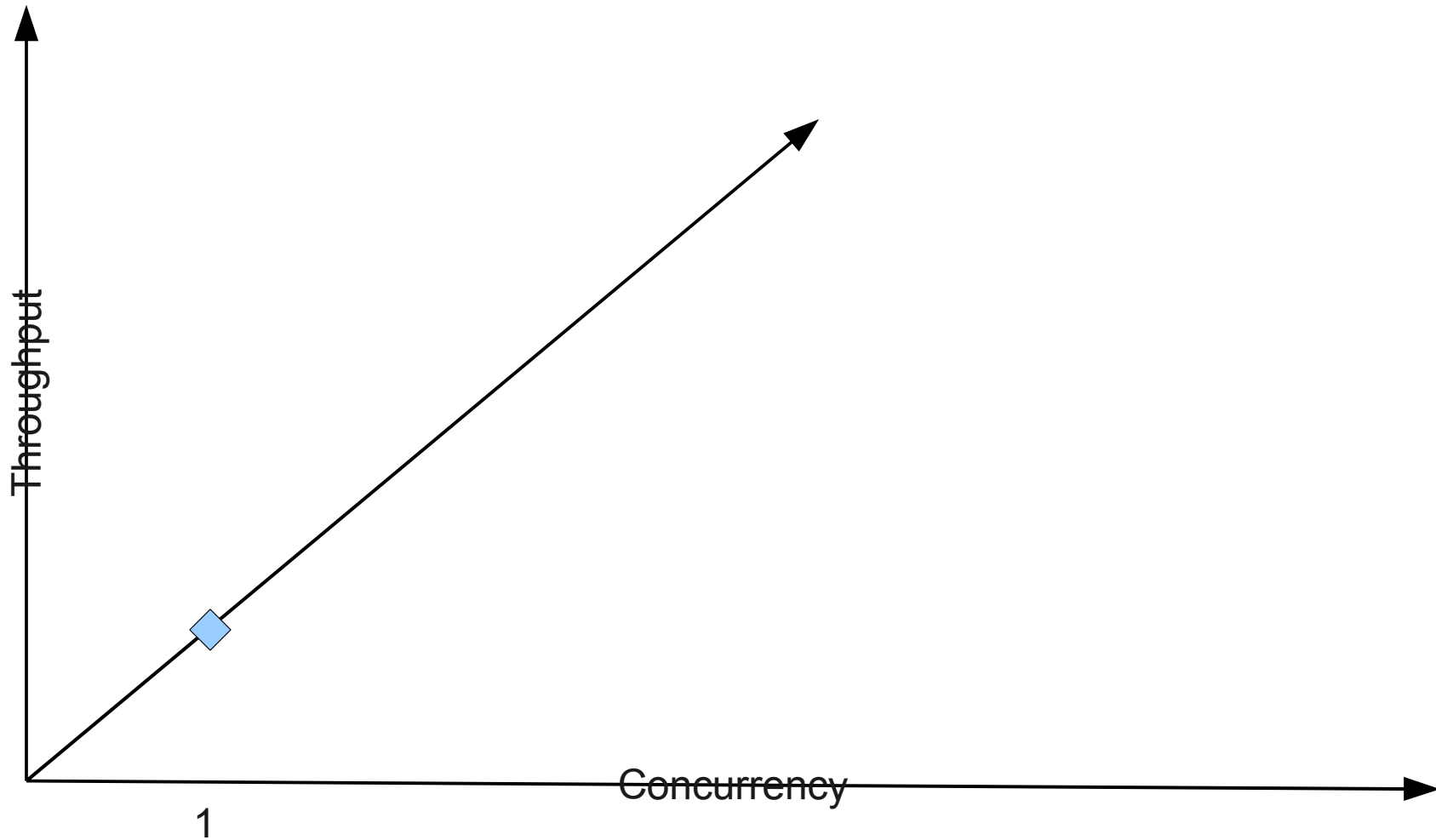
# Time-Series, All Together Now



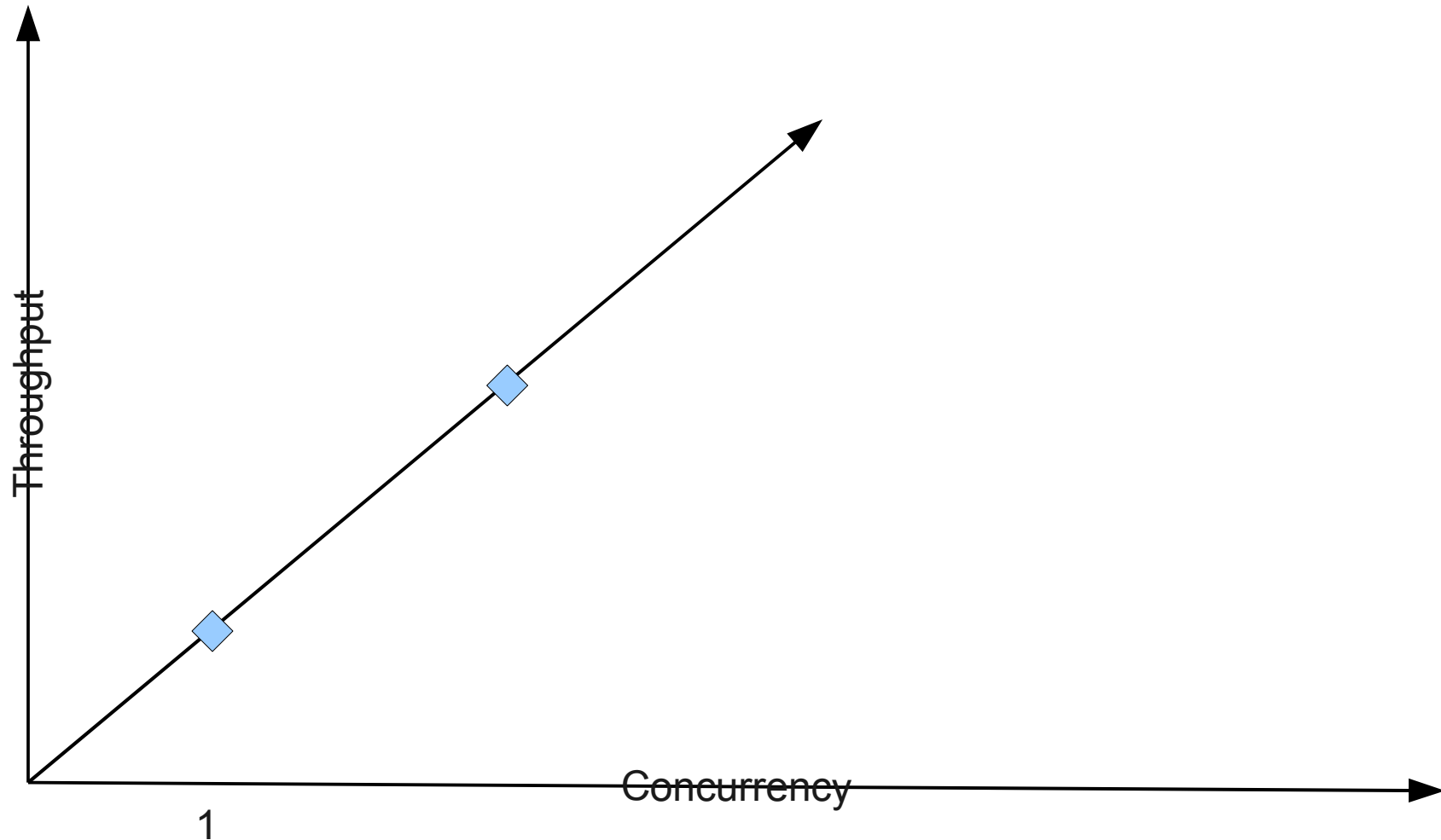
# Modeling Scalability

- Scalability is a function (equation)
- The X-axis is Concurrency
- The Y-axis is Throughput

