Distributed Systems
Performance

2014-09-30

Cristian Klein, Ewnetu Bayuh
Department of Computing Science
Umeå University

Outline

- Why do performance evaluation?
- Performance metrics
- System
- Workloads
- Analytic models

Instead of introduction

“Amazon found every 100ms of latency cost them **1% in sales**”

“Bing found that a 2 second slowdown changed queries/user by **-1.8%**”

“Diablo III launch **overwhelms** Blizzard servers”

Why understand performance?

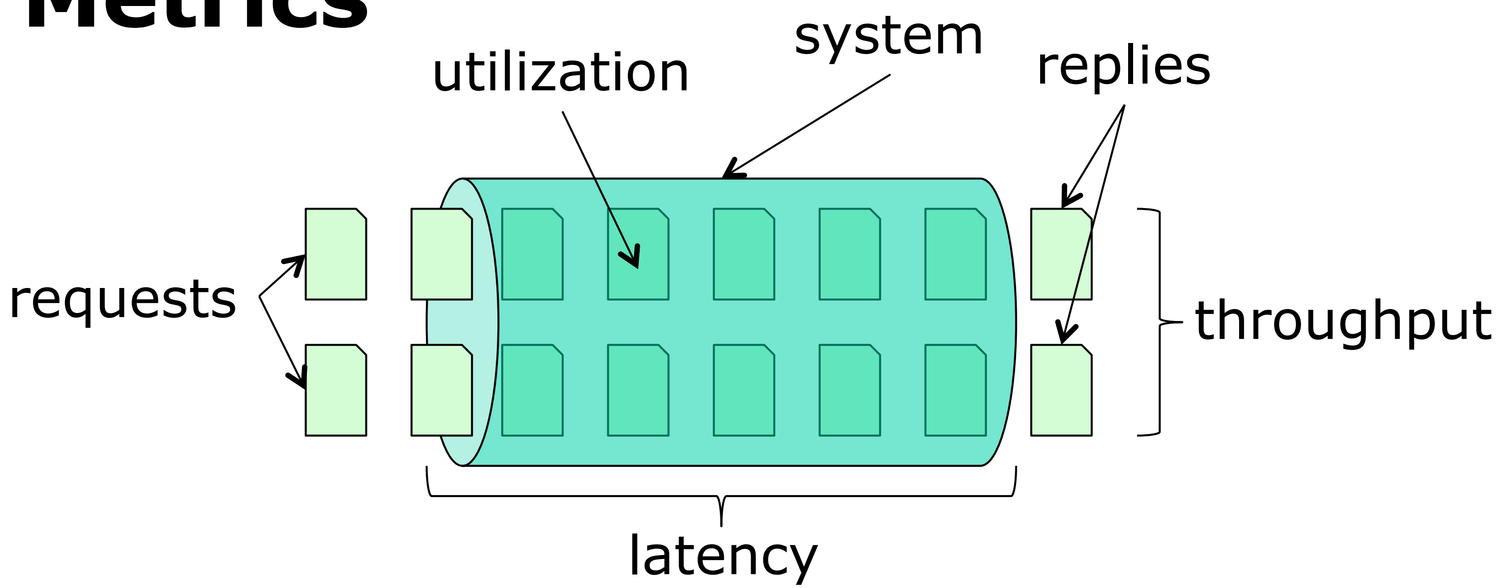
- To make users happy
- To provision computing resources
  - Manually
  - Auto-scaling à la Clouds
- To detect faulty components
  - Limplock: some component is sloooow
- Auto-tuning
  - System has some knobs
  - How to auto-adjust those for peak performance
  - E.g., number of worker threads

Questions to ask?

- What do we want to improve?
- How do we measure that?
- In what conditions?

