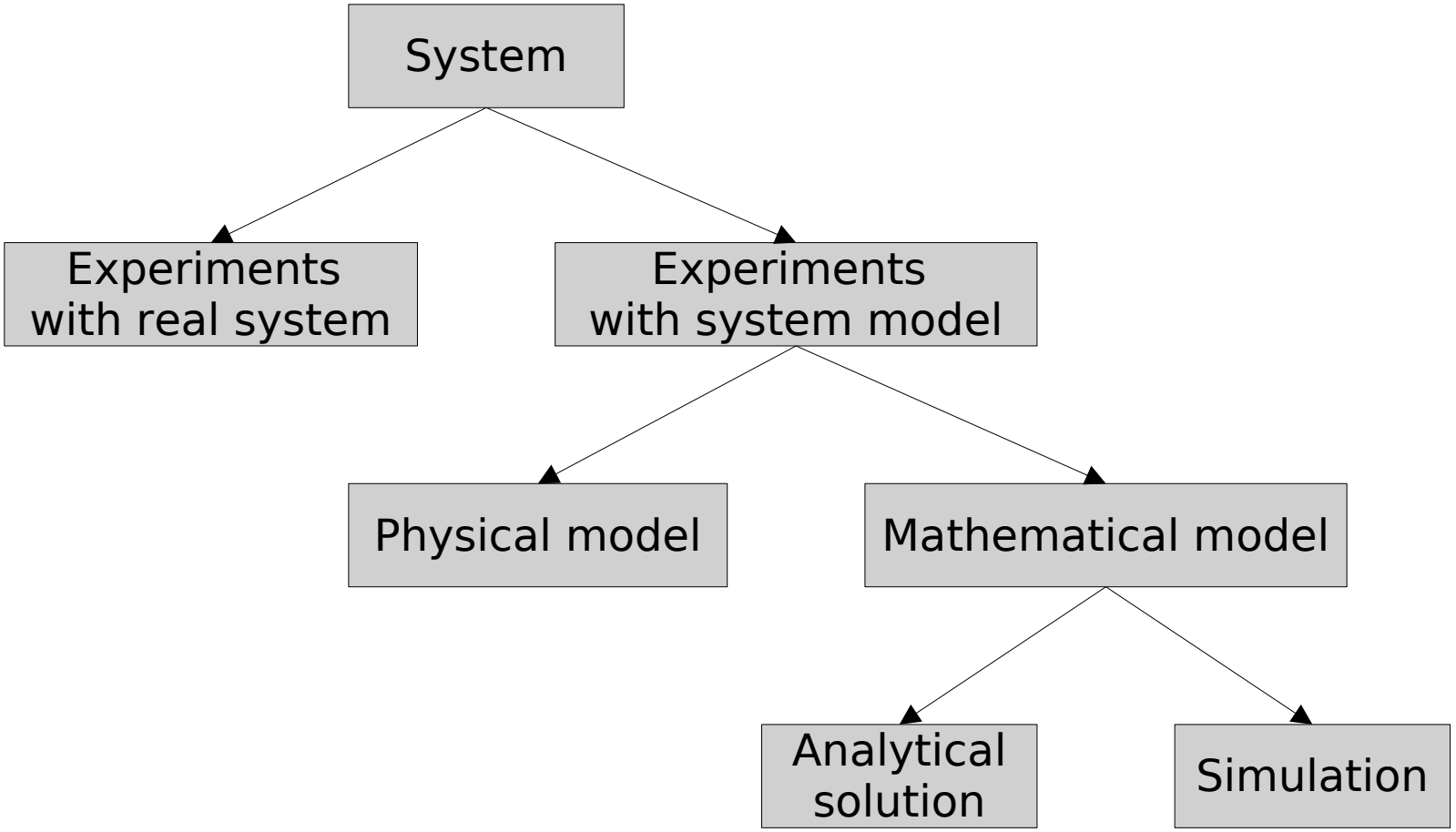
UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Performance evaluation, capacity planning and reliability

UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Performance evaluation

Possible approaches



Possible approaches

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Real system
  - Full-scale system
  - Testing and Benchmarking
- Physical model
  - Small scale prototype
  - Testing and benchmarking
  - Scaling up results
  - May be tricky in case of non-linear scalability

Possible approaches

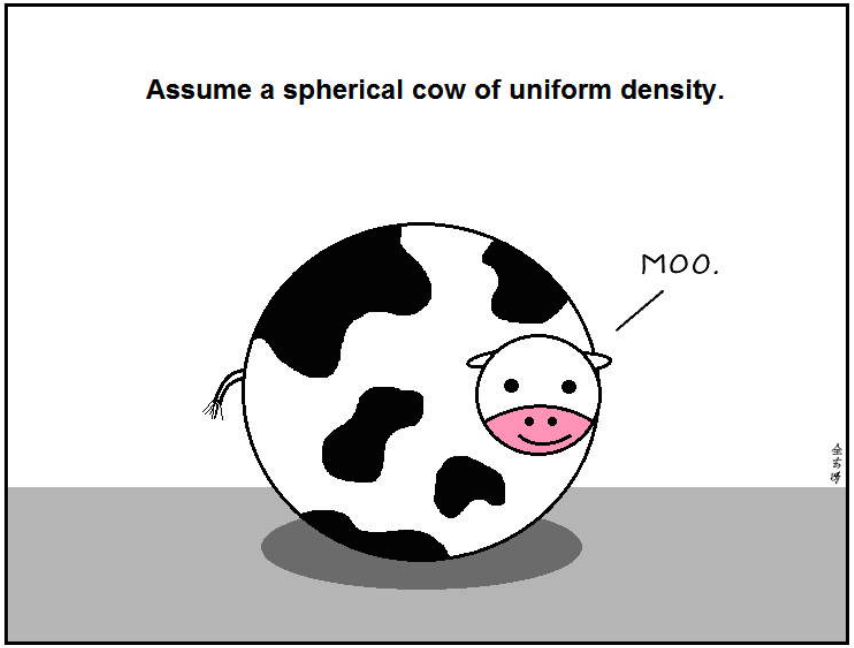
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Mathematical model
  - Equations that describe system behavior
  - Easy to handle
  - What-if scenarios, comparison fo alternatives
  - Part of management algorithms
  - Simplified system description
- Simulator
  - Software that implement system behavior
  - More complex and realistic than mathematical model

Comparison of approaches

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Actual system *vs*. System model
  - Easier to do/Less dangerous (if system exists)
  - Less expensive than building a real system
- Physical *vs*. Mathematical model
  - Easier and less expensive to implement
  - Easier to modify
  - Mathematical model must be accurate enough
  - Remember the Spherical Cow
- Analytical solution *vs.* Simulation
  - Analytical is more precise
  - Simulation is more flexible



Motivational examples

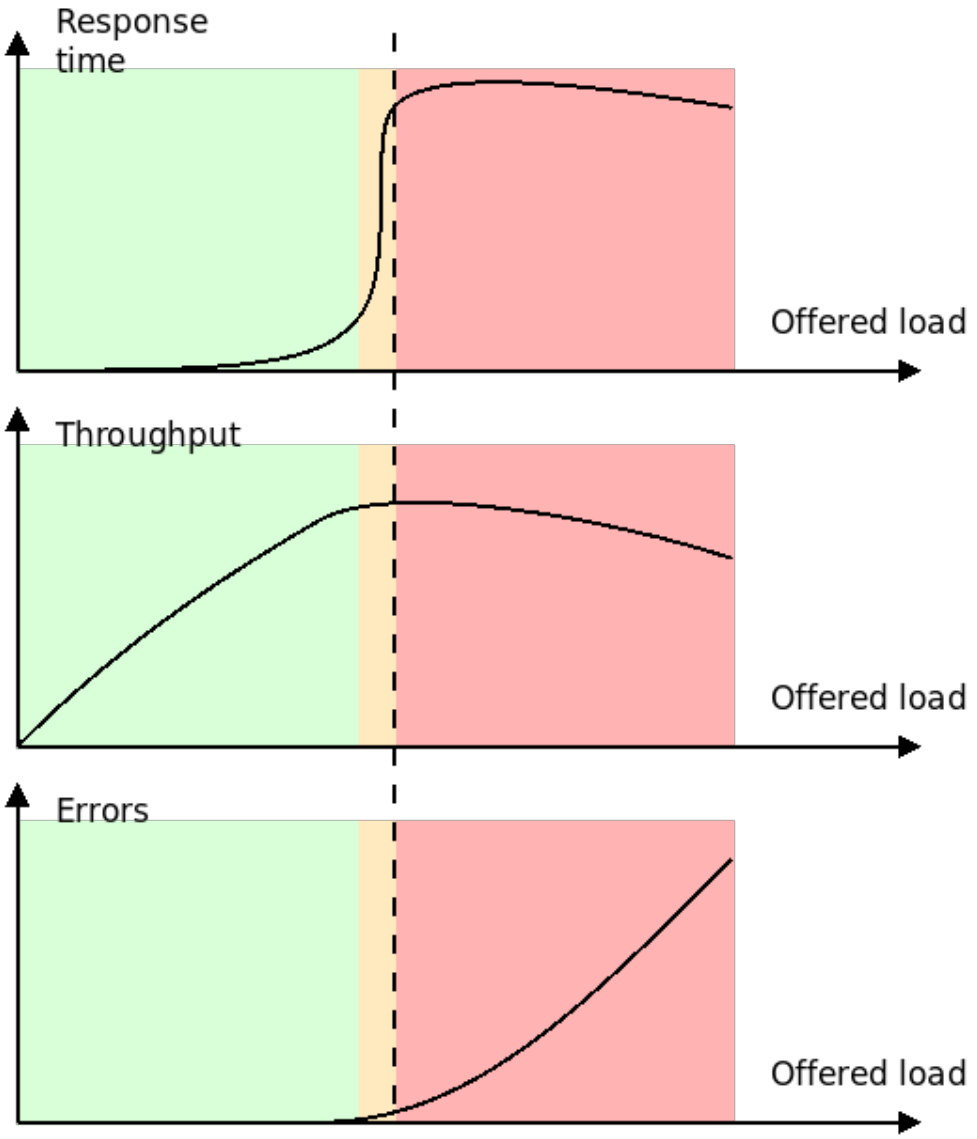
Performance parameters

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Response time:
  - Elapsed time between
    - Beginning of request
    - End of Response
- Throughput:
  - Number of requests satisfied within a time unit
- Error rate:
  - Fraction fo requests not server correctly
- Parameters depend on current load