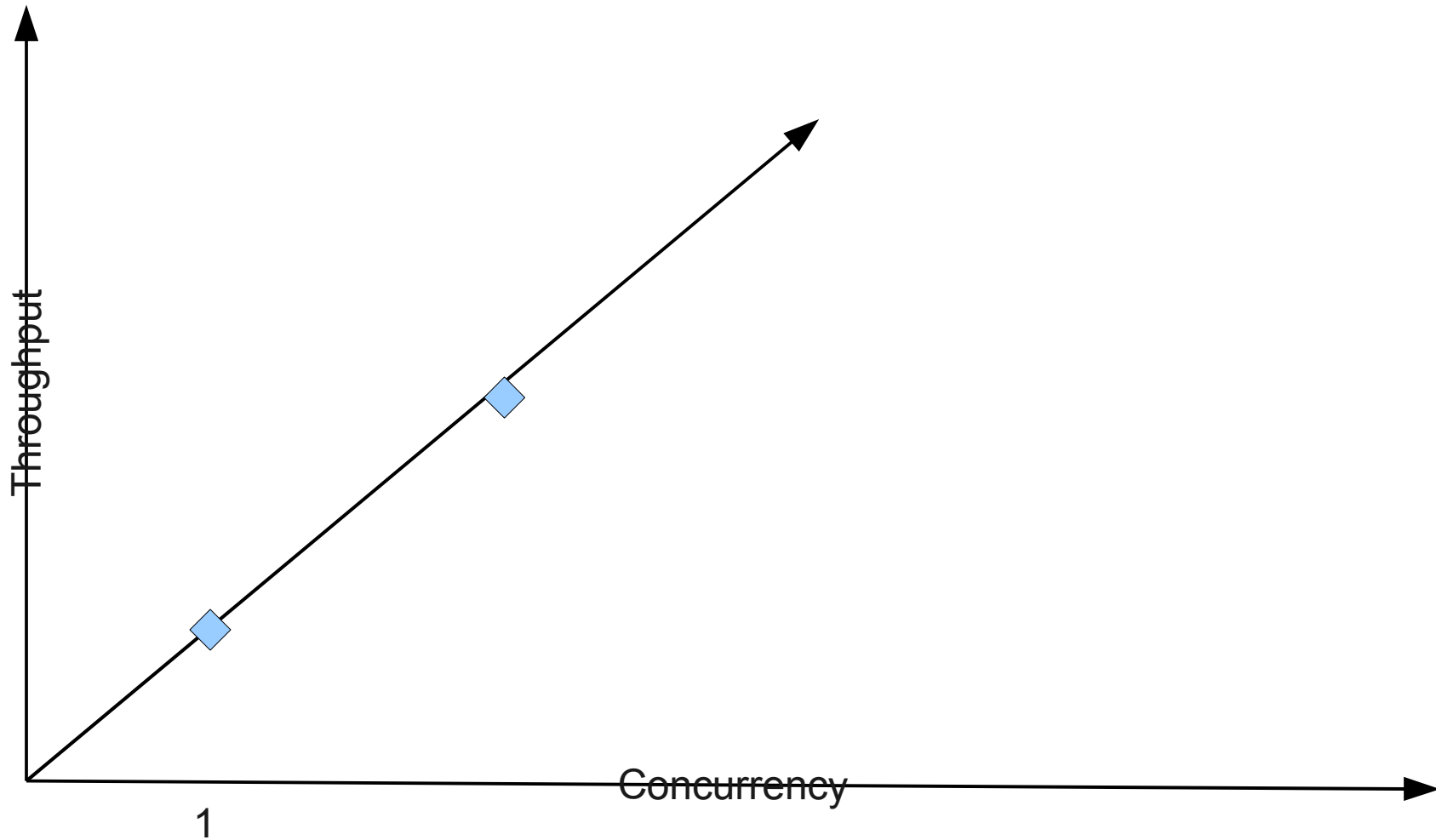
# The Scalability Function



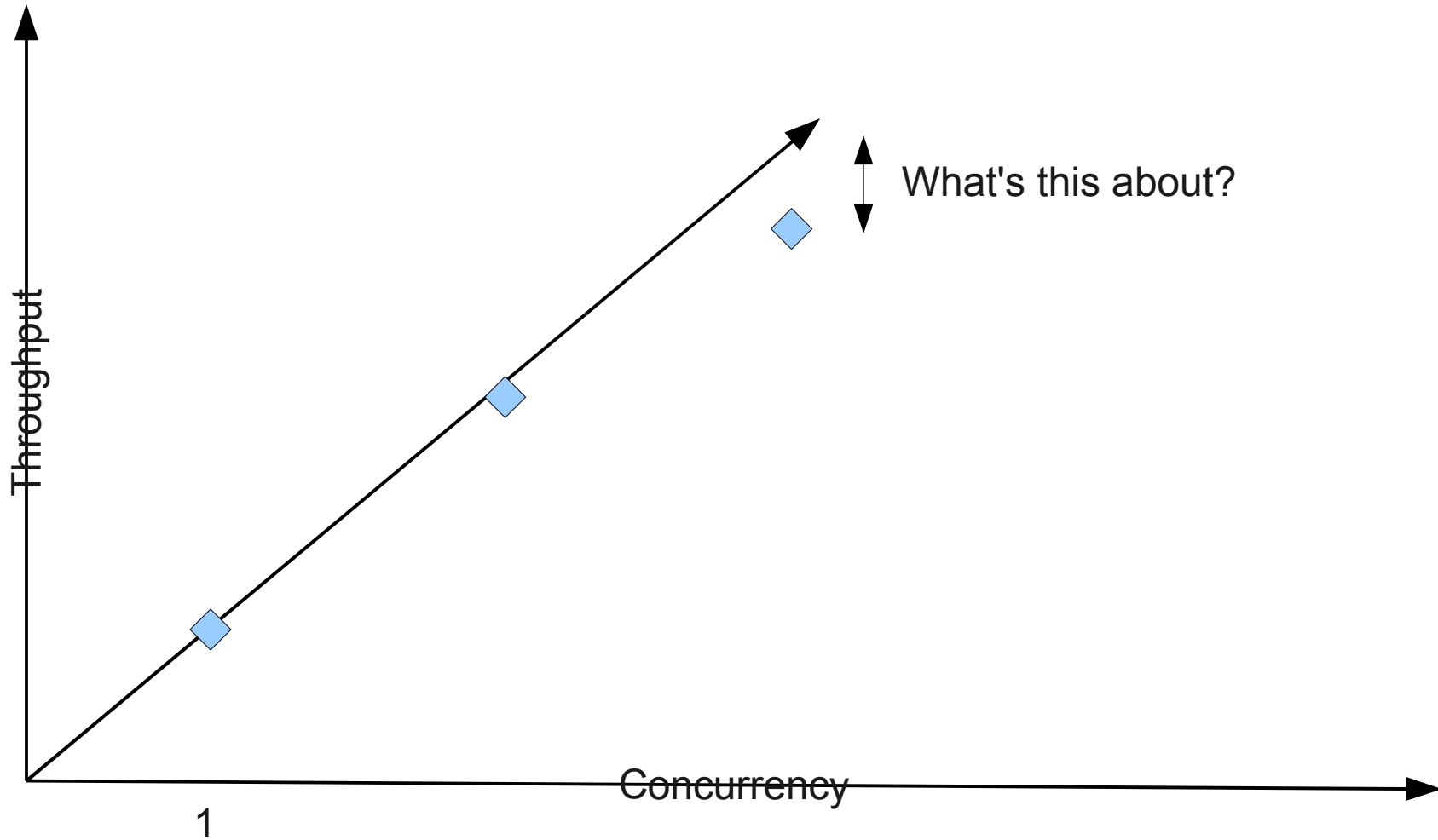
# This is Linear Scalability



# This is Not Linear Scalability



# What Causes Non-Linearity?



# Factor #1: Serialization

- Some portion of the work cannot be done in parallel
- This is Amdahl's Law

$$C(N) = \frac{N}{1 + \sigma(N - 1)}$$

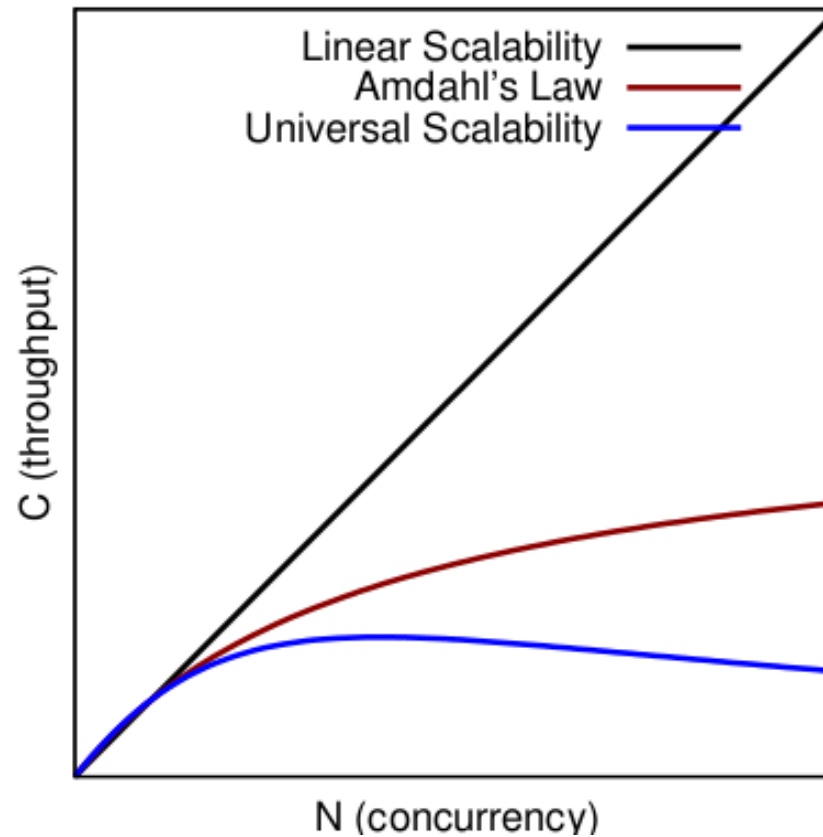
# Factor #2: Coherency

- Some portion of the work relies on IPC, cross-node communication, etc
- Dr. Neil Gunther's University Scalability Law:

$$C(N) = \frac{N}{1 + \sigma(N - 1) + \kappa N(N - 1)}$$

# Loss of Scalability

- Most systems have serialization & coherency.



# Scalability Modeling Method

- Measure Throughput and Concurrency.
- Perform a regression against the USL.  
(Determine sigma and kappa coefficients)
- ?????
- Profit!

