Utilization is Virtually Useless as a Metric!

Adrian Cockcroft

Netflix Inc.

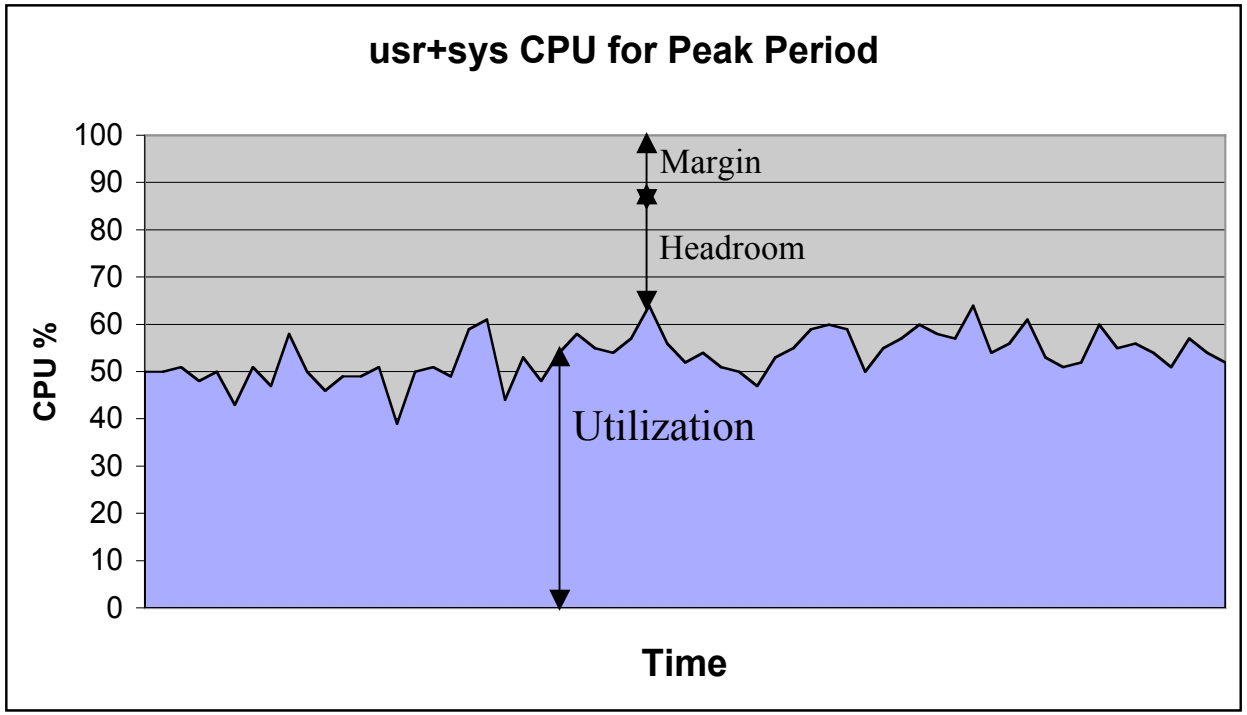
Agenda

- Headroom
- Utilization
- Response Time
- The Many Ways In Which Utilization Metrics Are Broken
- An Alternative
- Conclusions

