
PSYCHIC
Linux Systems Capacity Planning
Rodrigo Campos
camposr@gmail.com - @xinu
USENIX LISA '11 - Boston, MA

Agenda

- Where, what, why?
- Performance monitoring
- Capacity Planning
- Putting it all together

Where, what, why ?

- 75 million internet users
- 1,419.6% growth (2000-2011)
- 29% increase in unique IPv4 addresses (2010-2011)
- 37% population penetration

Sources:
Internet World Stats - http://www.internetworldstats.com/stats15.htm
Akamai's State of the Internet 2nd Quarter 2011 report - http://www.akamai.com/stateoftheinternet/

Where, what, why ?

- High taxes
- Shrinking budgets
- High Infrastructure costs
- Complicated (immature?) procurement processes
- Lack of economically feasible hardware options
- **Lack of technically qualified professionals**

Where, what, why ?

- **Do more with the same infrastructure**
- Move away from tactical fire fighting
- While at it, handle:
  - Unpredicted traffic spikes
  - High demand events
  - Organic growth

Performance Monitoring

- Typical system performance metrics
  - CPU usage
  - IO rates
  - Memory usage
  - Network traffic

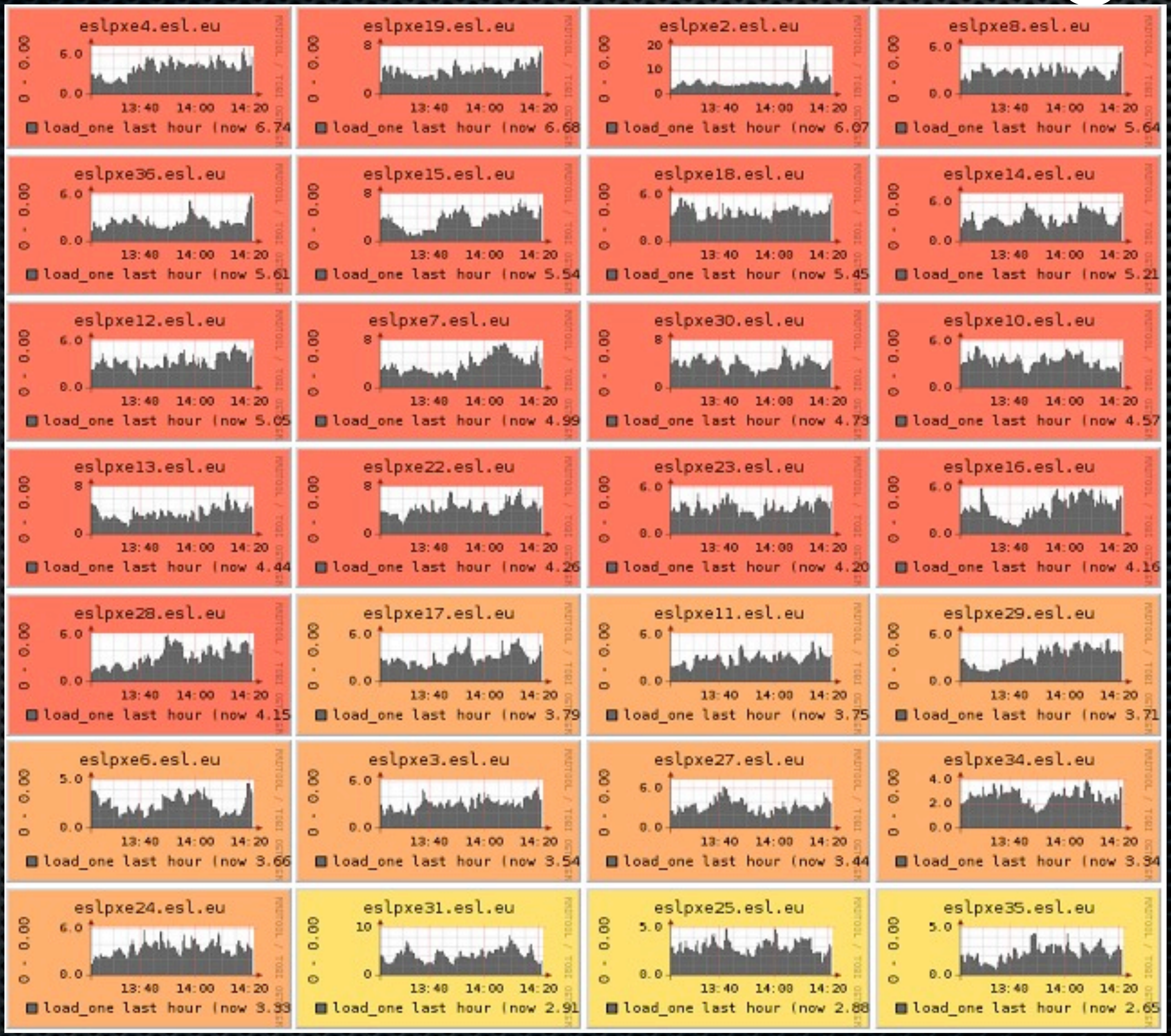
Performance Monitoring

- Commonly used tools:
  - Sysstat package - iostat, mpstat *et al*
  - Bundled command line utilities - ps, top, uptime
  - **Time series charts (orcallator's offspring)**
    - Many are based on RRD (cacti, torrus, ganglia, collectd)

Performance Monitoring

- Time series performance data is useful for:
  - Troubleshooting
  - Simplistic forecasting
  - Find trends and seasonal behavior

Performance Monitoring



Performance Monitoring

- **"Correlation does not imply causation"**
- Time series methods won’t help you much for:
  - Create what-if scenarios
  - Fully understand application behavior
  - Identify non obvious bottlenecks

Monitoring vs. Modeling

“The difference between performance modeling and performance monitoring is like the difference between weather prediction and simply watching a weather-vane twist in the wind”