Performance parameters

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA



Operating conditions

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- OK/Underutilized (Green)
  - Response time is low
  - Response time scarcely correlated with load
  - Throughput increases linearly with offered load
  - Error rate remains low
- Knee region (Yellow)
  - Response time increases dramatically with load
  - Throughput saturates
  - Error rate starts to grow

Operating conditions

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Overload/Thrashing (Red)
  - Response time explodes
  - Response time limited by timeout
  - Throughput is constant (overload)
  - Throughput drops (trashing)
  - High error rate
- Our goal:

  → remaining at the end of the green zone

The cost of errors

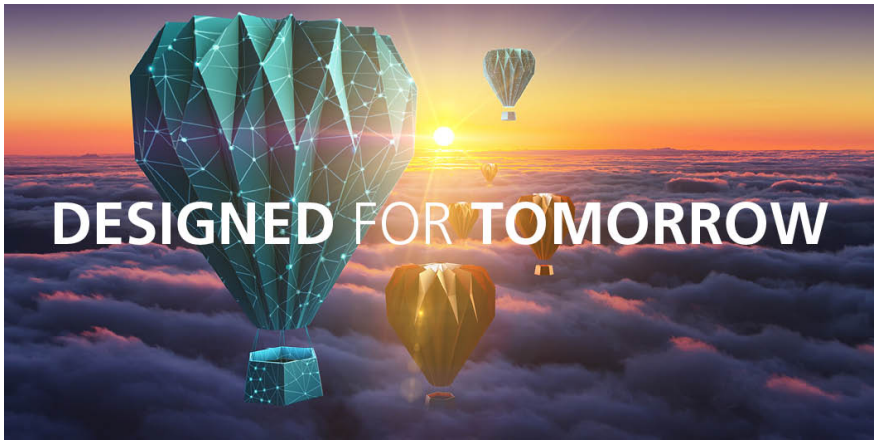
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Configuration errors have a cost
- A mis-configured system can be:
- Poor performing
  - 60% slowdown observed in cloud deployments
  - A single bottleneck can slow down a huge chain of services
- Expensive
  - Performance are OK, but cost is huge
  - 10× cost increase observed in cloud deployments

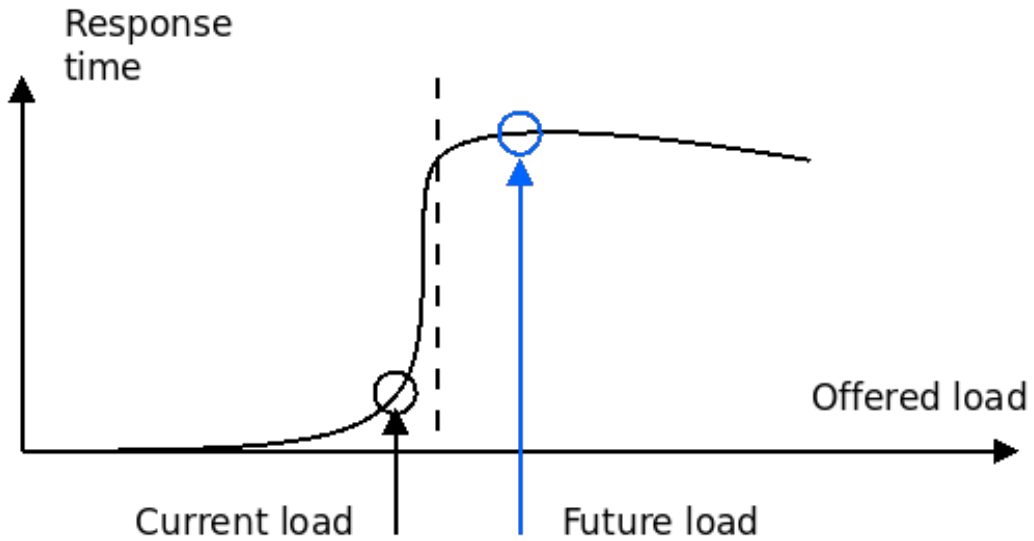
Capacity planning

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- A system may be running fine
  - Today
  - Tomorrow?
- What about workload evolution?
- What if the load increases by *X*%?
  - Are we still providing acceptable performance
  - If not, how do we cope with this?

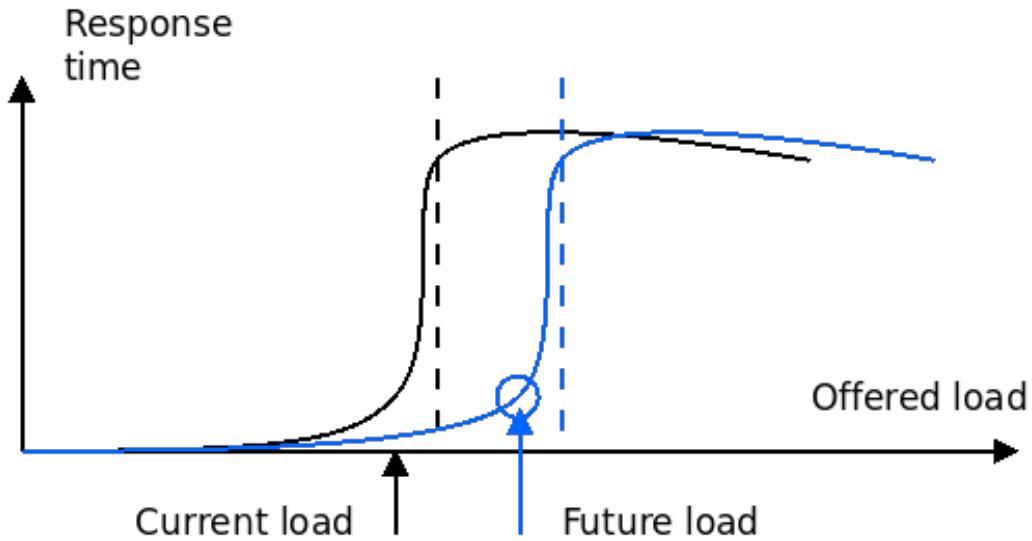


Load evolves



- Current situation:
  - Ideal operating condition
- Future situation
  - High risk of overload

Capacity planning



- Scaling is required
- New provisioning
  - How much more processing power do we need?
  - How to design the new infrastructure

Elasticity done well

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Cloud computing is elastic
  - Can purchase new computing power as needed
- We still need to understand performance:
  - How many VMs?
  - Which size of VMs?
  - For how much time?
  - What resource is the performance bottleneck?
- Cloud without understanding performance ~~*can be*~~ *is*
  - Tricky
  - Expensive
- Would risk your credit card?

Another (unfortunate) case

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Scaling based on a model
  - Distributed infrastructure
  - Compute number of replicas to meet SLA
- Use of a (over-)simplified model
  - M/M/1 queuing model

$$N = \sum_{j \in \mathcal{F}} E_j \geq \left\lceil \frac{\Lambda}{\overline{\mu}} \cdot \frac{K-1}{K} \right\rceil$$

  - Neglecting variance of service time variable
- What happens if the service time is highly variant
  - Common scenario: high variance
  - Should use log-normal distribution to describe service

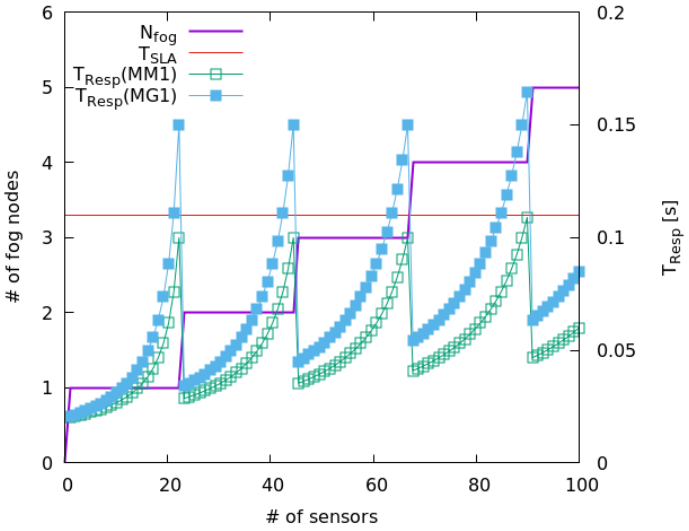
Another (unfortunate) case

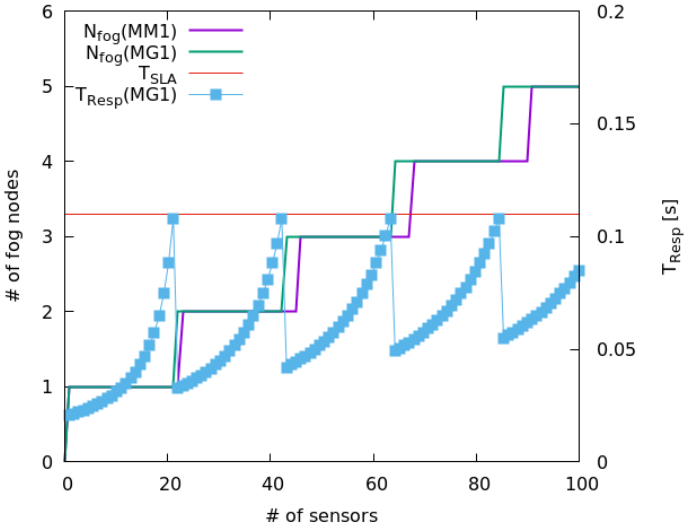
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Wrong model leads to wrong results
  - Expected (green) → OK
  - Real (blue) → SLA violations
- Should use correct model
  - Based on Pollaczek-Khinchin formula

$$N \geq \left\lceil \frac{\Lambda}{\overline{\mu}} \cdot \frac{\mathrm{CoV}^2 - 2K - 1}{2K - 2} \right\rceil$$