NETFLIX

Headroom

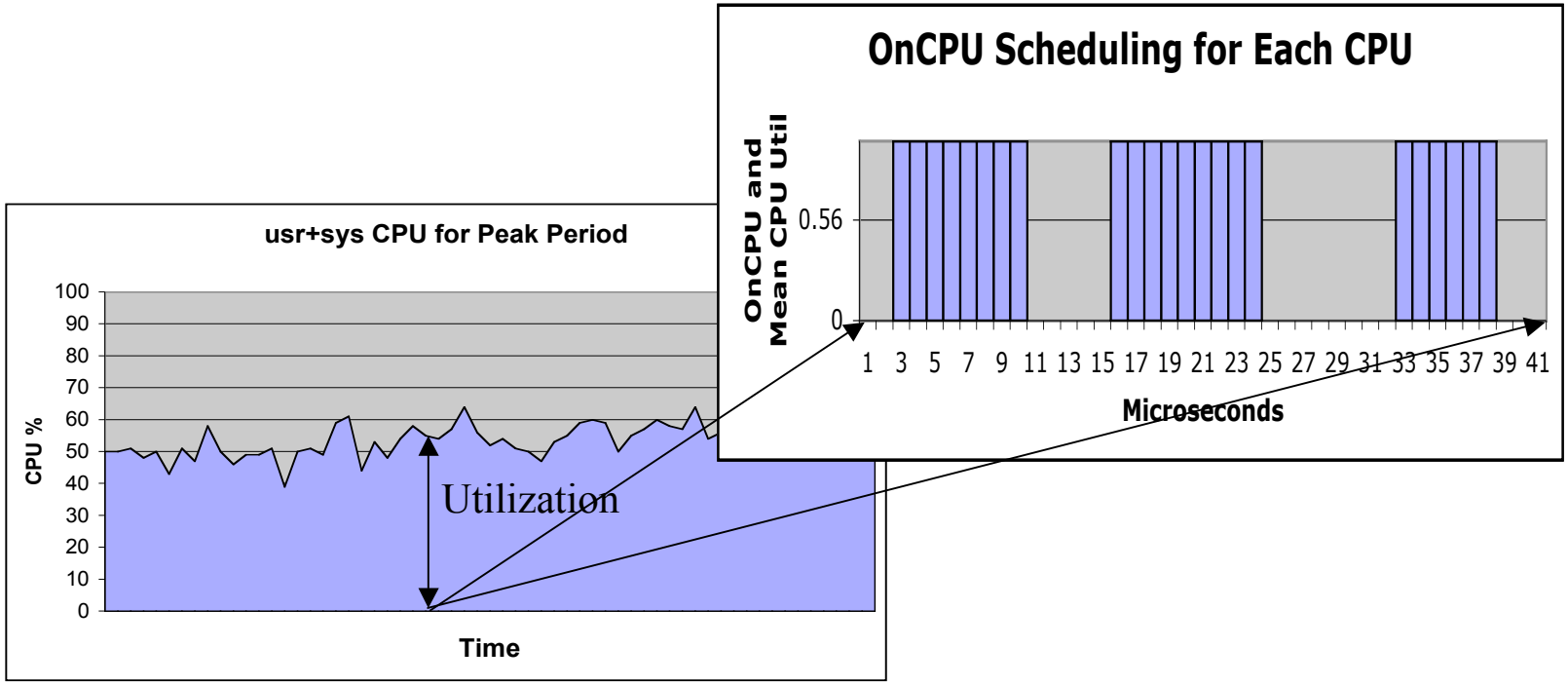
- Headroom is available usable resources
  - Total Capacity minus Peak Utilization and Margin
  - Applies to CPU, RAM, Net, Disk and OS



NETFLIX

Utilization

- Utilization is the proportion of busy time
- Always defined over a time interval



NETFLIX

Response Time

- Response Time = Queue time + Service time
- **The Usual Assumptions…**
  - **Steady state averages**
  - **Random arrivals**
  - **Constant service time**
  - **M servers processing the same queue**
- Approximations
  - Queue length = Throughput x Response Time
    - (Little's Law)
  - Response Time = Service Time / (1 - Utilization^{M})