Source: http://www,perfdynamics,com/Manifesto/gcaprules,html

Capacity Planning

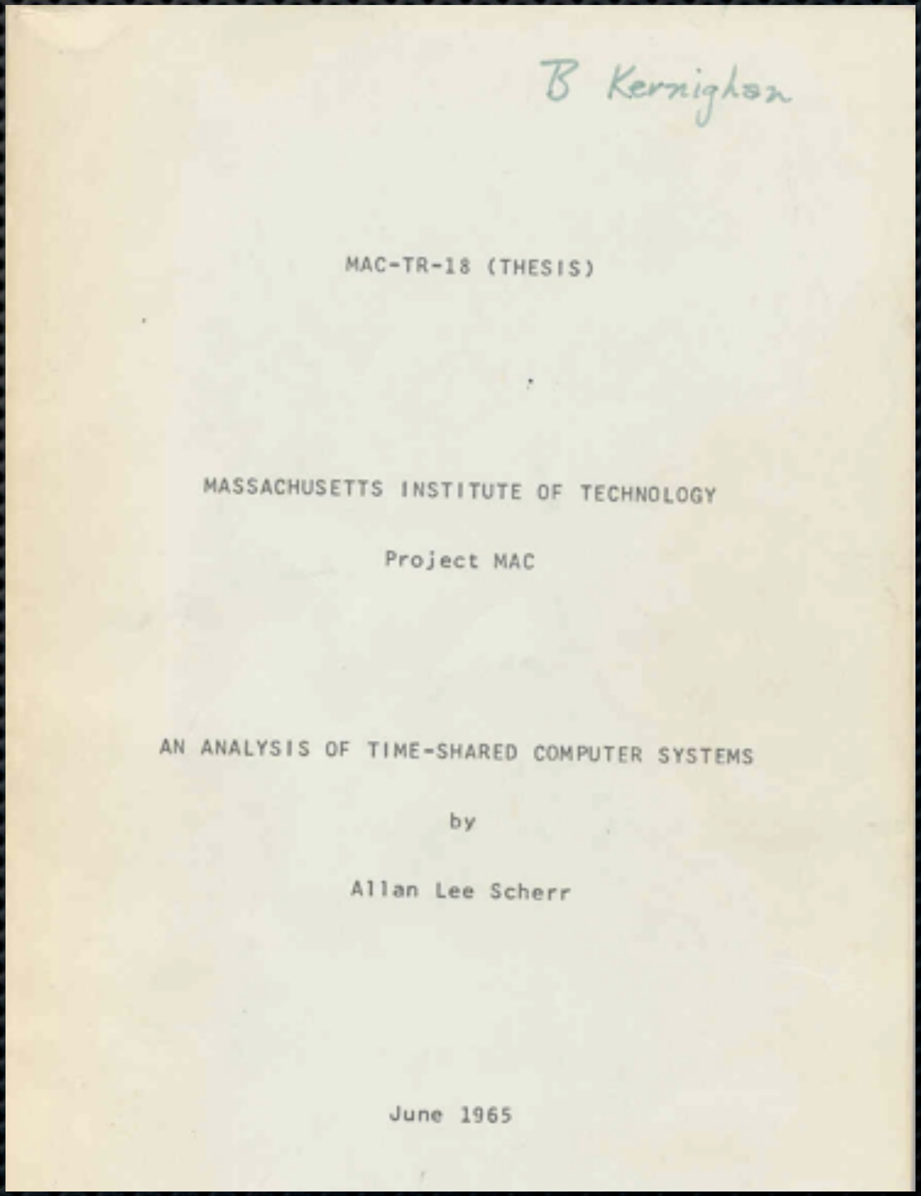
- Not exactly something new...
- Can we apply the very same techniques to modern, distributed systems ?
- Should we ?

What’s in a queue ?

- Agner Krarup Erlang
- Invented the fields of traffic engineering and queuing theory
- 1909 - Published “The theory of Probabilities and Telephone Conversations”

What's in a queue ?

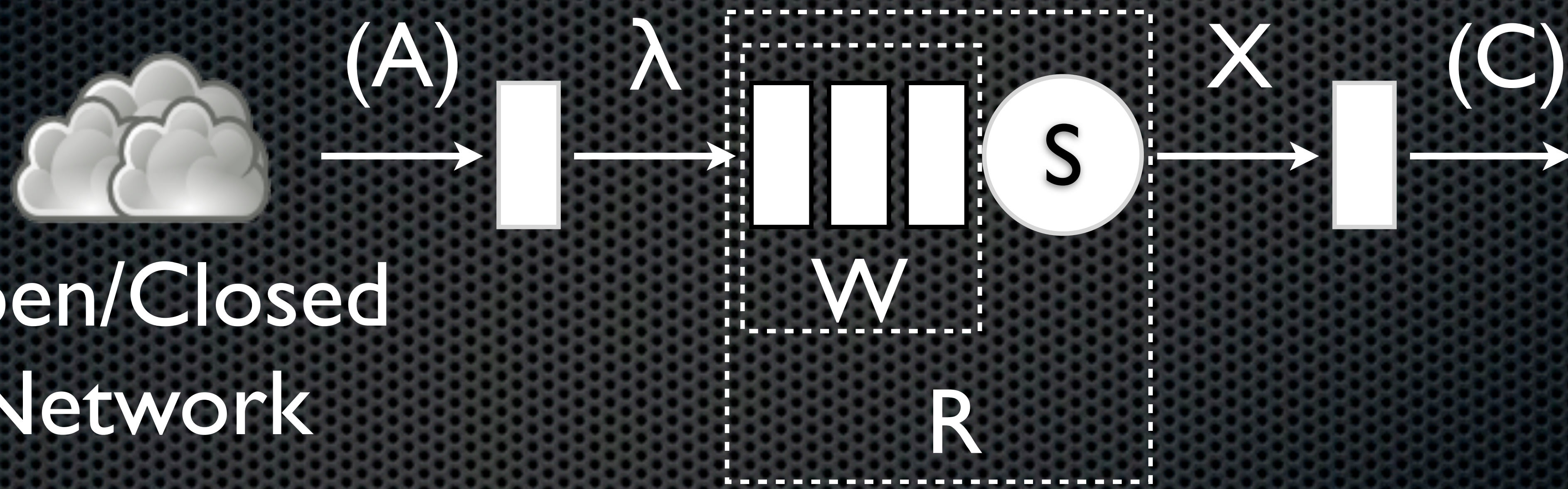
- Allan Scherr (1967) used the machine repairman problem to represent a timesharing system with *n* terminals



What’s in a queue ?