$$C(N) = \frac{N}{1 + \sigma(N - 1) + \kappa N(N - 1)}$$

# Concurrency from TCP

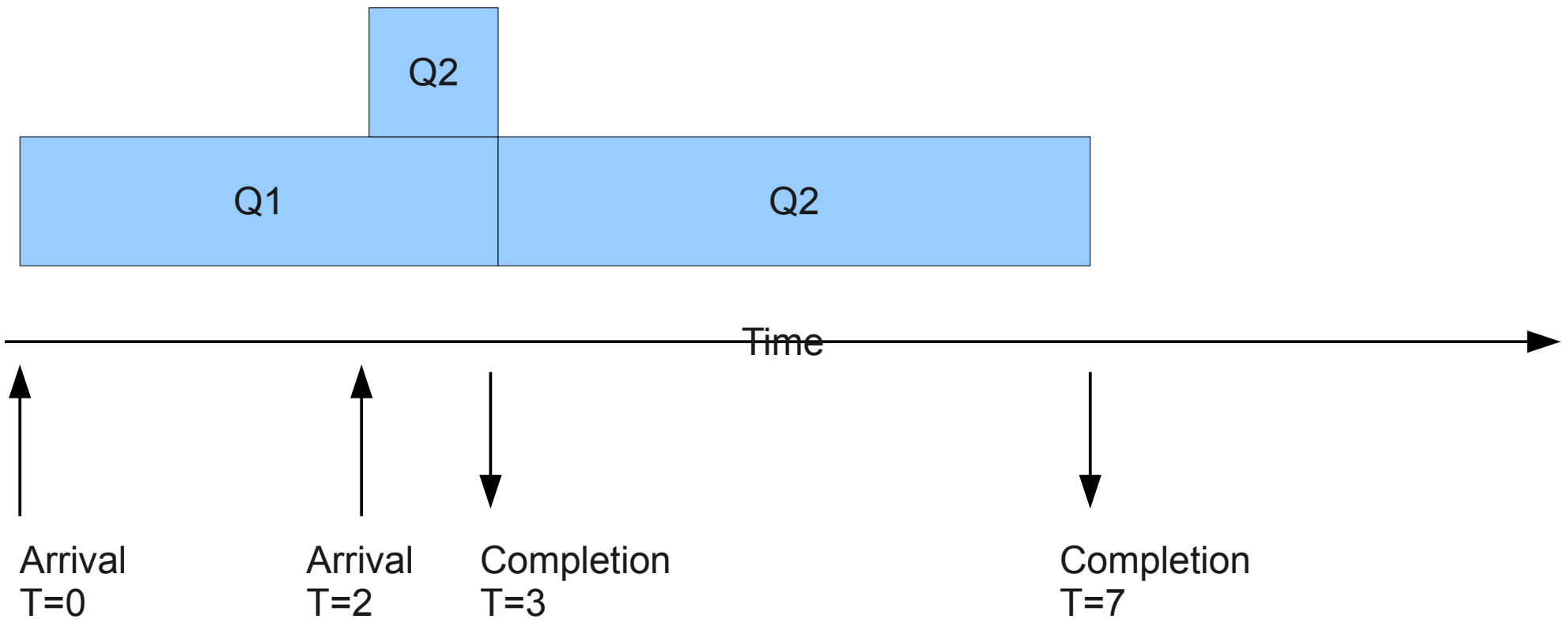
- Throughput is easy (queries per second)
- Concurrency is harder.

# Concurrency from TCP

- Sort the arrivals and departures in order.
  - Each arrival increments concurrency.
  - Each departure decrements it.
- Calculate a moving weighted average.

# Visually...

Observation Time: 7  
Total Query Time: 8  
Average Concurrency: 8/7



# Using mk-tcp-model

- mk-tcp-model knows how to compute this.

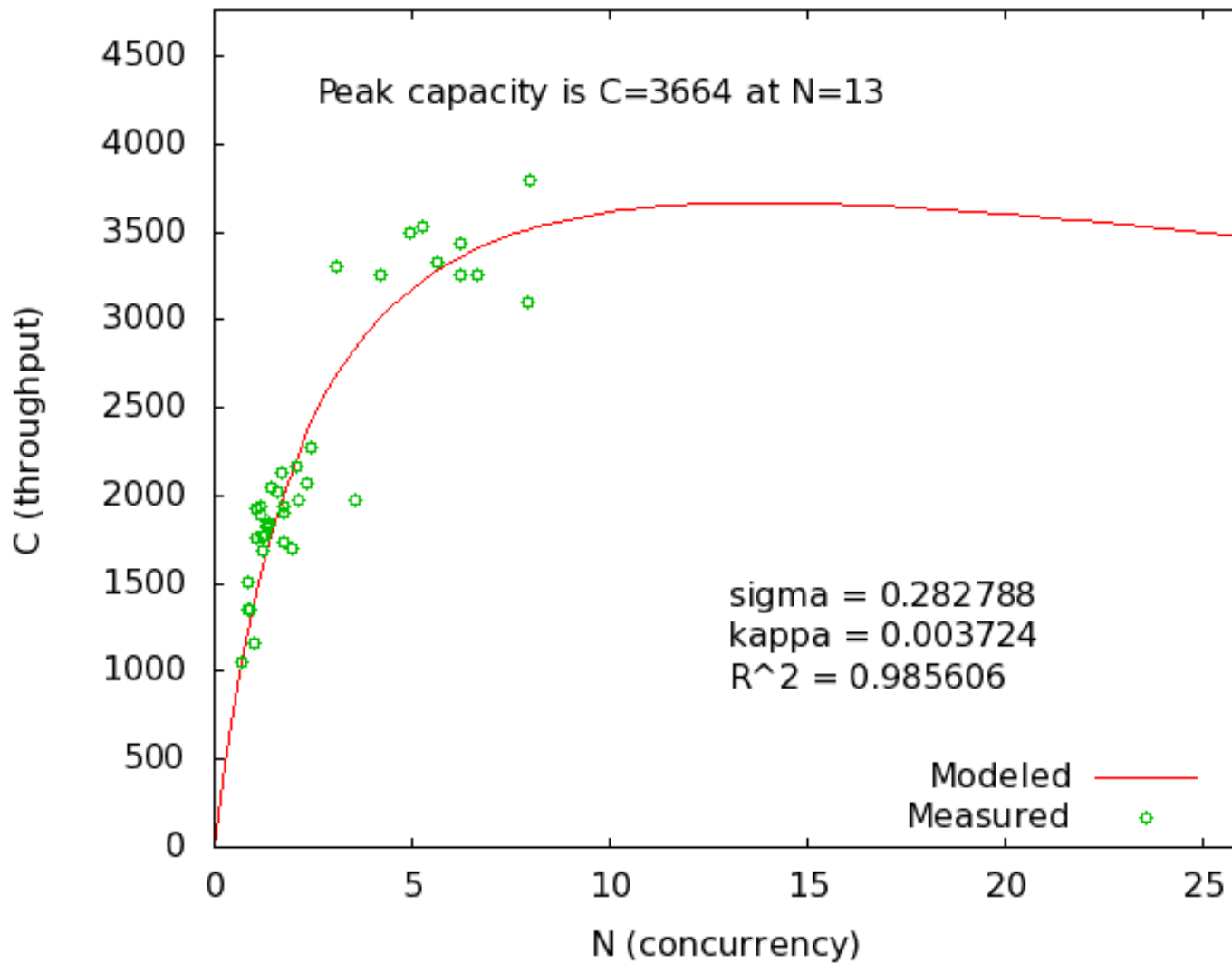
```
sort -n -k1,1 requests.txt > sorted.txt
```

```
mk-tcp-model --type=requests sorted.txt > sliced.txt
```