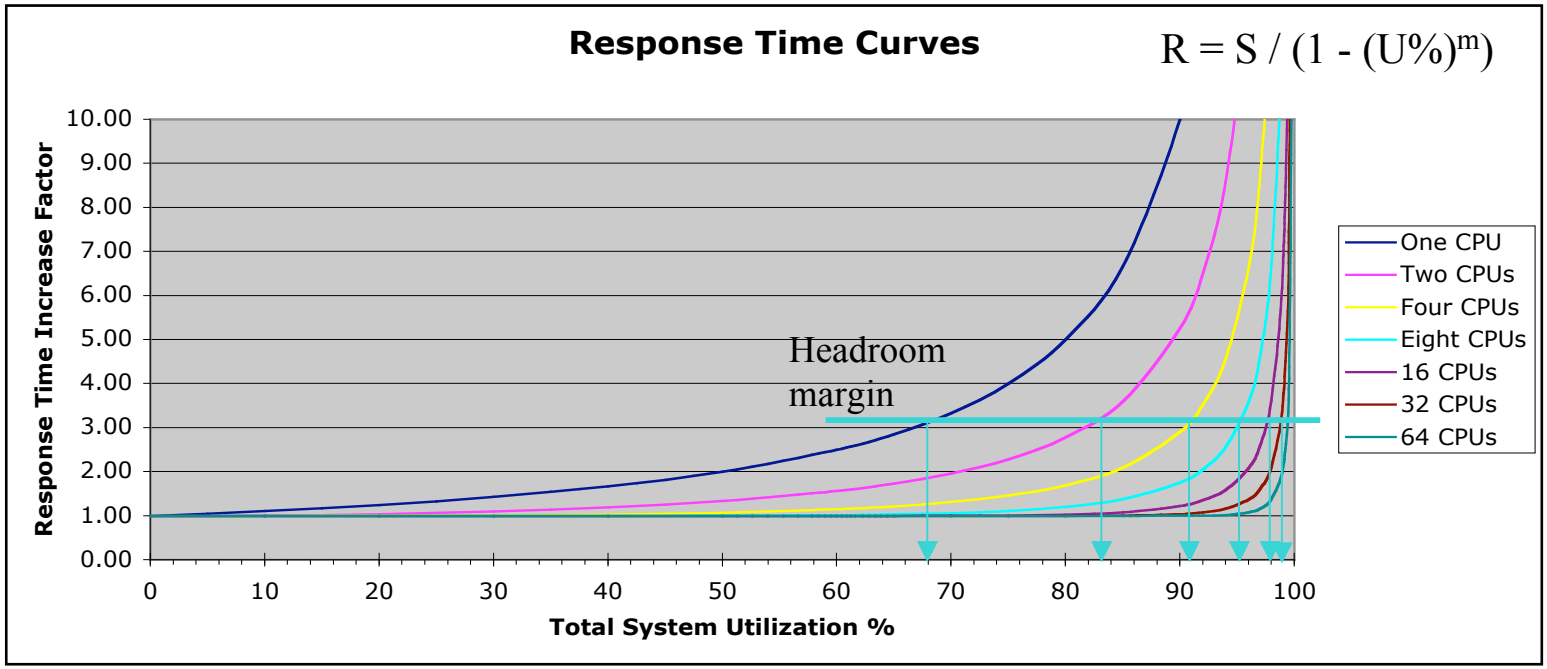
NETFLIX

Response Time Curves

The traditional view of Utilization as a proxy for response time

Systems with many CPUs can run at higher utilization levels, but degrade more rapidly when they run out of capacity

Headroom margin should be set according to a response time target.



NETFLIX

So what's the problem with Utilization?

- Unsafe assumptions! Complex adaptive systems are not simple!
- Random arrivals?
  - Bursty traffic with long tail arrival rate distribution
- Constant service time?
  - Variable clock rate CPUs, inverse load dependent service time
  - Complex transactions, request and response dependent
- M servers processing the same queue?
  - Virtual servers with varying non-integral concurrency
  - Non-identical servers or CPUs, Hyperthreading, Multicore, NUMA
- Measurement Errors?
  - Mechanisms with built in bias, e.g. sampling from the scheduler clock
  - Platform and release specific systemic changes in accounting of interrupt time

NETFLIX

Threaded CPU Pipelines

- CPU microarchitecture optimizations
  - Extra register sets working with one execution pipeline
  - When the CPU stalls on a memory read, it switches registers/threads
  - Operating system sees multiple schedulable entities (CPUs)
- Intel Hyperthreading
  - Each CPU core has an extra thread to use spare cycles
  - Typical benefit is 20%, so total capacity is 1.2 CPUs
  - I.e. Second thread much slower when first thread is busy
  - Hyperthreading aware optimizations in recent operating systems
- Sun “CoolThreads”
  - "Niagara" SPARC CPU has eight cores, one shared floating point unit
  - Each CPU core has four threads, but each core is a very simple design
  - Behaves like 32 slow CPUs for integer, snail like uniprocessor for FP
  - Overall throughput is very high, performance per watt is exceptional
  - New Niagara 2 has dedicated FPU and 8 threads per core (total 64 threads)