- Dr. Leonard Kleinrock
- “Queueing Systems” (1975) - ISBN 0471491101
- Created the basic principles of packet switching while at MIT

What's in a queue ?



A	Arrival Count
λ	Arrival Rate (A/T)
W	Time spent in Queue
R	Residence Time (W+S)
S	Service Time
X	System Throughput (C/T)
C	Completed tasks count

Service Time

- Time spent in processing (S)
  - Web server response time
  - Total Query time
  - Time spent in IO operation

System Throughput

- Arrival rate (λ) and system throughput (X) are the same in a steady queue system (i.e. stable queue size)
  - Hits per second
  - Queries per second
  - IOPS

Utilization

- Utilization (ρ) is the amount of time that a queuing node (e.g. a server) is busy (B) during the measurement period (T)
- Pretty simple, but helps us to get processor share of an application using `getrusage()` output
- Important when you have multicore systems

$$\rho = B/T$$

Utilization

- CPU bound HPC application running in a two core virtualized system
- Every 10 seconds it prints resource utilization data to a log file

Utilization

```
(void)getrusage(RUSAGE_SELF, &ru);
(void)printRusage(&ru);
...
static void printRusage(struct rusage *ru)
{
   fprintf(stderr, "user time = %lf\n",
      (double)ru->ru_utime.tv_sec + (double)ru->ru_utime.tv_usec / 1000000);
   fprintf(stderr, "system time = %lf\n",
      (double)ru->ru_stime.tv_sec + (double)ru->ru_stime.tv_usec / 1000000);
} // end of printRusage
```

```
10 seconds wallclock time
377,632 jobs done
user time = 7.028439
system time = 0.008000
```

Utilization

We have 2 cores so we can run 3 application instances in each server (200/70.36) = 2.84

$$\rho = B/T$$

$$\rho = (7.028+0.008) / 10$$

$$\rho = 70.36\%$$

Little's Law

- Named after MIT professor John Dutton Conant Little
- The long-term average number of customers in a stable system L is equal to the long-term average effective arrival rate, λ, multiplied by the average time a customer spends in the system, W; or expressed algebraically: $L = \lambda W$
- **You can use this to calculate the minimum amount of spare workers in any application**

Little’s Law

- L = λW
- λ = 120 hits/s
- W = Round-trip delay + service time
- W = 0.01594 + 0.07834 = 0.09428
- L = 120 * 0.09428 = 11,31

`tcpdump -vttttt`

Utilization and Little's Law

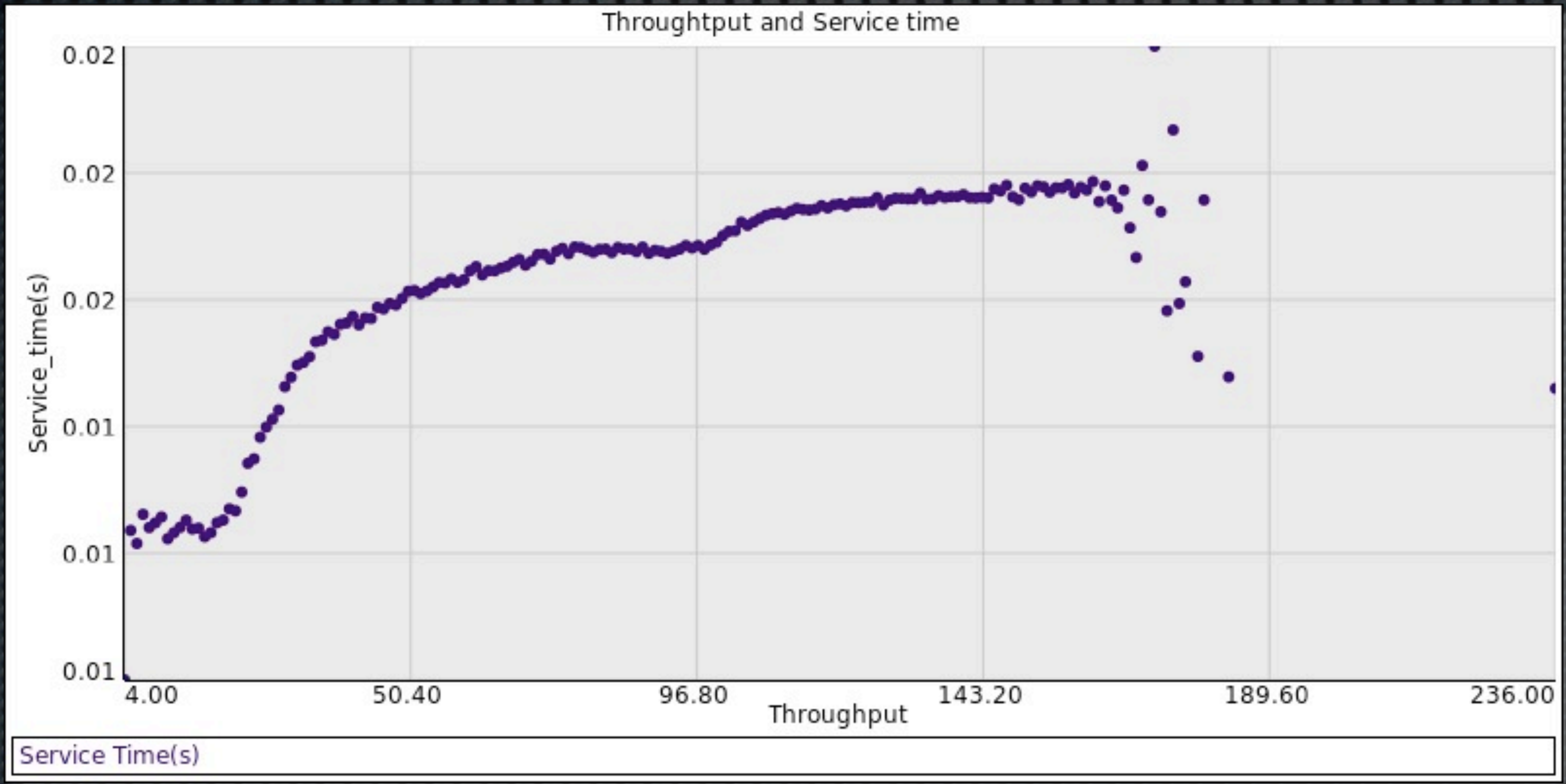
- By substitution, we can get the utilization by multiplying the arrival rate and the mean service time