  - Correct scaling

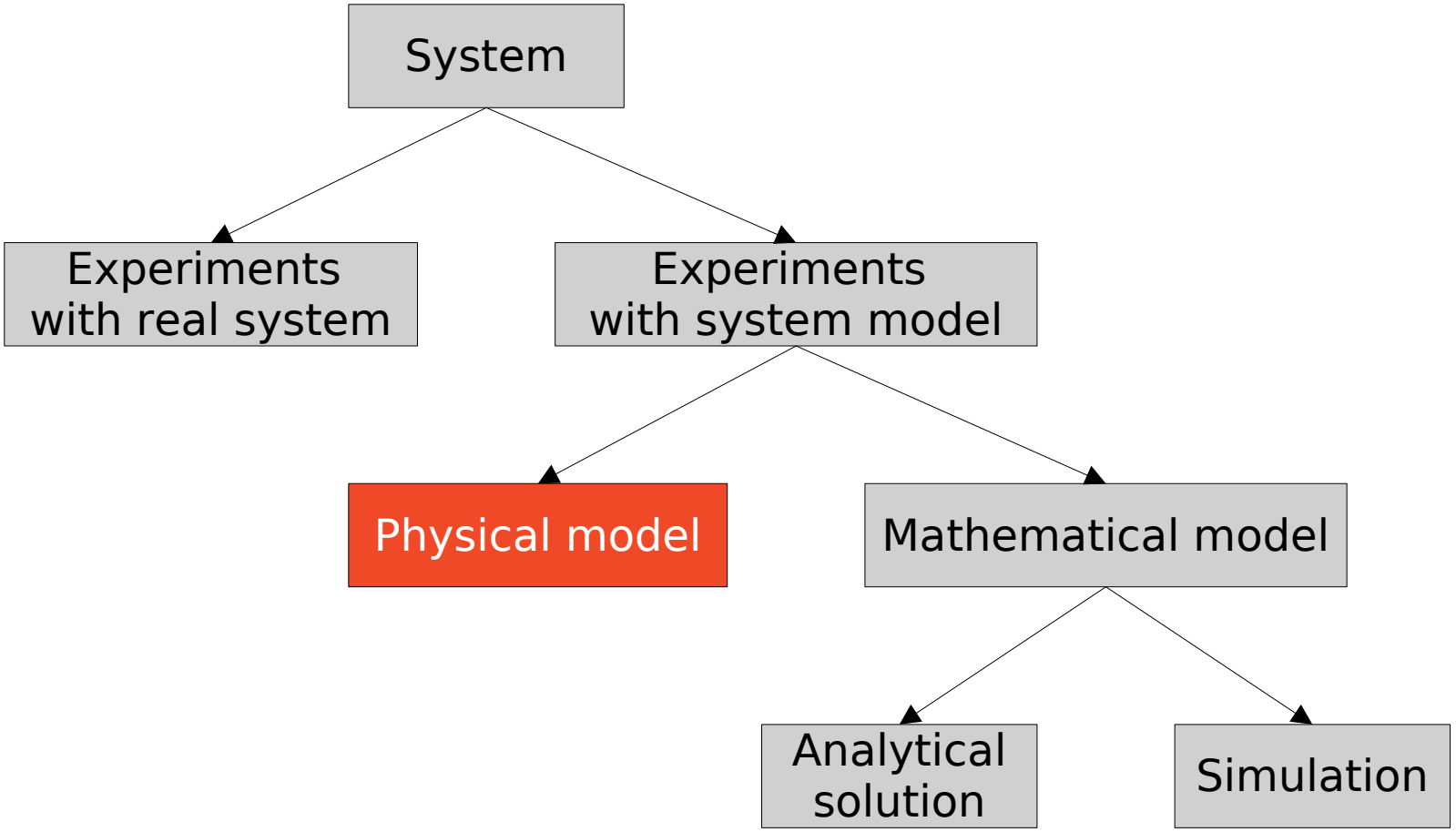




Real system testing

Possible approaches

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA



Types of test

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Functional testing
  - System behaves as expected
- Activity testing
  - System works under normal conditions
- Endurance testing
  - System can scale under (limited) pressure
- Stress testing
  - Limits of the system
- (Benchmarking)
  - Performance under standardized load

Functional testing

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Functional testing
- Definition of expected behavior
  - Test suite with known results
- Execution of tests
  - Checking expected results
  - Checking error catching
  - Checking for timeouts
  - Bugs/Regressions
- Example:
  - Unit testing

Activity testing

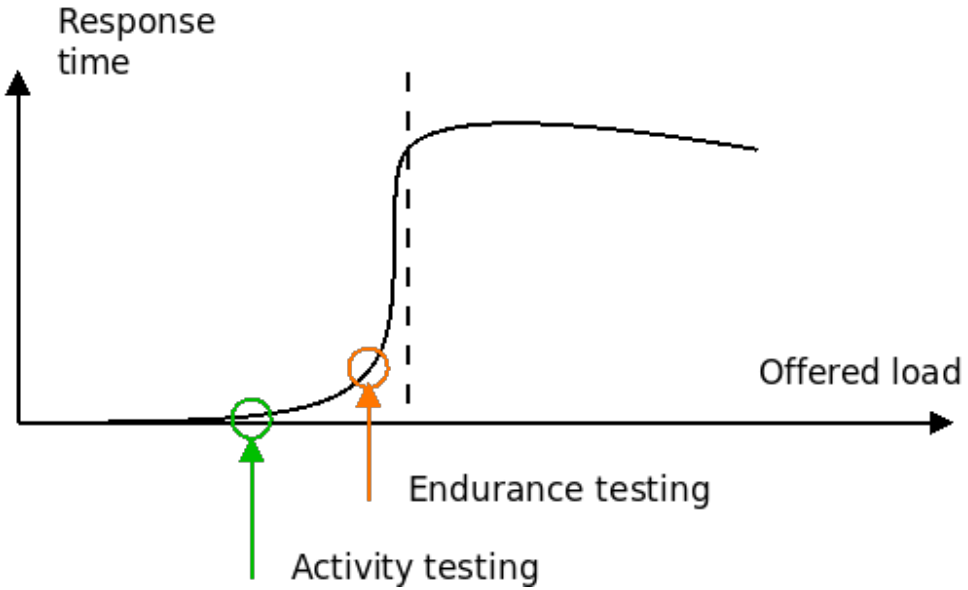
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Activity testing
- Typical workload
- Normal operation set
- Identification of problems
  - Beyond unit testing
- Example
  - Testing from users
  - Small production deployment

Endurance testing

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

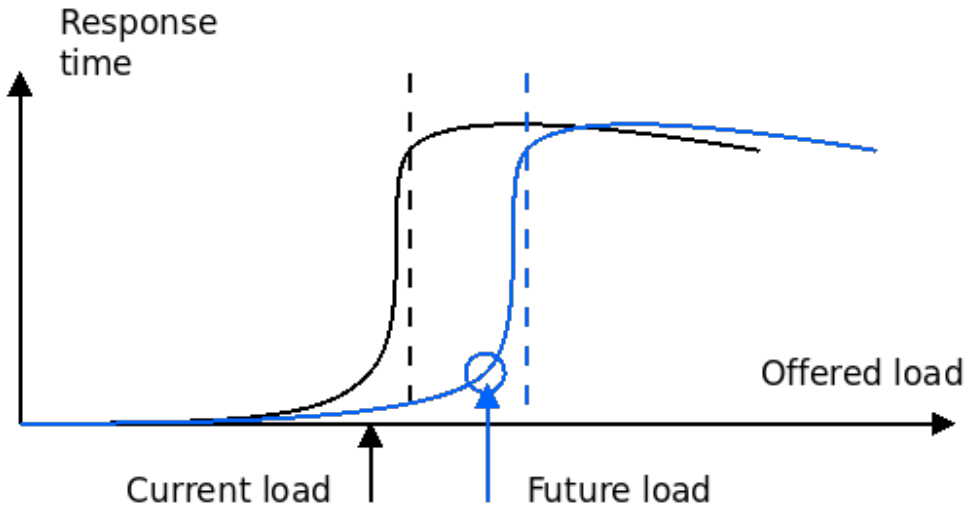
- Endurance testing
- High load → system under stress
- Main focus
  - Is the system still working on rush hours?
  - What-If scenarios
  - *what happens if load increases by 20%?*