Variable Clock Rate CPUs

- Laptop and other low power devices do this all the time
  - Watch CPU usage of a video application and toggle mains/battery power….
- Server CPU Power Optimization - AMD PowerNow!™
  - AMD Opteron server CPU detects overall utilization and reduces clock rate
  - Actual speeds vary, but for example could reduce from 2.6GHz to 1.2GHz
  - Changes are not understood or reported by operating system metrics
  - Speed changes can occur every few milliseconds (thermal shock issues)
  - Dual core speed varies per socket, Quad core varies per core
  - Quad core can dynamically stop entire cores to save power
- Possible scenario:
  - You estimate 20% utilization at 2.6GHz
  - You see 45% reported in practice (at 1.2GHz)
  - Load doubles, reported utilization drops to 40% (at 2.6GHz)
  - Actual mapping of utilization to clock rate is unknown at this point
- Note: Older and "low power" Opterons used in blades fix clock rate

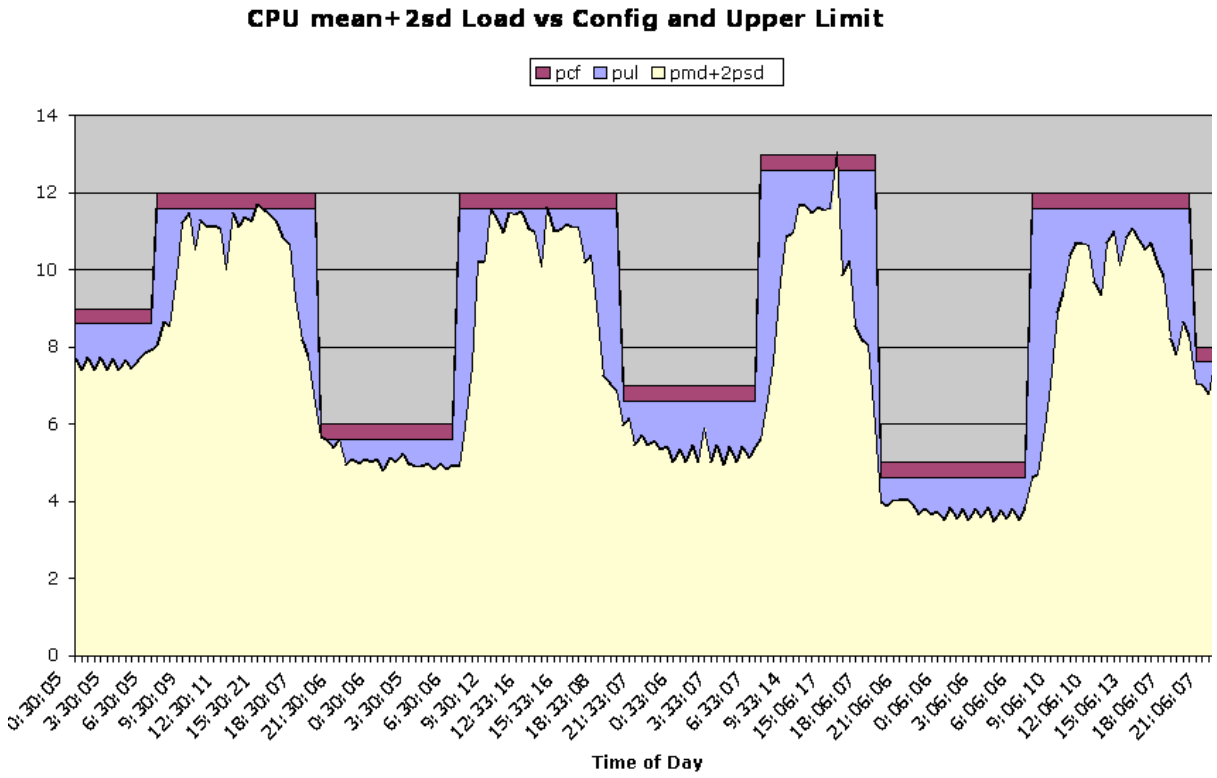
NETFLIX

Virtual Machine Monitors

- VMware, Xen, IBM LPARs etc.
  - Non-integral and non-constant fractions of a machine
  - Naiive operating systems and applications that don't expect this behavior
  - However, lots of recent tools development from vendors
- Average CPU count must be reported for each measurement interval
- VMM overhead varies, application scaling characteristics may be affected