$$\rho = \lambda S$$

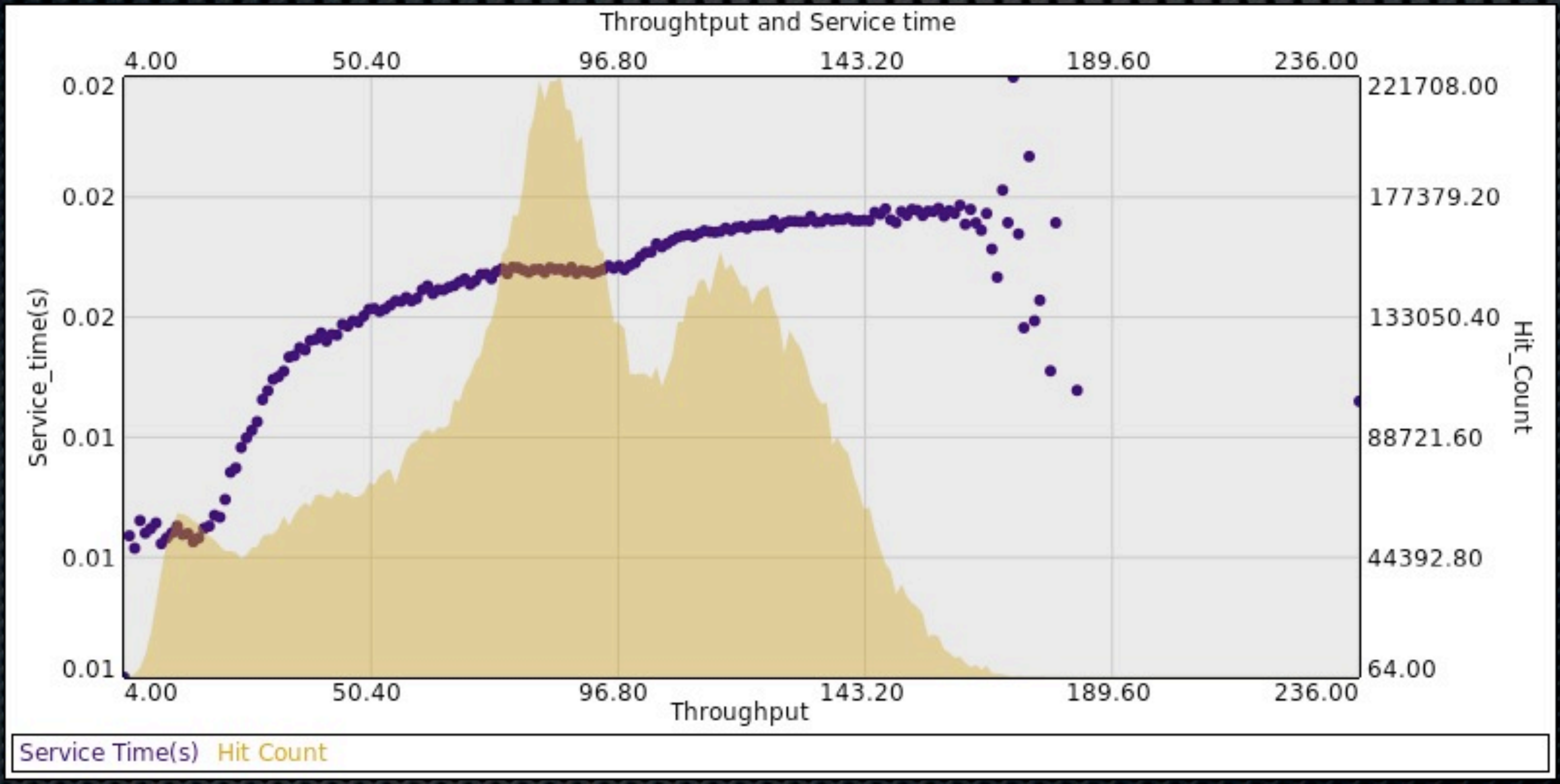
Putting it all together

- Applications write in a log file the service time and throughput for most operations
- For Apache:
  - %D in mod_log_config (microseconds)
  - "ExtendedStatus On" whenever it's possible
- For nginx:
  - $request_time in HttpLogModule (milliseconds)