# Using Aspersa's USL Tool

- Aspersa has a “usl” tool that does the regression for you.

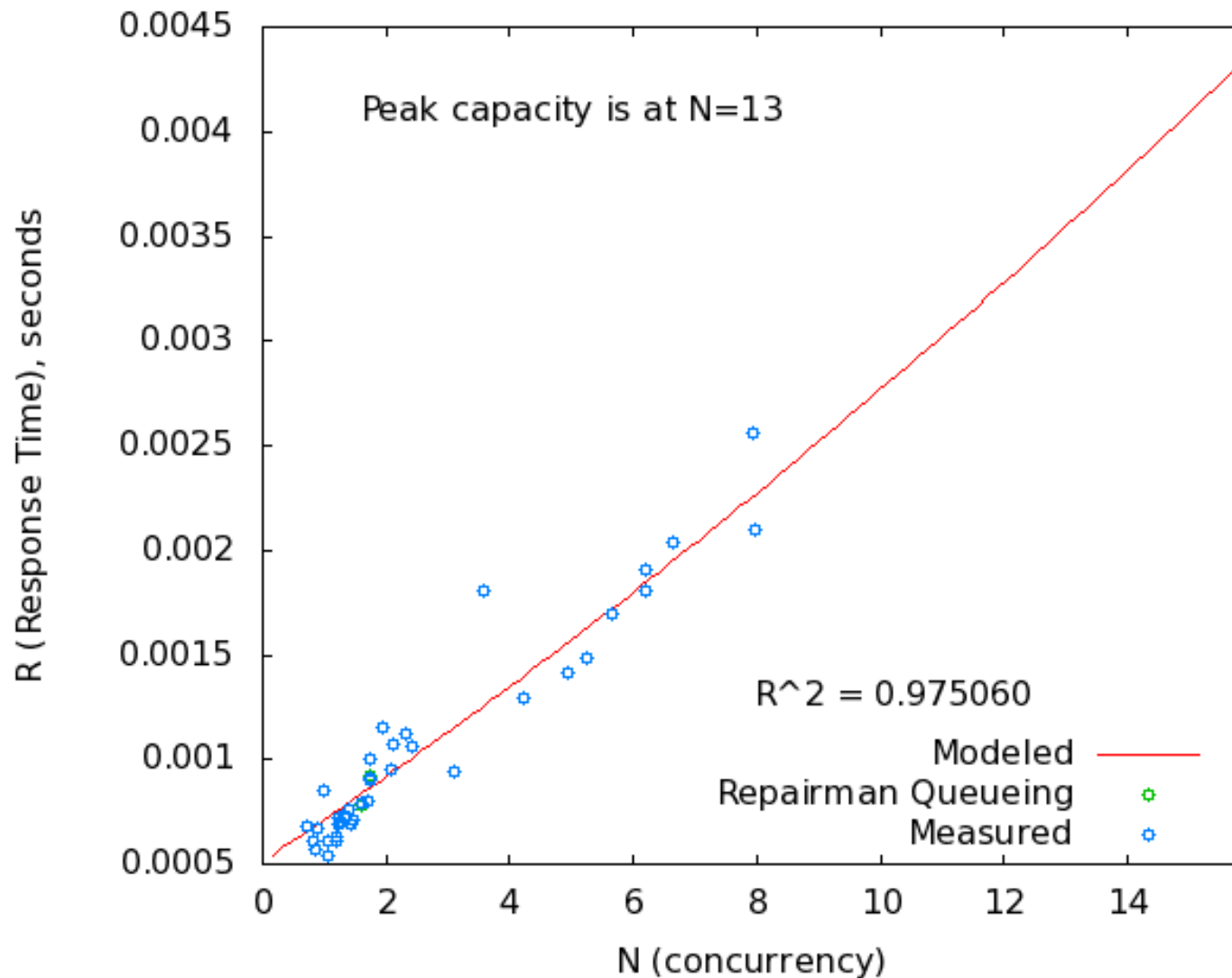
# Results



# Forecasting Performance

- Performance = Response Time
- Little's Law:  $N = XR$   
(Concurrency = Throughput \* Response Time)

# Forecasting Performance



# Validating Input

- Do not just throw data at the model.