- Metrics
- System
- Workload

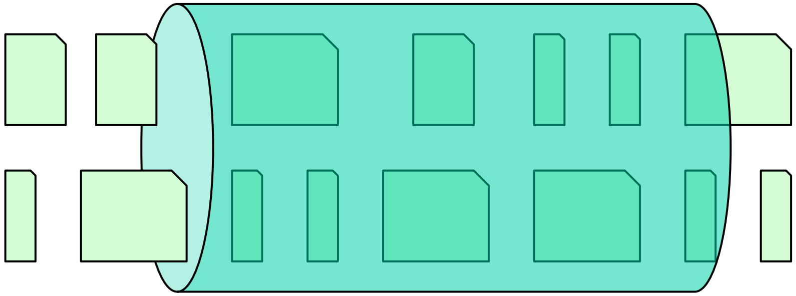
Metrics



Request = message sent to a process, needing processing

- Response time (latency)
  - Amount of time to serve a request
  - End-to-end latency (includes e.g., network latency)
- Throughput
  - Number of requests per time interval
- Utilization
  - Percentage of time the system is busy serving requests

Metrics (cont.)



- A lot of noise
  - Requests of different nature
  - OS noise
  - Caching, etc.
- Do statistics over some interval (1 second)
  - Average, minimum, maximum
  - Distribution

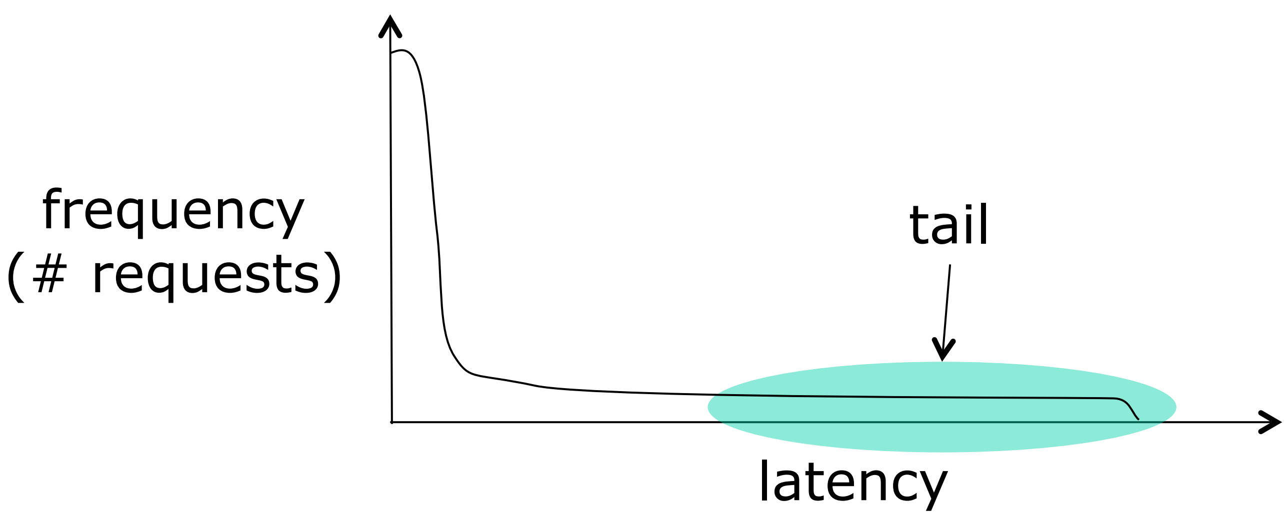
Latency (1/2)

Operation	Latency (ns)
L1 cache reference	0.5
Branch mispredict	5
L2 cache reference	7
Mutex lock/unlock	25
Main memory reference	100
Compress 1K bytes with Zippy	3,000
Send 2K bytes over 1 Gbps network	20,000
Read 1 MB sequentially from memory	250,000
Round trip within same datacenter	500,000
Disk seek	10,000,000
Read 1 MB sequentially from disk	20,000,000
Send packet CA->Netherlands->CA	150,000,000

Latency numbers every programmer should know, Jeff Dean
(http://research.google.com/people/jeff/)