Putting it all together



Putting it all together



Generated with HPA: https://github.com/camposr/HTTP-Performance-Analyzer

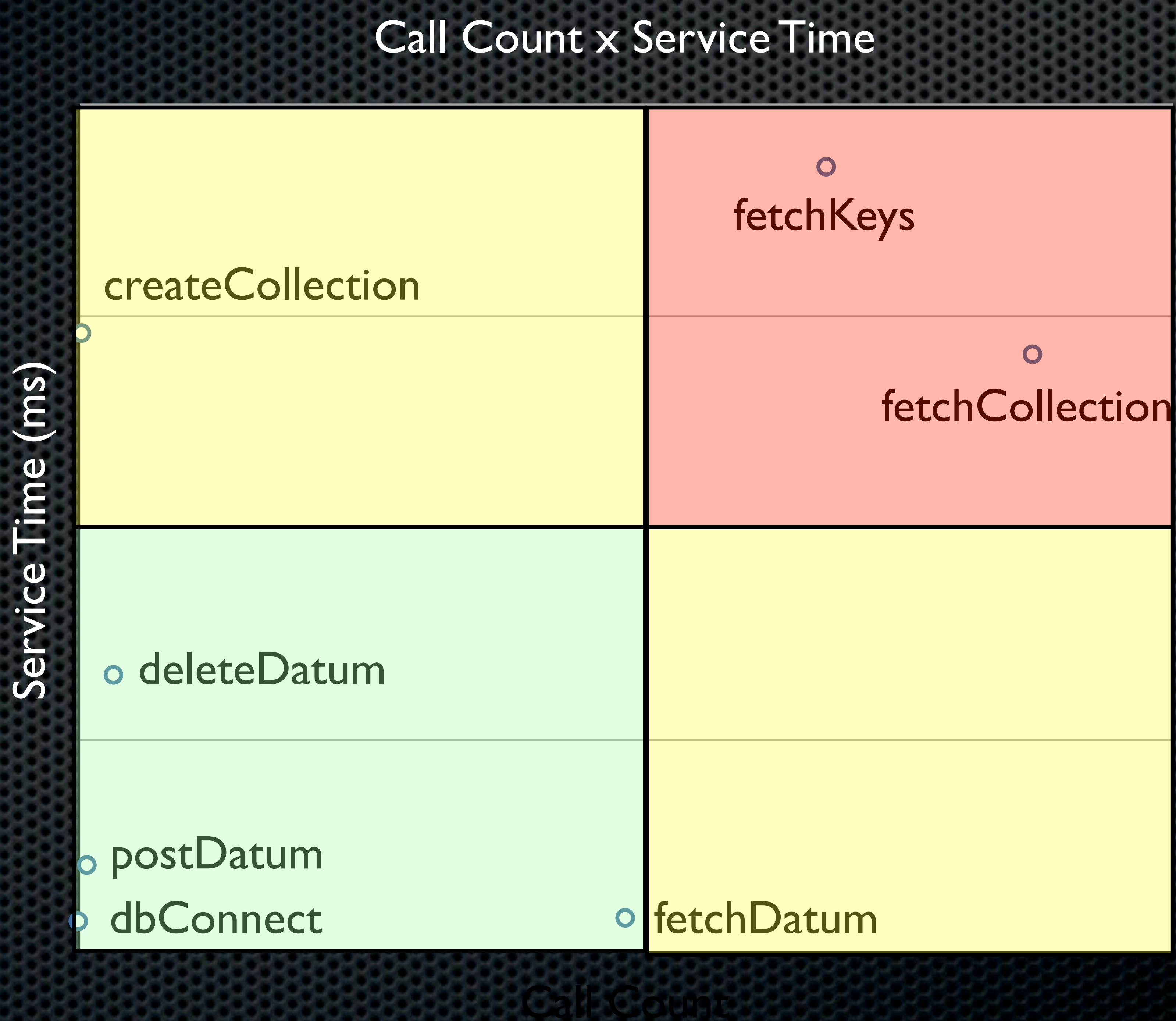
Putting it all together

- A simple tag collection data store
- For each data operation:
    - A 64 bit counter for the number of calls
    - An average counter for the service time

Putting it all together

Method	Call Count	Service Time (ms)
dbConnect	1,876	11.2
fetchDatum	19,987,182	12.4
postDatum	1,285,765	98.4
deleteDatum	312,873	31.1
fetchKeys	27,334,983	278.3
fetchCollection	34,873,194	211.9
createCollection	118,853	219.4