Endurance testing

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

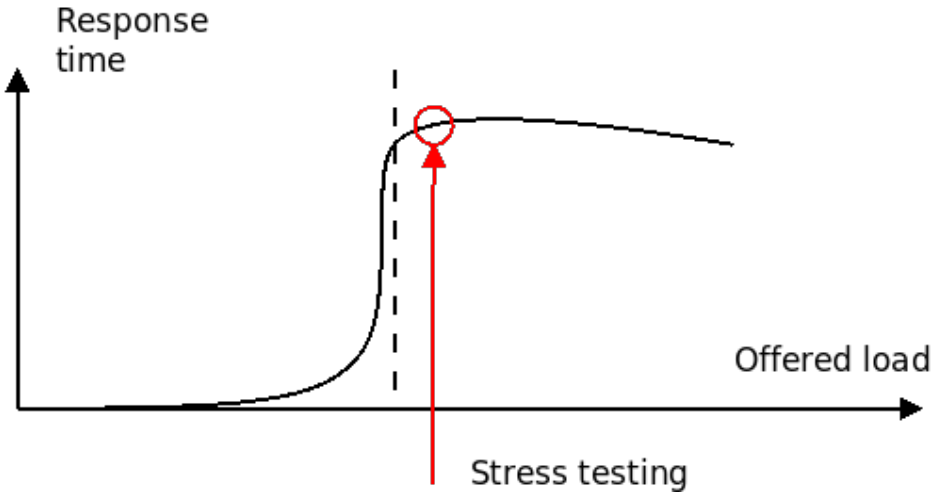
- Endurance testing
- Goals
  - Evaluate scalability
  - Input for capacity planning
- System is under stress
  - *But it is expected to cope with the new conditions*

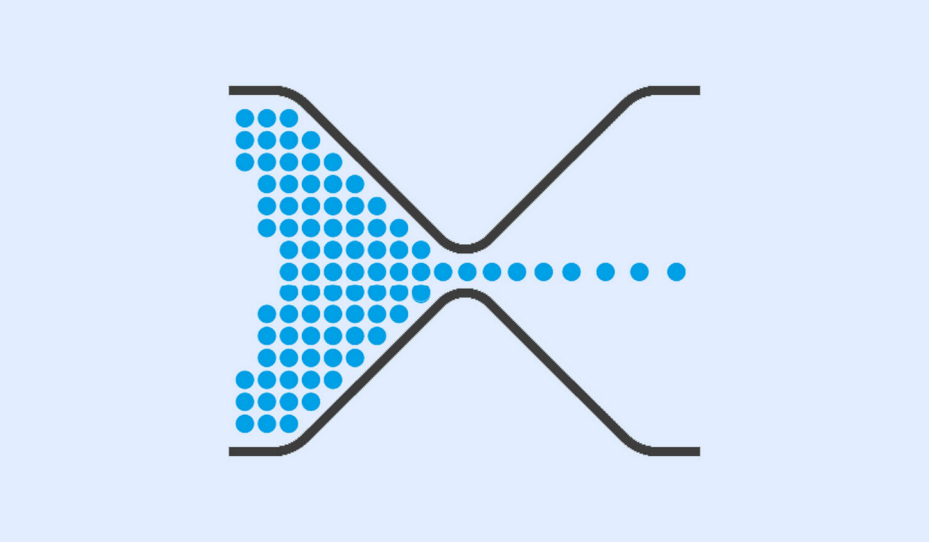


Stress testing

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Stress testing
- System is taken to the limit
  - We aim to create failures
- Main focus
  - What is the maximum load?
  - Where are the system bottlenecks?
  - Presence of trashing?

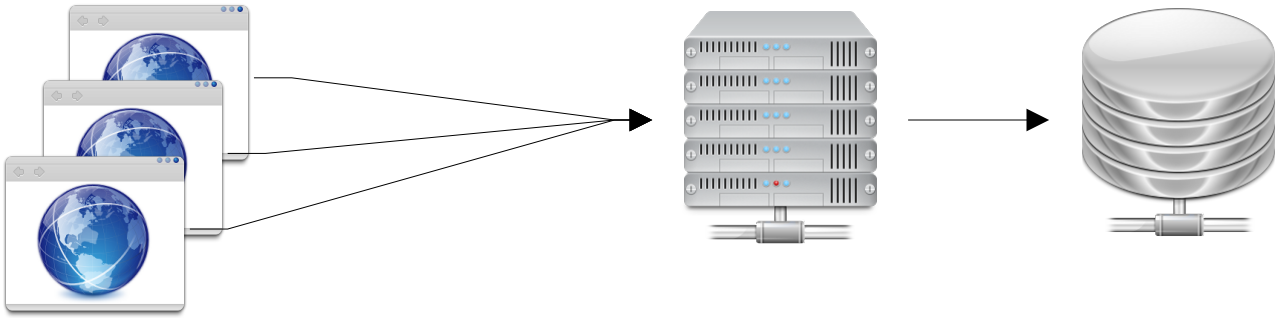




Example

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Example:
- 2-tier Web server
  - DBMS: up to 100 req/s
  - Web server: up to 500 req/s
  - 50% of Web page requests require access to DB
- What is the maximum throughput?
- Where is the bottleneck?



Example

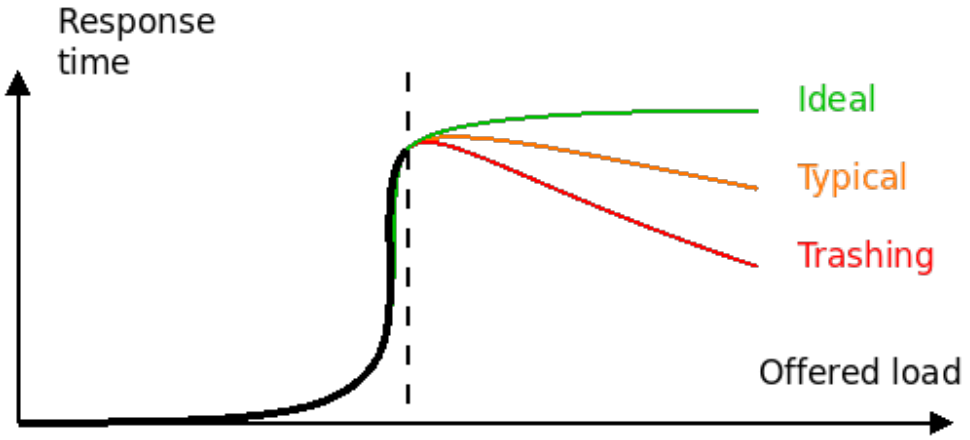
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Example:
- Incoming load of 500 req/s
  - Web server saturates
  - DBMS receives 500*0.5 = 250 req/s > 100 req/s
- Incoming load of 200 req/s
  - Web server utilization: 60%
  - DBMS is saturated (load = 100 req/s)
- Conclusion
  - Max load is 200 req/s
  - Bottleneck is DBMS

Trashing

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Stress testing
- Possible behavior
  - Ideal saturation
  - Trashing
  - Intermediate behavior
- Impact of thrashing
  - Cache locality disrupted
  - Resource contention



Benchmarking

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Performance evaluation requires a workload
- How can I compare my working condition with other scenarios?
- How can I select the best option of every experiment under different workload conditions?
- Need for a standard

Benchmarking

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Dictionary definition
  - *A point of reference by which something can be measured*
  - *(not necessary related to computers)*
- Computer industry definition
  - *A set of conditions against which a product or system is measured*
  - *A set of performance criteria which a product is expected to meet*
- Critical concepts:
  - Point of reference
  - Benchmark is related to measurement

Benchmarking

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Load testing
- Evaluating performance
- Realistic workload conditions
- Refers to specific application scenario
- Results cannot be compared outside the specific application

- Benchmarking
- Evaluating performance
- Using workload conditions
  - Well-defined
  - Standardized
- Results can be compared between different systems

Benchmarking

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Types of benchmark
- Toy benchmark
- Real programs

} Not very representative

- Synthetic benchmarks

} Slightly better, prone to abuse

- Benchmark based on real applications

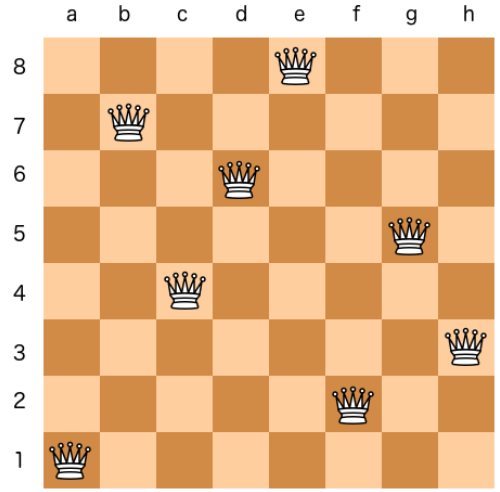
} Much better

Toy benchmark

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Toy benchmark
- Small algorithms
- Examples:
  - Quicksort
  - Matrix operations
  - 8 queens problem
- Simple to build
- Easy to run
- Not very representative
  - *Who runs 8-queen problem in daily work?*

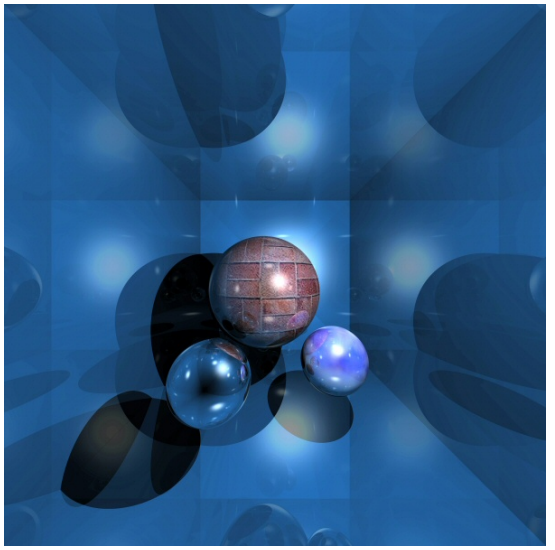
$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \cdots & u_{1,n} \\ & u_{2,2} & u_{2,3} & \cdots & u_{2,n} \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & u_{n-1,n} \\ 0 & & & & u_{n,n} \end{bmatrix}$$