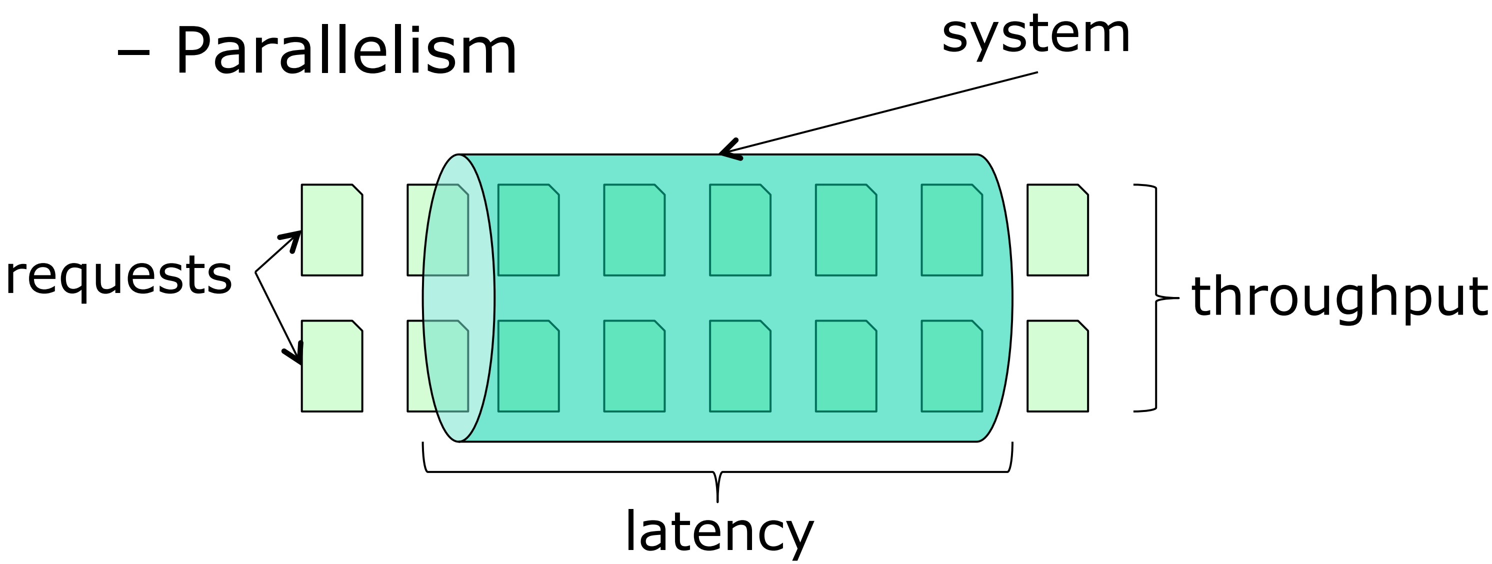
Latency (2/2)

- Average latency
- Tail of distribution
  - 95th percentile
  - 99th percentile
  - Maximum



Throughput

- Number of request per second
- Related to latency, but not the same due to
  - Pipelining
  - Parallelism



Utilization

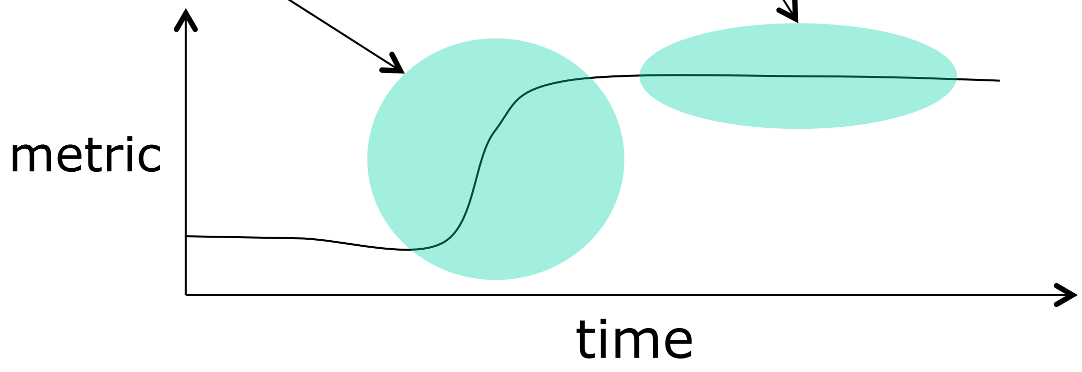
- Percentage of time the system was busy serving requests
- Examples:
  - 60% CPU utilization
  - 100% disk utilization
- Indication of spare capacity
- Spot bottlenecks