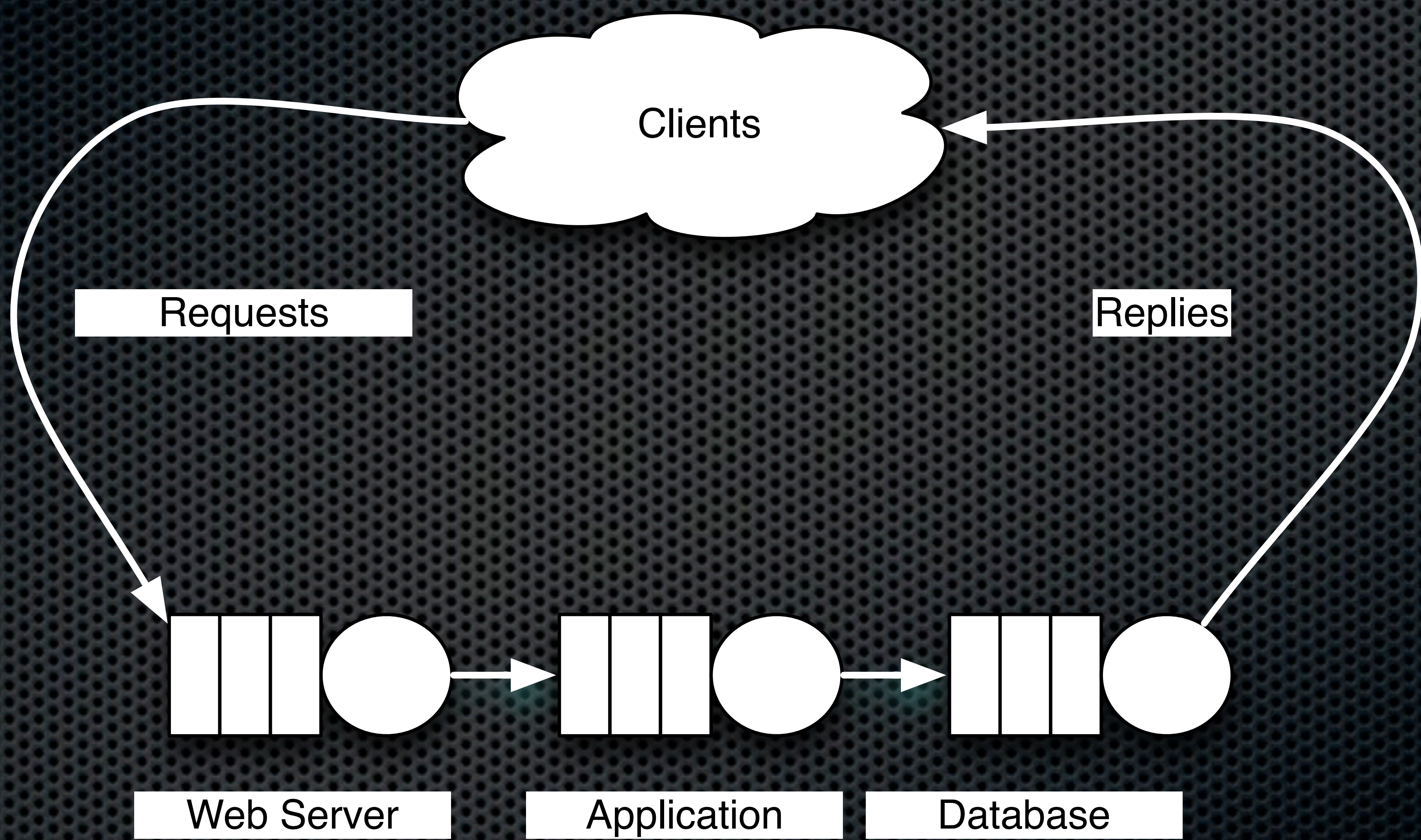
Putting it all together



Modeling

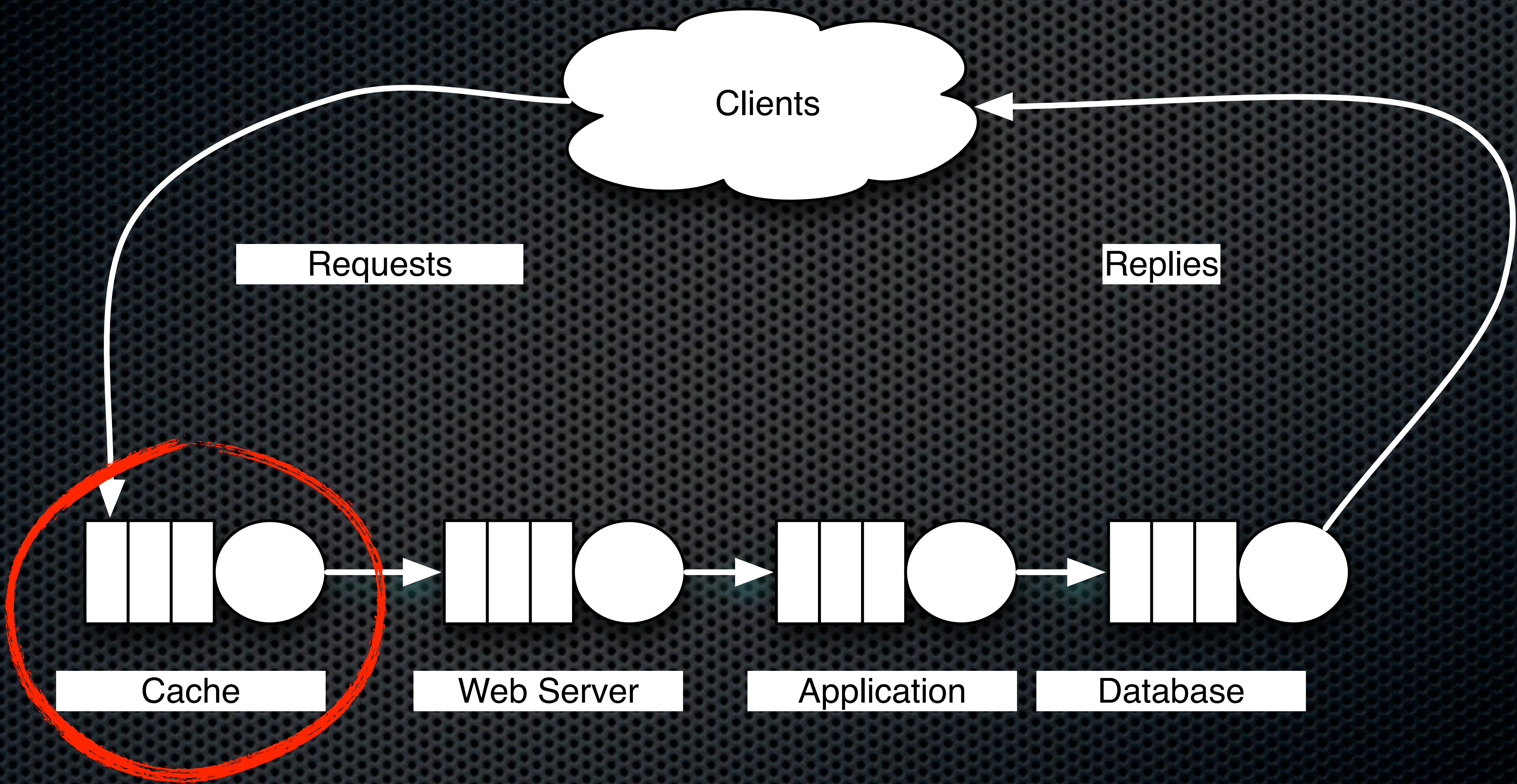
- An abstraction of a complex system
- Allows us to observe phenomena that can not be easily replicated
- “Models come from God, data comes from the devil” - Neil Gunther, PhD.