- Beware of dropped packets in tcpdump.
- You may need to remove outliers.
- Beware of highly mixed / changing workloads.

# When Is This Useful?

- Worst-case bound
- Best-case bound

# Be Vewwy Vewwy Quiet

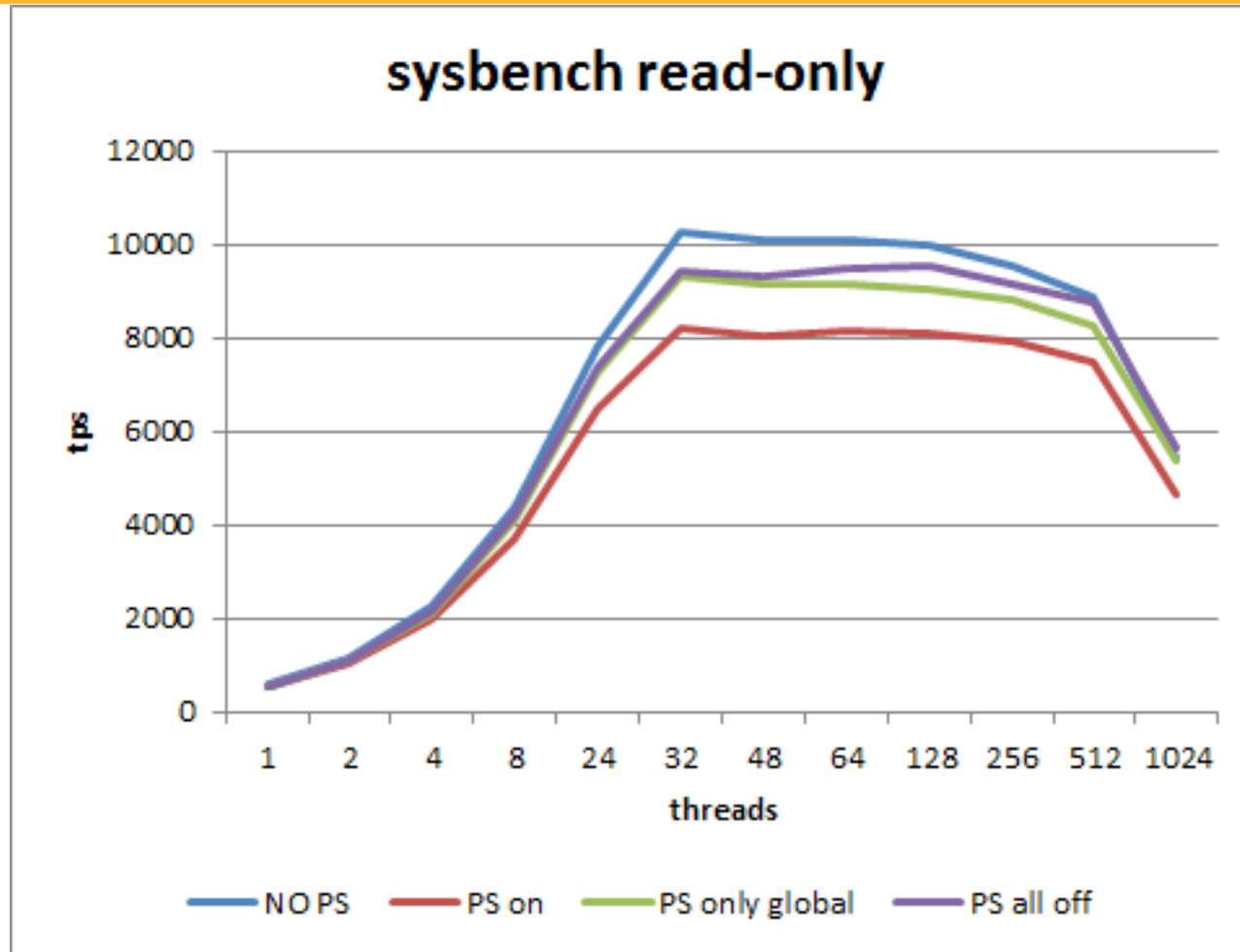


I'm wooking for bottwenecks!