Cold vs. warm

- Cold
  - System just booted
  - Caches are empty
    - E.g., database read from disk instead of memory
  - System did not adapt to workload
    - E.g., processes need to be fork()-ed

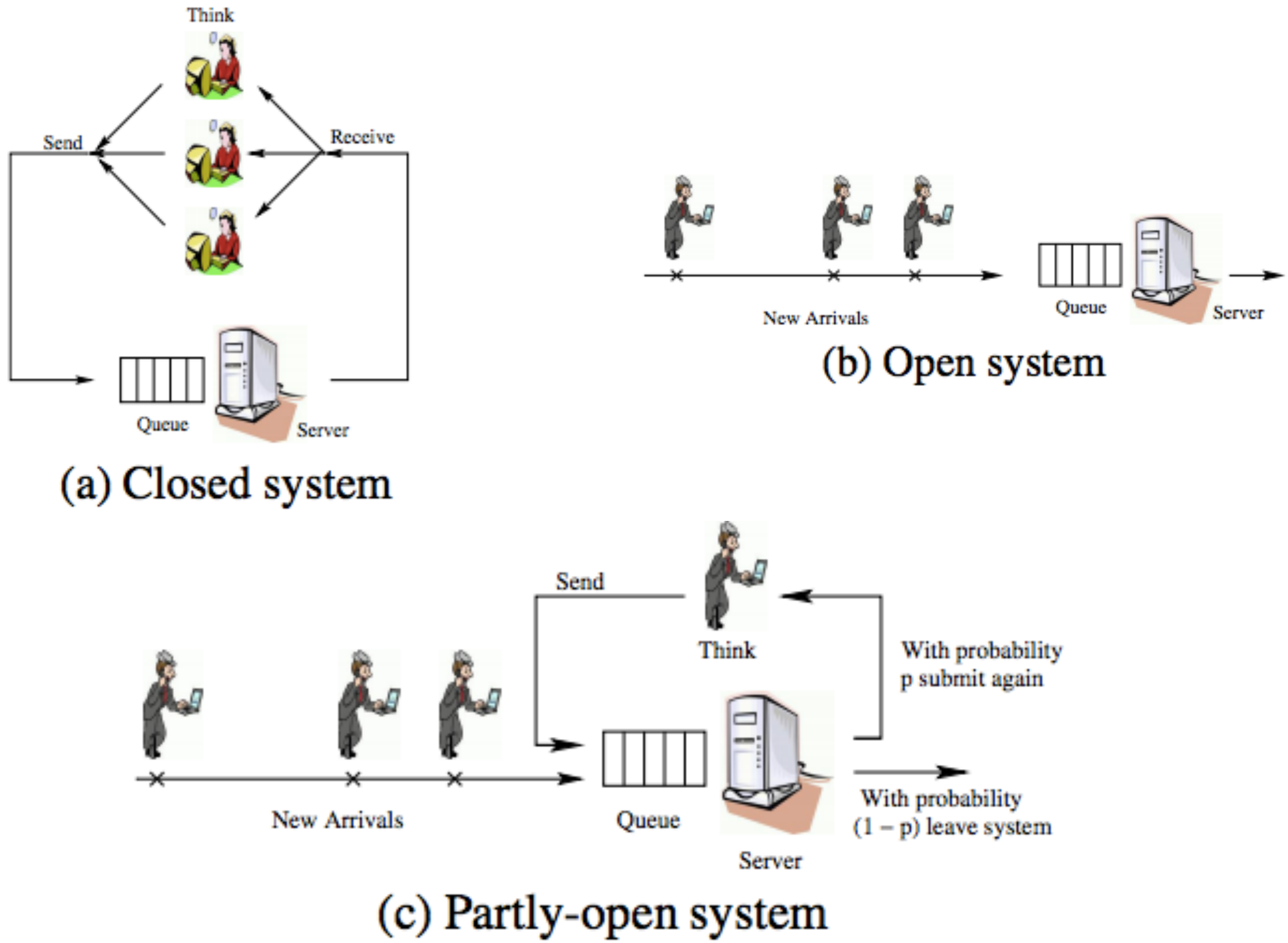
- Warm
  - System ran for some time
  - Filled caches, adapted to workload
  - Generally faster than cold