Modeling



Clients
Requests
Replies
Web Server
Application
Database

Modeling


Clients
Requests
Replies
Cache
Web Server
Application
Database

Modeling

- We're using PDQ in order to model queue circuits
- Freely available at:
  - http://www.perfdynamics.com/Tools/PDQ.html
- Pretty Damn Quick (PDQ) analytically solves queueing network models of computer and manufacturing systems, data networks, etc., written in conventional programming languages.

Modeling

CreateNode()	Define a queuing center
CreateOpen()	Define a traffic stream of an open circuit
CreateClosed()	Define a traffic stream of a closed circuit
SetDemand()	Define the service demand for each of the queuing centers

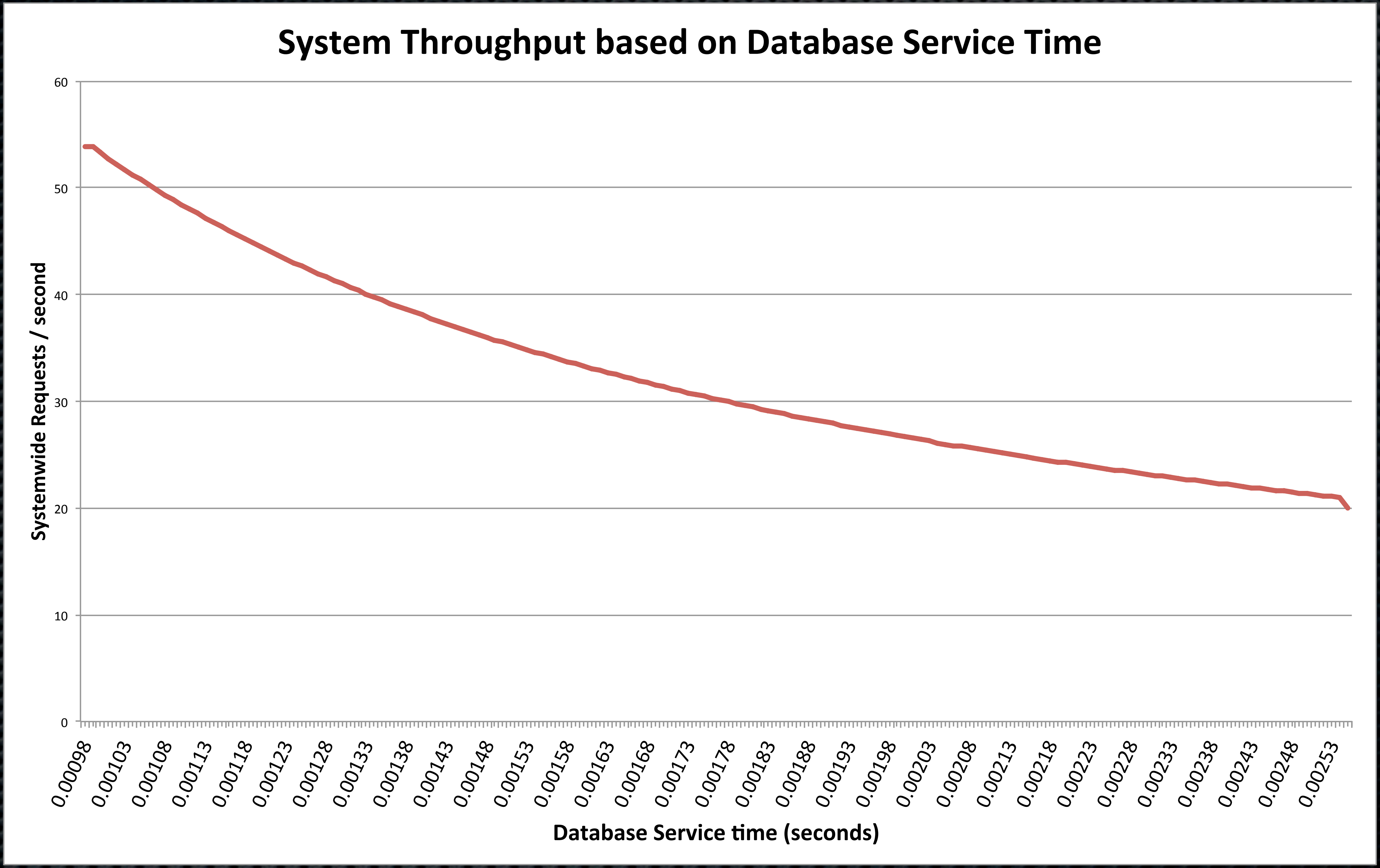
Modeling

```
$httpServiceTime = 0.00019;
$appServiceTime = 0.0012;
$dbServiceTime = 0.00099;
$arrivalRate = 18.762;

pdq::Init("Tag Service");

$pdq::nodes = pdq::CreateNode('HTTP Server', ↵
$pdq::CEN, $pdq::FCFS);
$pdq::nodes = pdq::CreateNode('Application Server',↵
$pdq::CEN, $pdq::FCFS);
$pdq::nodes = pdq::CreateNode('Database Server',↵
$pdq::CEN, $pdq::FCFS);
```

Modeling

```
                ========================================
                ******    PDQ Model OUTPUTS      *******
                ========================================

Solution Method: CANON

                ******    SYSTEM Performance     *******

Metric                        Value    Unit
------                        -----    ----
Workload: "Application"
Number in system             1.3379    Requests
Mean throughput             18.7620    Requests/Seconds
Response time                0.0713    Seconds
Stretch factor               1.5970

Bounds Analysis:
Max throughput              44.4160    Requests/Seconds
Min response                 0.0447    Seconds
```