How to plot Headroom

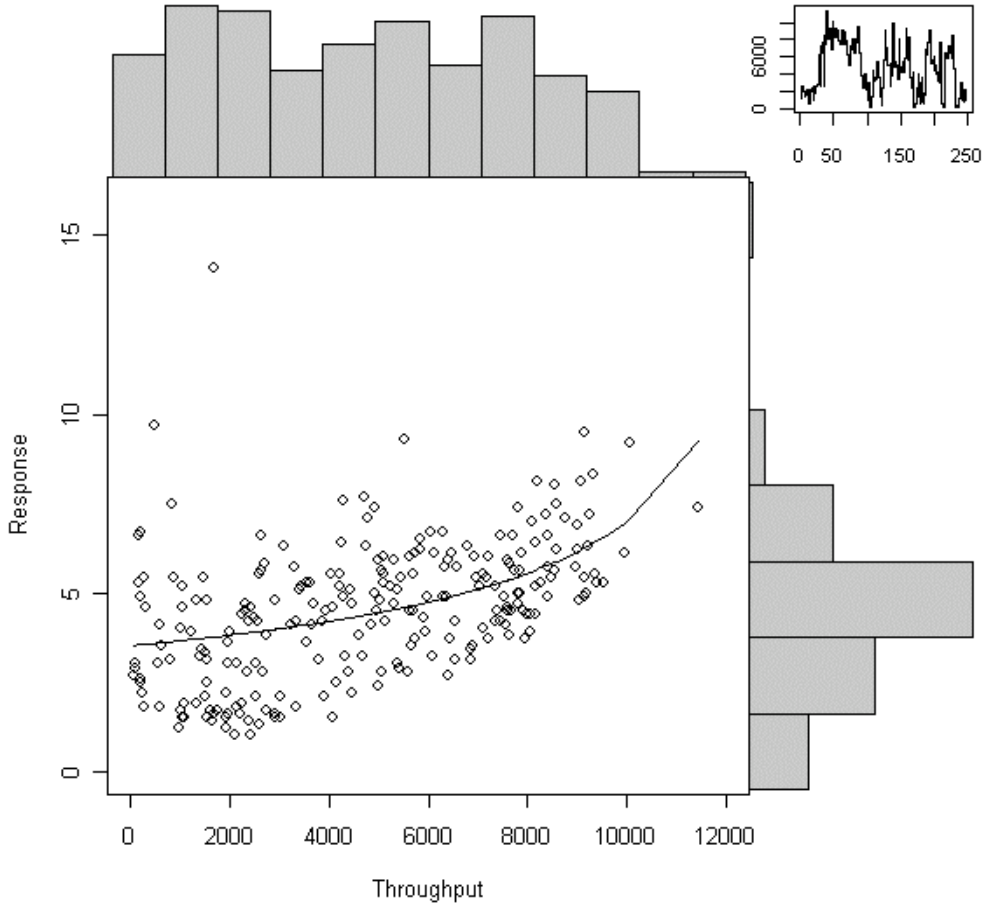
- Measure and report absolute CPU power if you can get it…
- Plot shows headroom in blue, margin in red, total power tracking day/night workload variation, plotted as mean + two standard deviations.