Transient- vs. steady-state



- Transient-state
  - Some parameters just changed
    - E.g., number of users, distribution of requests, number of CPUs, …
  - System needs time to adapt
    - E.g., refill caches, create/destroy processes
- Steady-state
  - Parameters do not change
  - System adapted to peak-performance
  - Generally faster than transient-state

Workload: closed vs. open



Bianca Schroeder et al, Open versus closed: a cautionary tale, NSDI'06

Workloads: distribution of requests

- E.g., Video sharing website (YouTube)
- Some videos are more popular
- Cache them in memory
  - Improves performance
- Testing the system with a uniform distribution would hide any potential improvements

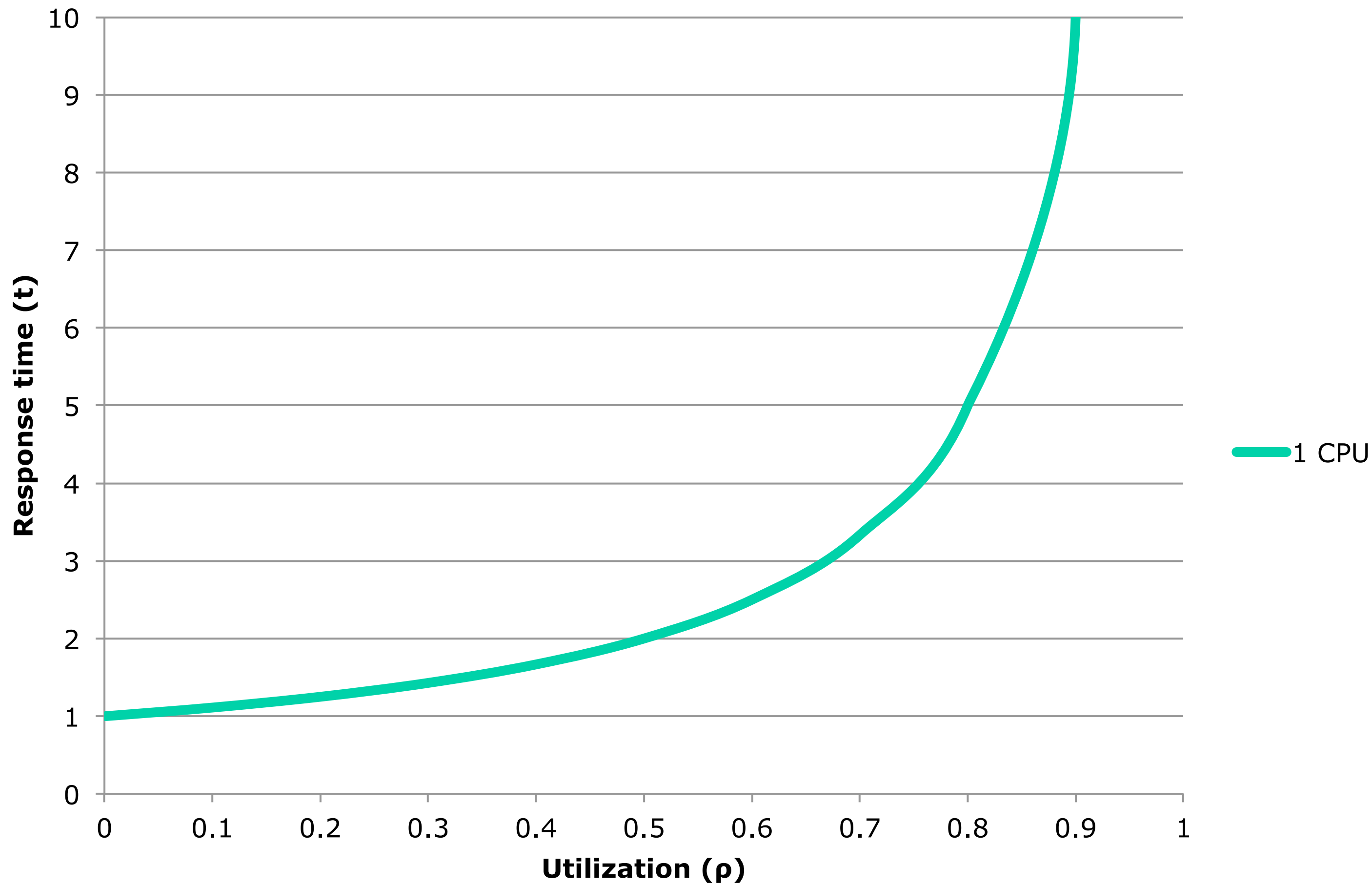
Queuing theory

- Analytic formula for metrics
  - Assumes a certain model (i.e., a simplification of the actual system)
- Helpful to bridge theory with practice