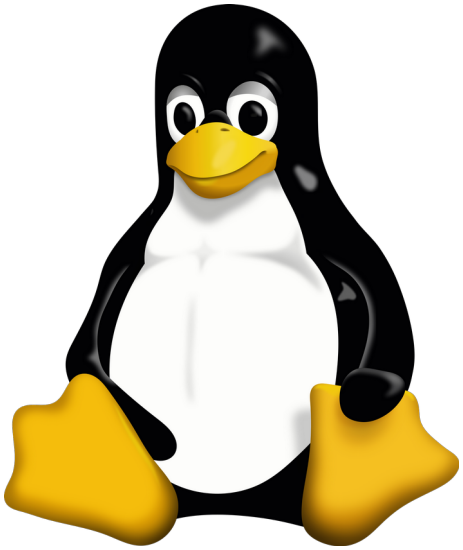


Real programs

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Real programs
- Run a well known software on a platform
- Time for the task is a measure of system performance
- Examples
  - Ray tracing SW
  - Compiling the linux kernel





Real programs

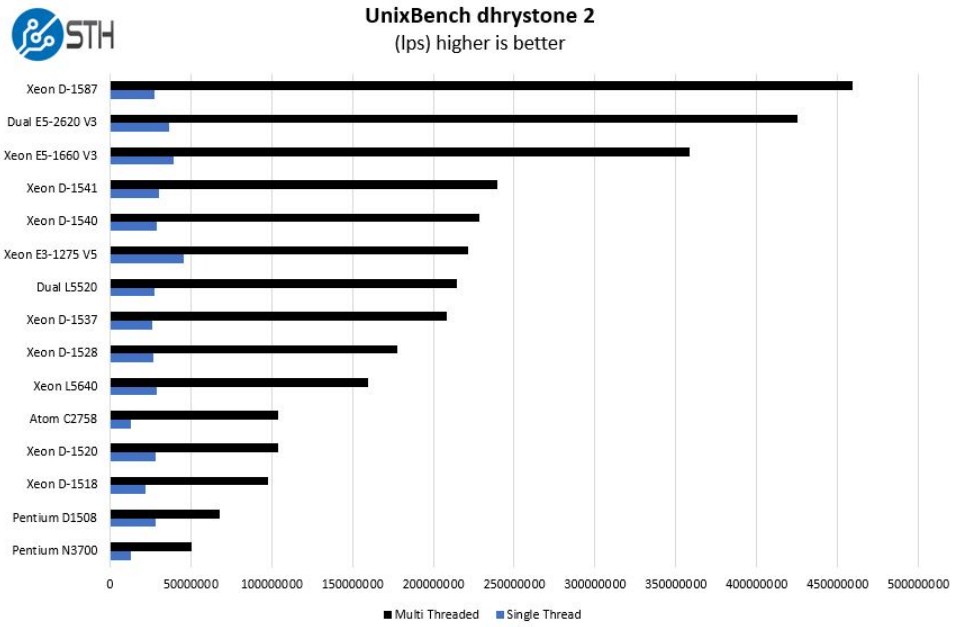
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Some common problems
- Not every software works on every platform
  - Porting issues of gcc toolchain
- Optimization problems
  - Driver issues for some GPU
- Not clear if the workload is
  - Significant
  - General to draw meaningful conclusions
- If multiple SW are used
  - How to weight them properly?

Synthetic benchmark

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

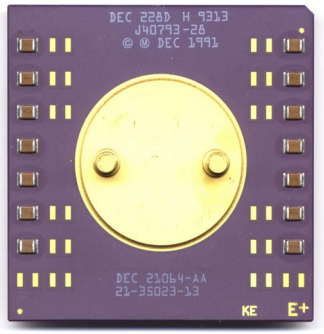
- Synthetic benchmarks
- Based on common operations in software
- Typically focused on some limited aspect of HW architecture
- Dhrystone
  - Integer math performance
- Whetstone
  - Floating point performance



Benchmark abuse

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Cases of benchmark abuse
  - Marketing impact of benchmark results
- Hardware architecture optimized for benchmark performance
  - Multiply-and-sum instruction (Power PC architecture)
- Compiler optimization
  - Unroll-loops in benchmark code
  - Jump optimization (Alpha CPU)
- Benchmark can be
  - Binary (pre-compiled)
  - Source (to compile)



Benchmark abuse

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Some common tricks
- Use home-brewed benchmark, give no reference
- Show only a subset of suites with better performance
- Modify benchmark source code
- Use only part of the benchmark tests
- Use timing of measures to manipulate overheads
  - E.g., discard first runs to mask cache misses
  - Measure only some overheads that make competitors look bad
- Everything should be clearly stated and motivated

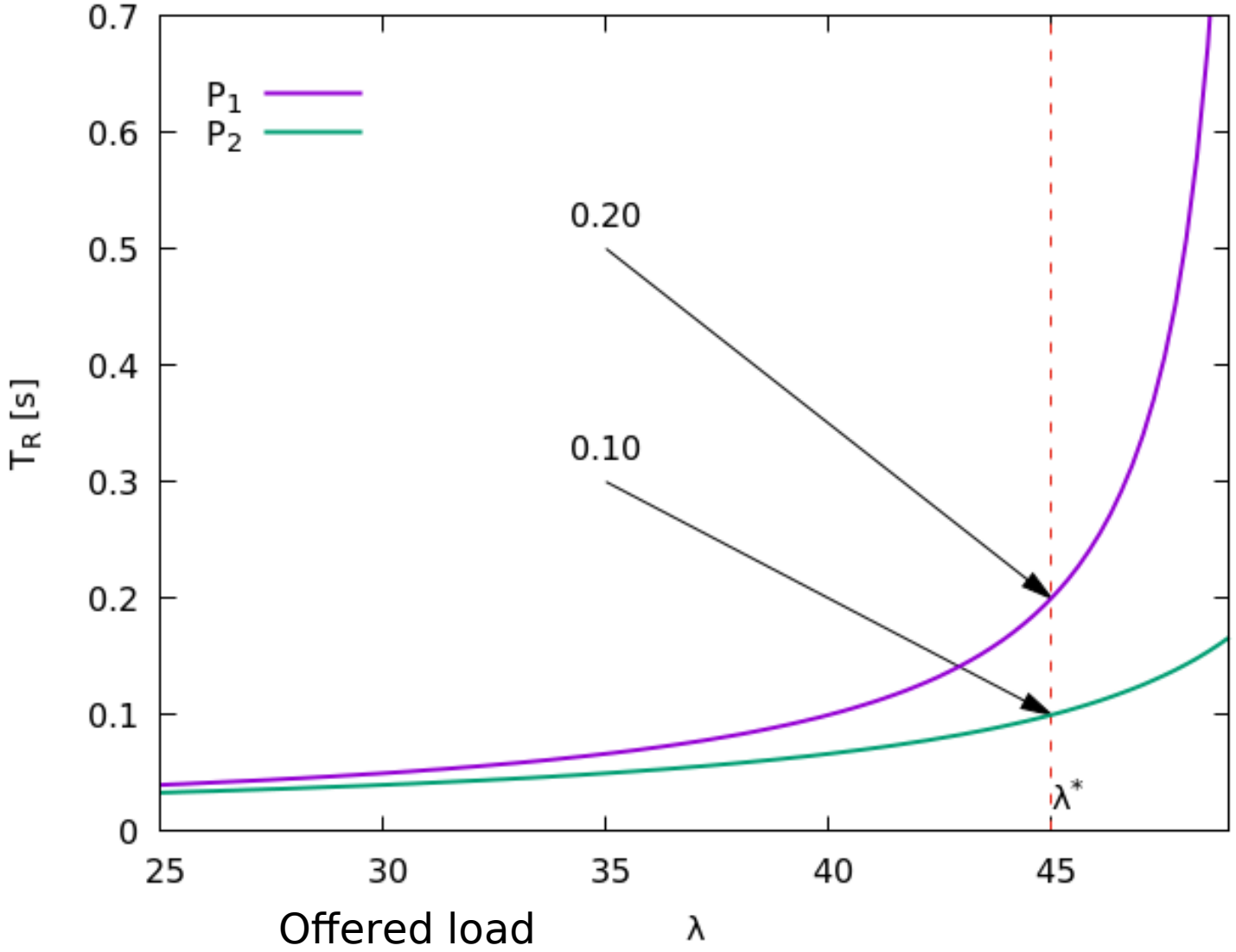
A dirty trick

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Let's turn Mr. Hyde for an example
- We want to sell a new CPU
  - Slightly faster than competitors
  - Expensive

- Need good advertisement
  - ~~Pay 50% more for a 10% performance increase~~
- We can *tune* our benchmark
  - When system is heavily loaded small changes in power make the difference in response time
  - Pay 50% more for 2× faster

A dirty trick



A dirty trick

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- We will see the model detail later
- Lessons to bring home
  - Understanding performance model is important
  - Knowing benchmark test conditions is vital
- Now back to Dr. Jekill mode

Real applications

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Benchmark based on real applications
- Goals:
  - Make benchmark representative of real problems
  - Provide open source code and clear performance measurements
- Compare with previous types of benchmark
  - Full application (not just algorithms)
  - Representative problems (not just kernel compiling)
  - More than ALU testing (not just Int or FP math)
  - Fully standardized scenarios

Standards

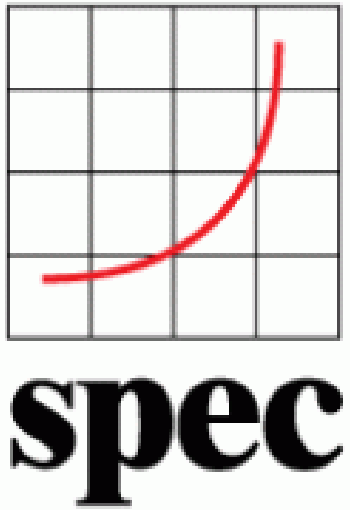
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Old-school benchmarks
  - SPEC
  - TPC
- More cloud-friendly benchmarks
  - SPEC
  - Yahoo YCSB
  - PerfKit Benchmarker

SPEC Web

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Developed by SPEC
  - System Performance Evaluation Consortium
  - Non-profit group developing several benchmark
- Representative workload
  - Web requests based on log traces
  - Logs from NCSA, HP, ...