# Worst-Case Scaling

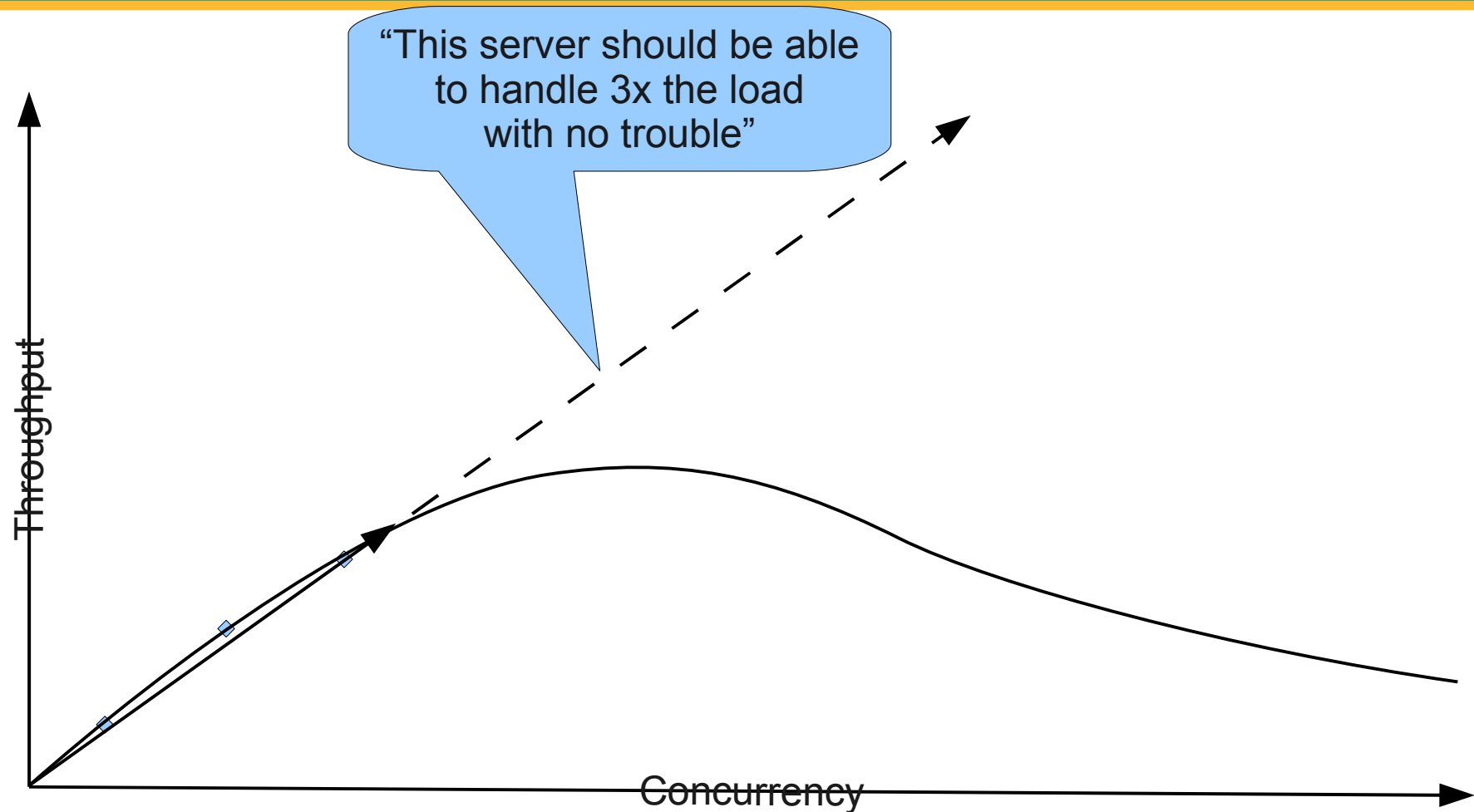
- The USL models synchronous repairman queuing behavior.
- This is worst-case, and your system should scale *better* than this.
- If not, you have a bottleneck. Go hunting.

# Double Rainbow!



<http://www.mysqlperformanceblog.com/2011/04/25/performance-schema-overhead/>

# “Assuming Linear Scalability...”



# Best-Case Bounds

- We know most systems don't scale as well as they're supposed to.
- Therefore, the USL can be used for capacity planning purposes as a *best-case* bound.
- “I expect this system not to scale as well as predicted, so I'd better not count on getting any more performance than the model indicates.”

# Summary

- We can get arrivals & completions from TCP
- And measure query performance easily
- And find problems the eye can't see
- And forecast beyond what we can observe
- For nearly any system
- Including ones that have no instrumentation!
- Forecasting/modeling requires experience and judgment, and does not give exact answers.

# Further Research

[percona.com/about-us/mysql-white-papers](http://percona.com/about-us/mysql-white-papers)

Contact information: [baron@percona.com](mailto:baron@percona.com)

Thanks!



[www.percona.com/live](http://www.percona.com/live)