NETFLIX

“Cockcroft Headroom Plot”

- Scatter plot of response time (ms) vs. Throughput (KB) from iostat metrics
- Histograms on axes
- Throughput time series plot
- Shows distributions and shape of response time
- Fits throughput weighted inverse gaussian curve
- Coded using "R" statistics package
- Blogged development at
  http://perfcap.blogspot.com/search?q=chp

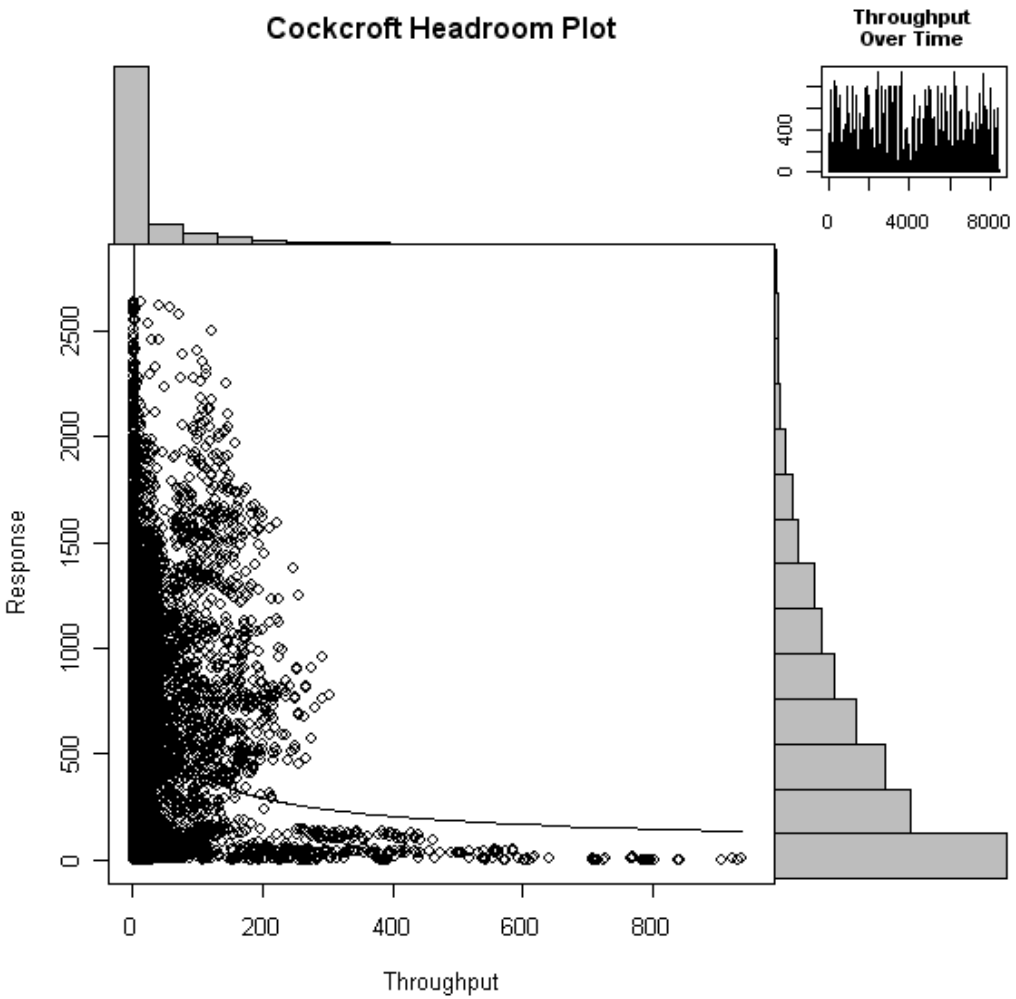


NETFLIX

Response Time vs. Throughput

- A different problem…
- Thread-limited appserver
- CPU utilization is low
- Measurements are of a single SOA service pool
- Response is in milliseconds
- Throughput is executions/s

```
Exec                 Resp
Min.    :     1.00   Min.    :     0.0
1st Qu.:     2.00   1st Qu.:   150.0
Median :     8.00   Median :   361.0
Mean    :    64.68   Mean    :   533.5
3rd Qu.:    45.00   3rd Qu.:   771.9
Max.    :10795.00   Max.    :19205.0
```



NETFLIX

Conclusion

- Check your assumptions…
- Record and plot absolute capacity for each measurement interval
- Plot response time as a function of throughput, not just utilization
- SOA response characteristics are complicated…
- More detailed discussion in my CMG06 Paper

Questions?

acockcroft@netflix.com

http://perfcap.blogspot.com/search?q=utilization

http://perfcap.blogspot.com/search?q=chp

NETFLIX