SPEC Web

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Workload
  - 4 classes of files
  - Poisson file size distribution in classes
- Scaling
  - N. directory in data set ×2 as data set size ×4
  - Requests evenly distributed on dirs
- Emulated Users model
  - On-off model
  - Think time for pages

Fraction [%]	Size
35%	0 – 1 KB
50%	1 – 10 KB
14%	10 – 100 KB
1%	100 KB – 1 MB

- Support for levels of burstiness
- Architecture:
  - One controller
  - Multiple clients
  - One server

SPEC Web

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Evolution of benchmark
- Different directory size
- Support for HTTP/1.1
  - Keep-alive requests (70%)
  - Cookies
- Support fo dynamic content
- Support for advertisement based on user-id cookie
- Zipf file popularity model

Fraction [%]	Type
70.00%	Static GET
12.45%	Dynamic GET
12.60%	Dynamic GET+ Ad
4.80%	Dynamic POST
0.15%	Dynamic GET + CGI

TPC

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- TPC:
  - Transaction Processing Performance Council
- Several classes of benchmarks:
- TPC-W:
  - Web-based benchmark
- TPC-D
  - Database benchmark
- Each benchmark is based on a test application

TPC

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

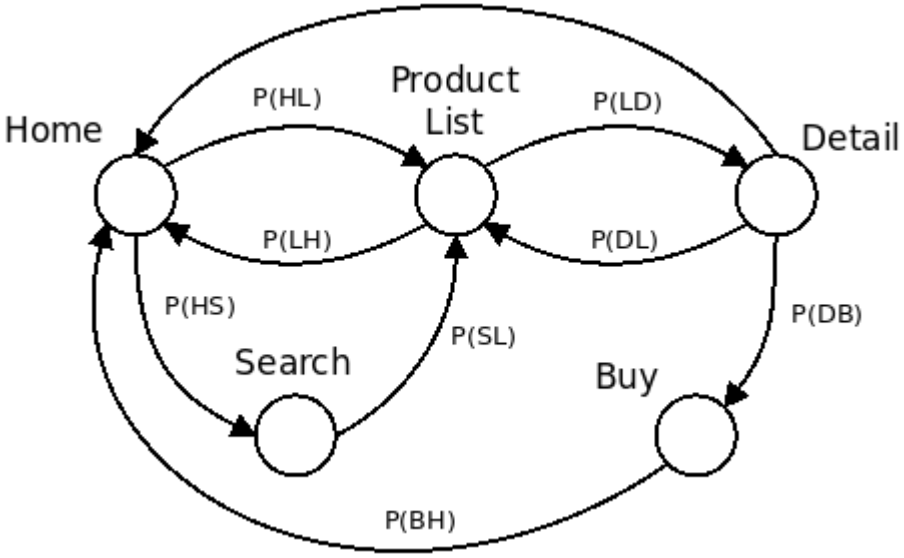
- TPC-W
- Dynamic Web site (changes with version)
  - Book store (like Amazon)
  - Auction system (like ebay)

- Auction system inspired by RUBiS benchmark
  - Rice University Bidding System
  - Auction in Java and PHP

TPC-W

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Definition of workload
  - Several dynamic web pages
  - Database size (product catalog)
- Definition of user behavior
  - User defined as stochastic FSM
  - Each page is a state
  - Probability of passing from a page to another
- Population of users
- Multiple workload mixes (browsing/shopping/...)



SPEC-Cloud

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- SPEC-Cloud
- IaaS-oriented benchmark
- Focus on
  - Scalability
  - Elasticity
- Test:
  - Scalability → load increases without creating errors
  - Elasticity → load changes, SLA preserved

YCSB

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- YCSB
  - Yahoo Cloud Serving Benchmark
- Developed by Yahoo!
- Focus on Cassandra Database
- Available on github
- Test of transactions on a DB
  - Replicated
  - NoSQL

tjake/**YCSB-test**

Yahoo! Cloud Serving Benchmark

0 Contributors | 0 Issues | 2 Stars | 1 Fork

YCSB

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Access to Key-Value storage
- Multi-phase benchmark
- Baseline phase
  - Run once the application
- Elasticity+Scalability phase
  - Re-run application with varying workload intensity
  - Collect measures
  - Compare with baseline

YCSB

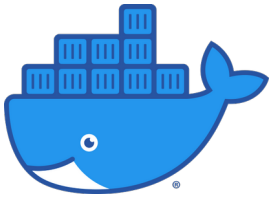
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

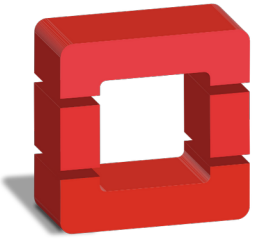
- Scalability metric
  - Throughput speedup compared to baseline
- Elasticity metric
  - Performance deviation from baseline
- Other metrics
  - Instance provision time (time to deploy new VM)
  - Instance run success (percentage of successful runs)
  - Provisioning success (percentage of new instance successfully deployed)

PerfKit Benchmarker

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Google tool
- Comparison of cloud solutions
  - Focus on network performance
- PKB features
  - Automatic setup and tear-down
  - Support for Docker and Open Stack
  - Most common workloads supported