Modeling


System Throughput based on Database Service Time
Systemwide Requests / second
60
50
40
30
20
10
0
0.00098
0.00103
0.00108
0.00113
0.00118
0.00123
0.00128
0.00133
0.00138
0.00143
0.00148
0.00153
0.00158
0.00163
0.00168
0.00173
0.00178
0.00183
0.00188
0.00193
0.00198
0.00203
0.00208
0.00213
0.00218
0.00223
0.00228
0.00233
0.00238
0.00243
0.00248
0.00253
Database Service time (seconds)

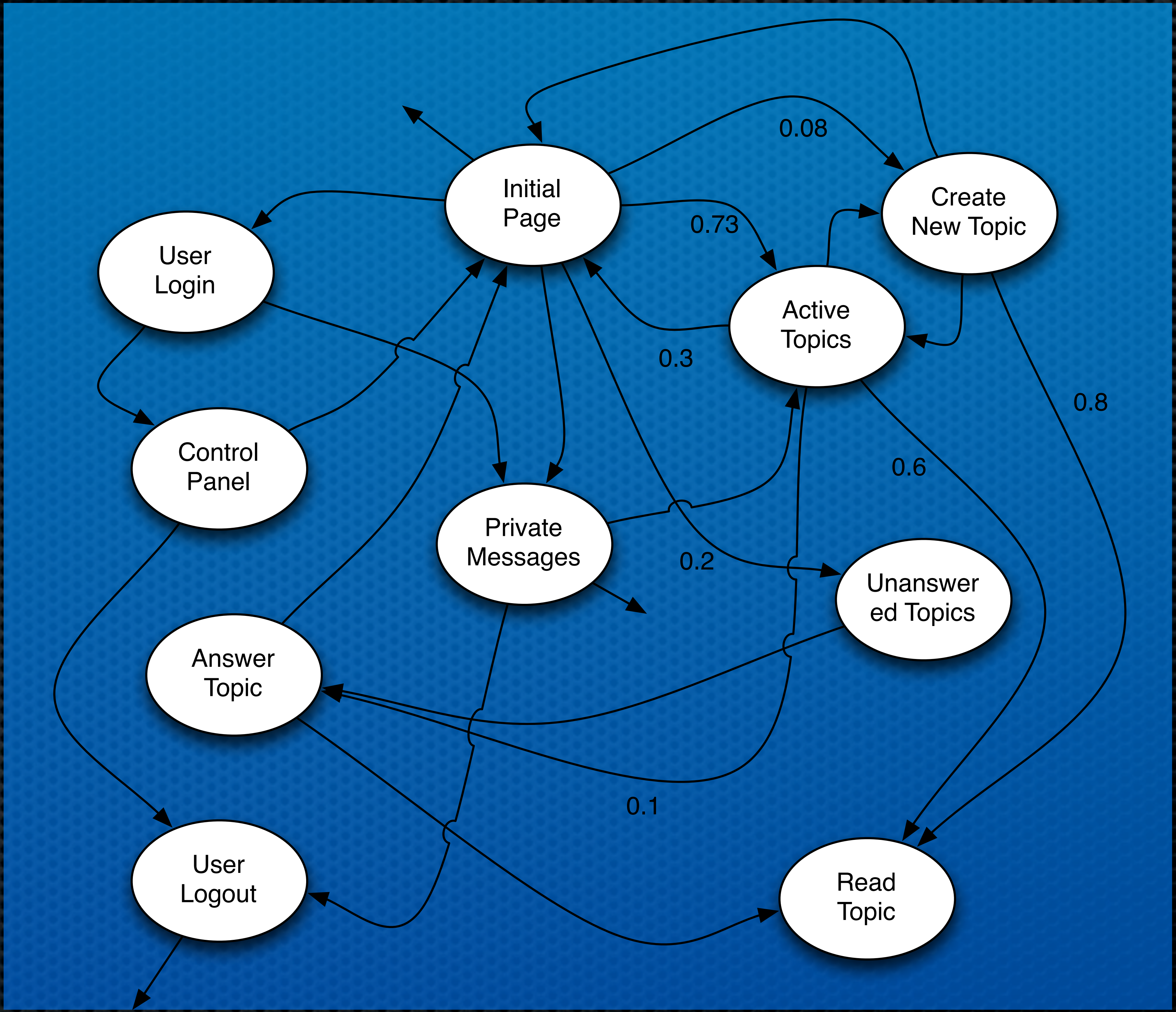
Modeling

- Complete makeover of a web collaborative portal
- Moving from a commercial-of-the-shelf platform to a fully customized in-house solution
- **How high it will fly?**

Modeling

- Customer Behavior Model Graph (CBMG)
  - Analyze user behavior using session logs
  - Understand user activity and optimize hotspots
  - Optimize application cache algorithms

Modeling



Initial Page
User Login
Create New Topic
Active Topics
Control Panel
Private Messages
Unanswered Topics
Answer Topic
User Logout
Read Topic
0.08
0.73
0.3
0.8
0.6
0.2
0.1

Modeling

- Now we can mimic the user behavior in the newly developed system
- The application was instrumented so we know the service time for every method
- Each node in the CBMG is mapped to the application methods it is related

References

- Using a Queuing Model to Analyze the Performance of Web Servers - Khaled M. ELLEITHY and Anantha KOMARALINGAM
- A capacity planning / queueing theory primer - Ethan D. Bolker
- Analyzing Computer System Performance with Perl::PDQ - N. J. Gunther
- Computer Measurement Group Public Proceedings

Questions answered here

Thanks for attending !

Rodrigo Campos

camposr@gmail.com

http://twitter.com/xinu

http://capacitricks.posterous.com