  - Either the model is too simple
  - Or you have a performance bug in your code

M/M/1 queue

- λ – arrival rate [requests / s]
- μ – service rate [requests / s]
- Utilization $\rho = \frac{\lambda}{\mu}$
- Response time $t = \frac{1}{\mu - \lambda}$
- What if μ < λ?

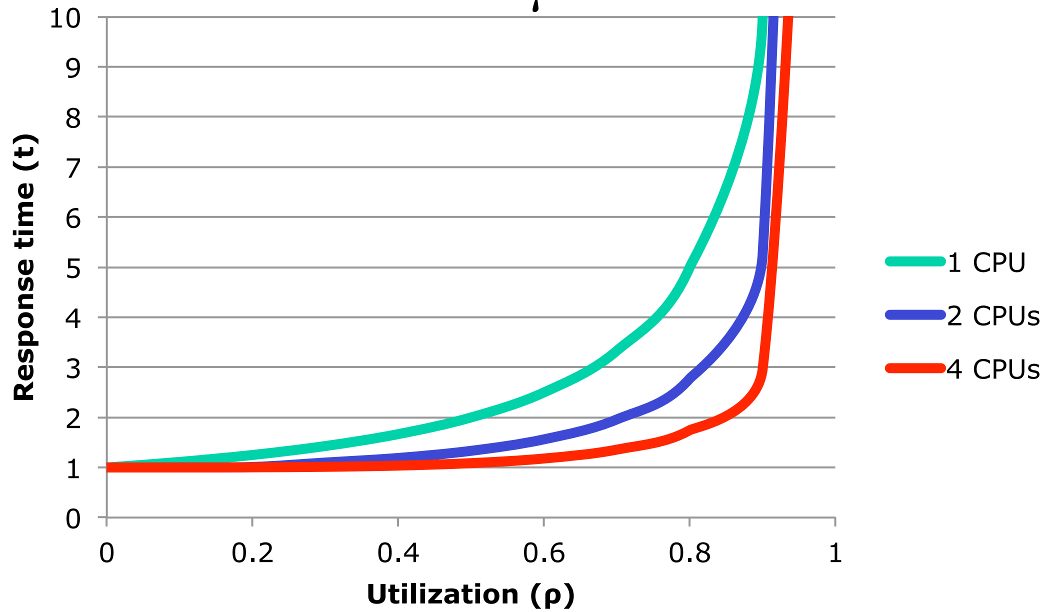
M/M/1 queue (cont.)



M/M/c queue

- c concurrent "servers" (e.g., CPUs)

- Utilization $\rho = \frac{c\lambda}{\mu}$



Summary

- Why do performance evaluation?
  - Make users happy, provisioning, etc.
- Metrics
  - Response time, throughput, utilization
- System issues
  - Cold vs. hot, transient vs. steady
- Workload
  - Closed vs open, distribution of requests
- Queuing theory
  - M/M/c queues

“In theory, …

- theory and practice are the same. In practice, they are not.”
- Due to
  - Caching effects
  - Context switches
  - OS noise
  - Lock contention
- We need to measure!