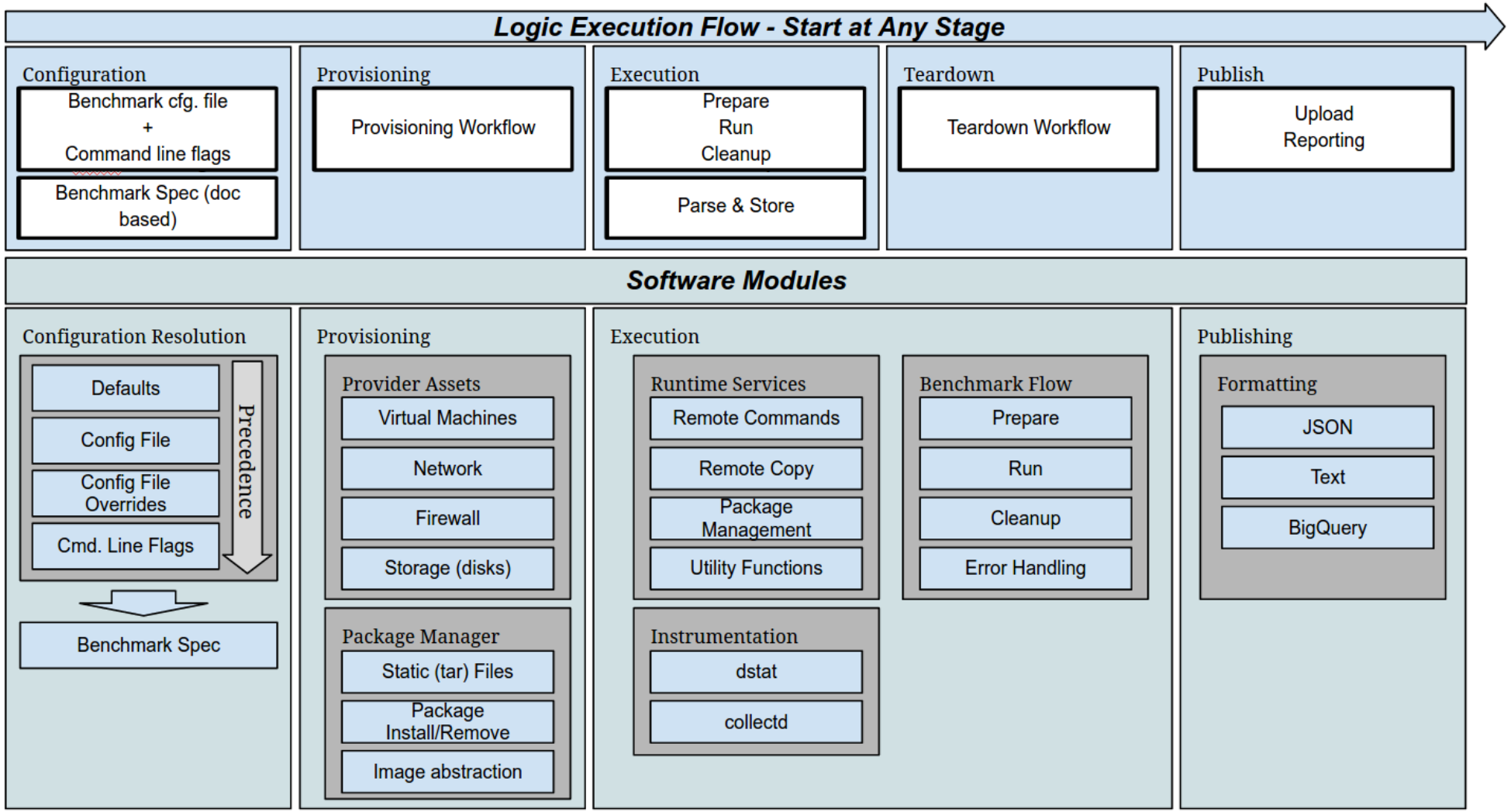




openstack
CLOUD SOFTWARE

PerfKit Benchmarker

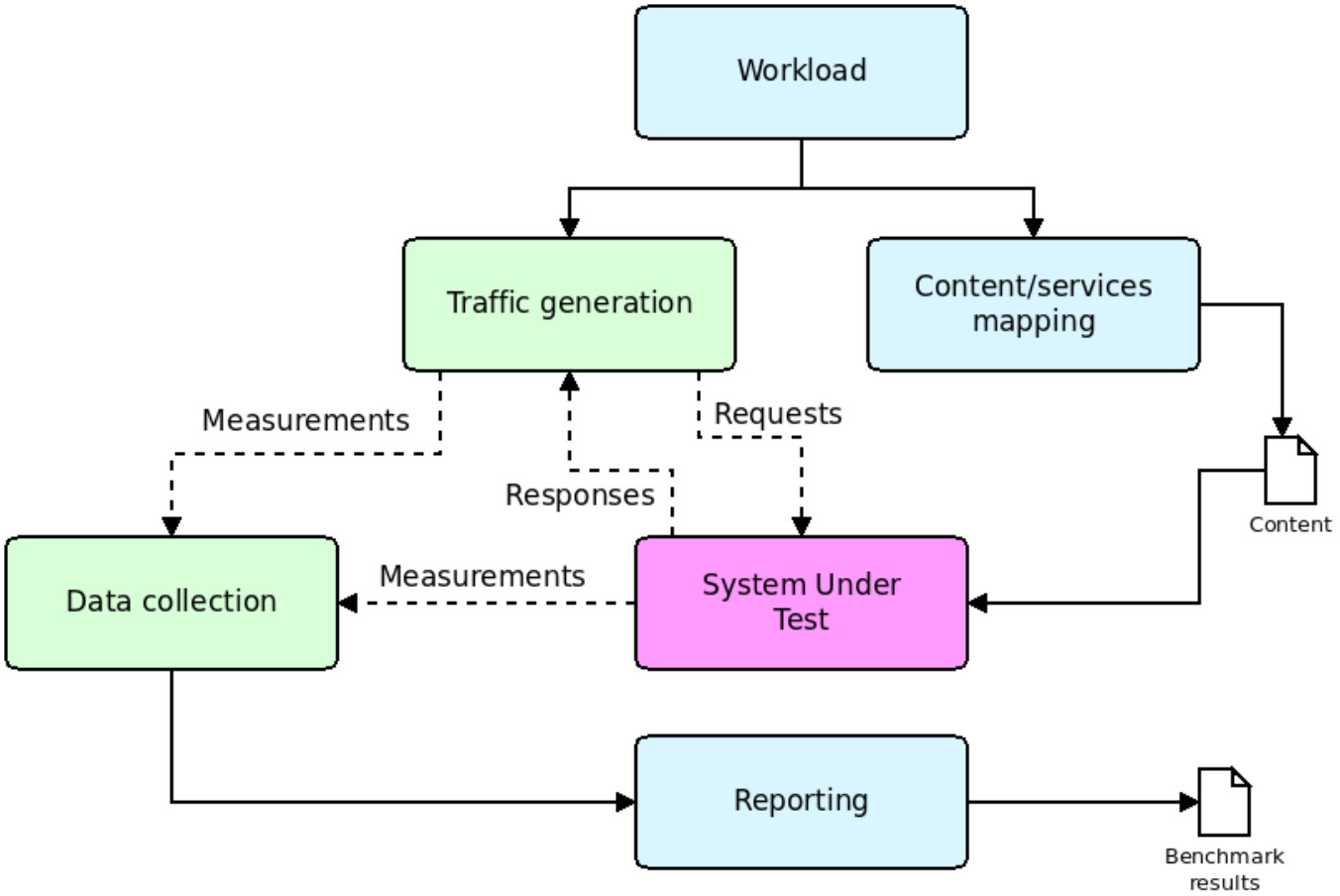
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA



Benchmark architecture

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

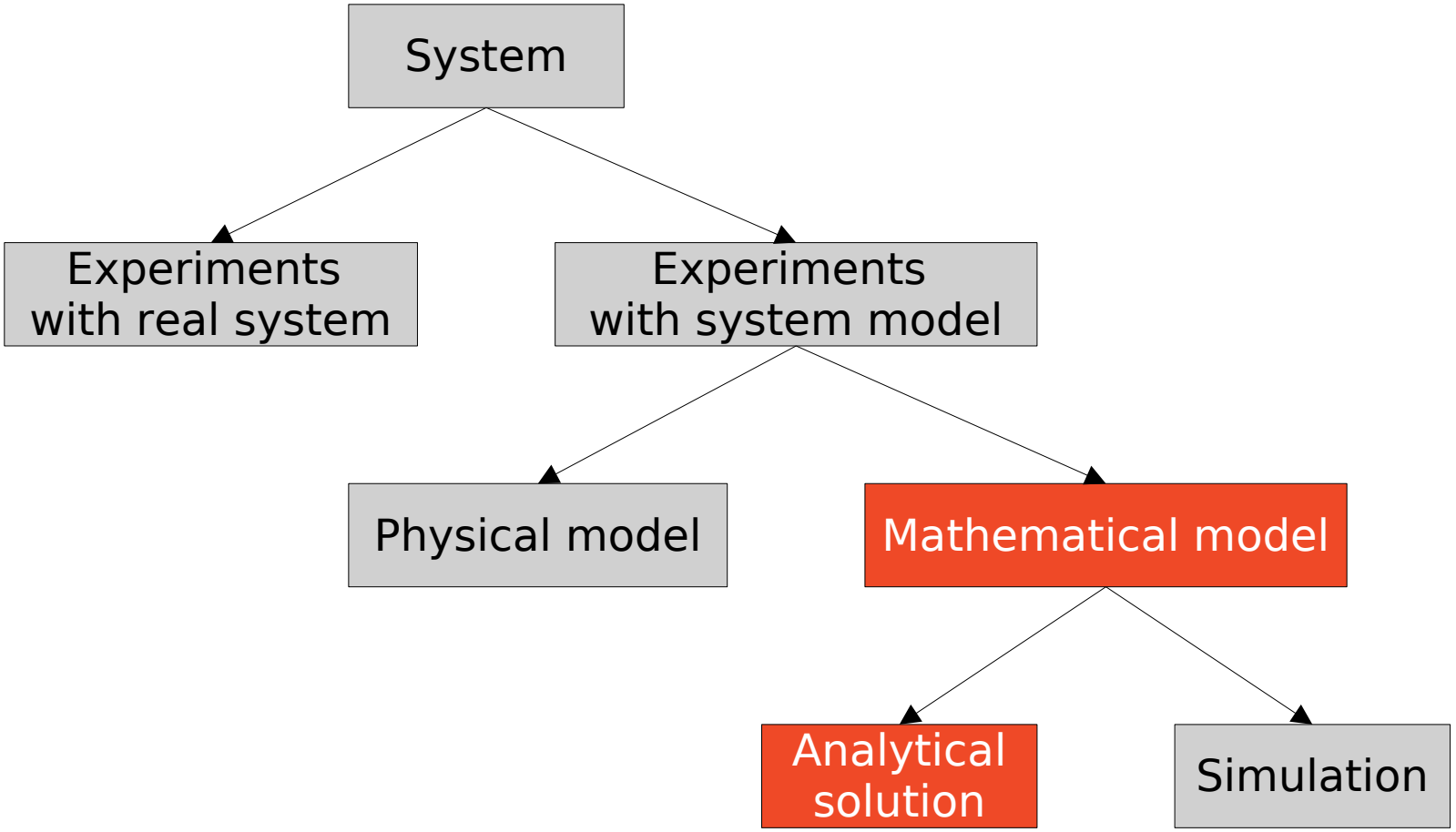
- Most examples seen share a common architecture
- Declined based on the goals of the performance study



Mathematical models

Possible approaches

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA



A simple example

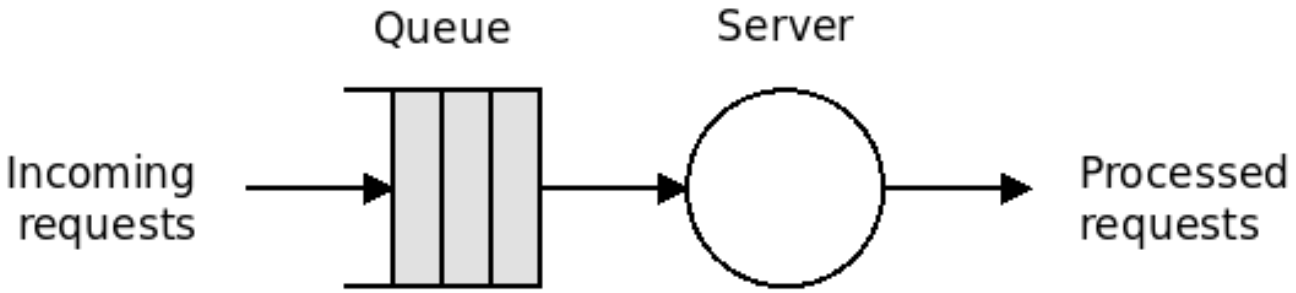
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Multi-tier Web server
- Main contributions of response time:
  - Network
  - Front-end node
  - Middle-tier node
  - Back-end node
- Each node response time:
  - CPU time
  - Storage time (data retrieval)
  - Communication (LAN)

Network model

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

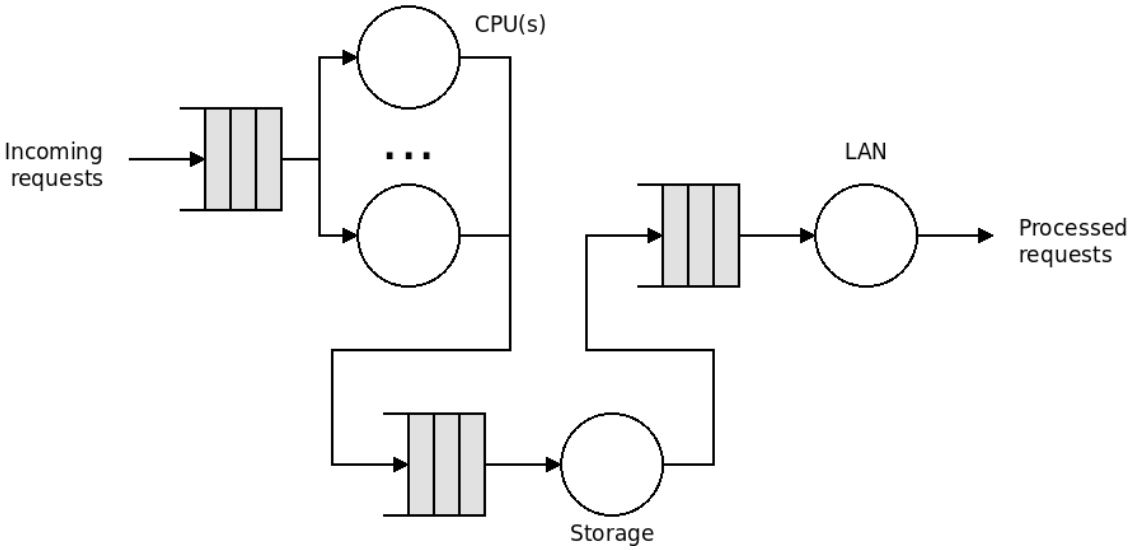
- Queuing model
- Example: network
  - Input: Incoming packets to be delivered
  - Output: delivered packets
  - Network is busy during transmission
  - Packets wait in routers queues



Nodes model

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

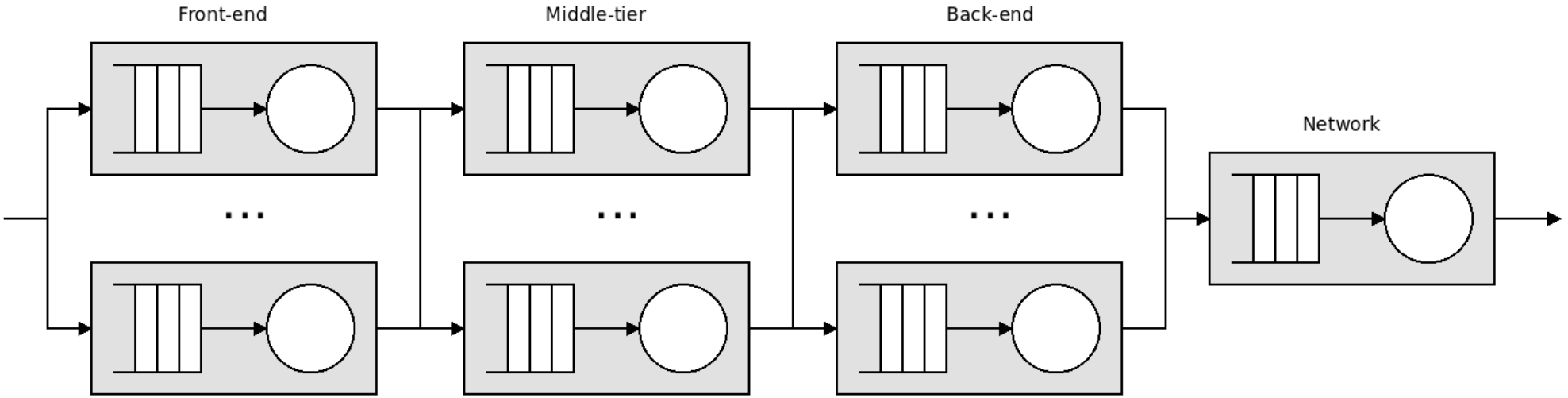
- Node model
  - CPU (multiple parallel servers for multi-core arch.)
  - LAN
  - Storage
- Each resource as a queue model



Big picture

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Complex model
- Several queues
- Same questions
  - Where is the bottleneck?
  - How to address it?



Modeling the problem

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Focus on a single node
  - Queues: CPU, storage, LAN
  - 1 server per each resource
- Workload
  - S_i: average service time at i-th resource ($1/\mu$)
  - W_i: average waiting time at i-th queue
  - λ_i: arrival rate at i-th queue

Modeling the problem

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Performance metrics
  - R_i: average response time at i-th queue
  - X_i: throughput of i-th queue
  - X_0: overall system throughput
- Number of jobs at a resource
  - N_{iw}: requests in waiting queue at i-th resource
  - N_{is}: number of requests being serviced [0,1] (or [0,m])
  - N_i: number of requests at i-th resource

Utilization law

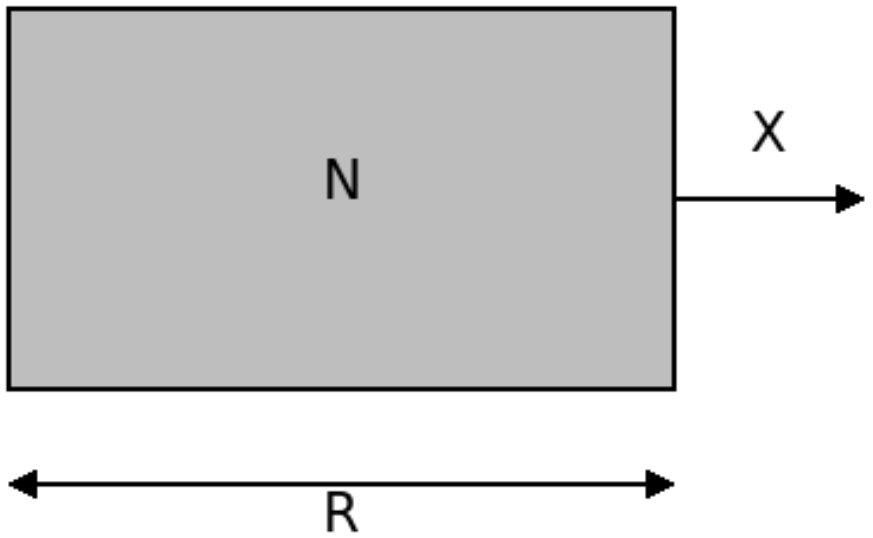
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- B_i the time the serve is busy in observation period τ
- C_i completed requests in period τ
- Utilization $U_i = B_i/\tau$
  - But $X_i = C_i/\tau$
  - Hence $1/\tau = X_i/C_i$
  - $U_i = B_i * X_i/Ci = (B_i/C_i) * X_i = S_i * X_i$
- In a steady state: $X_i = \lambda_i * S_i$
  - Often used notation: $\rho=\lambda/\mu$
- In a multiple server queue (with m servers)
  - $U_i = (X_i/m) * S_i$

Little's law

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- System as a black box
  - N requests inside
  - X rate of outgoing requests (throughput)
  - R time spent within the system



Little's Law

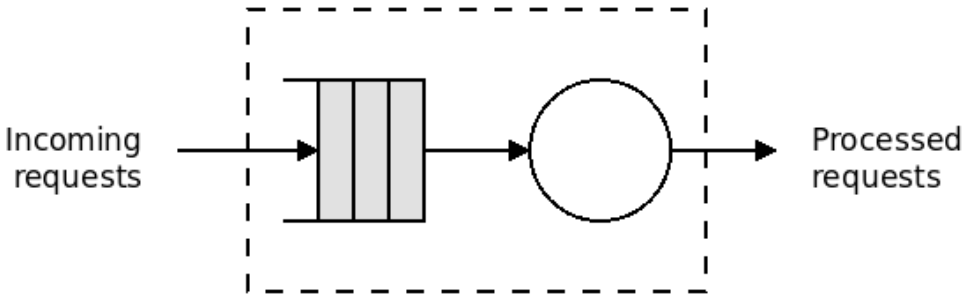
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

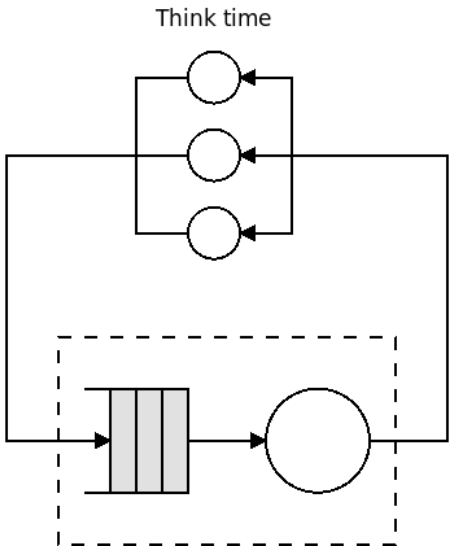
- General formulation:
  - $N=X*R$
- Applied to our problem:
  - Waiting: $N_{iw} = X_i * W_i$
  - Service: $N_{is} = X_i * S_i$
  - Response: $N_i = X_i * R_i$

Open/Closed loop models

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Open loop:
  - Potentially infinite population of users
  - Requests come in and go out form system
- Closed loop:
  - Finite population of users
  - Requests circle through the system
  - Think time of users

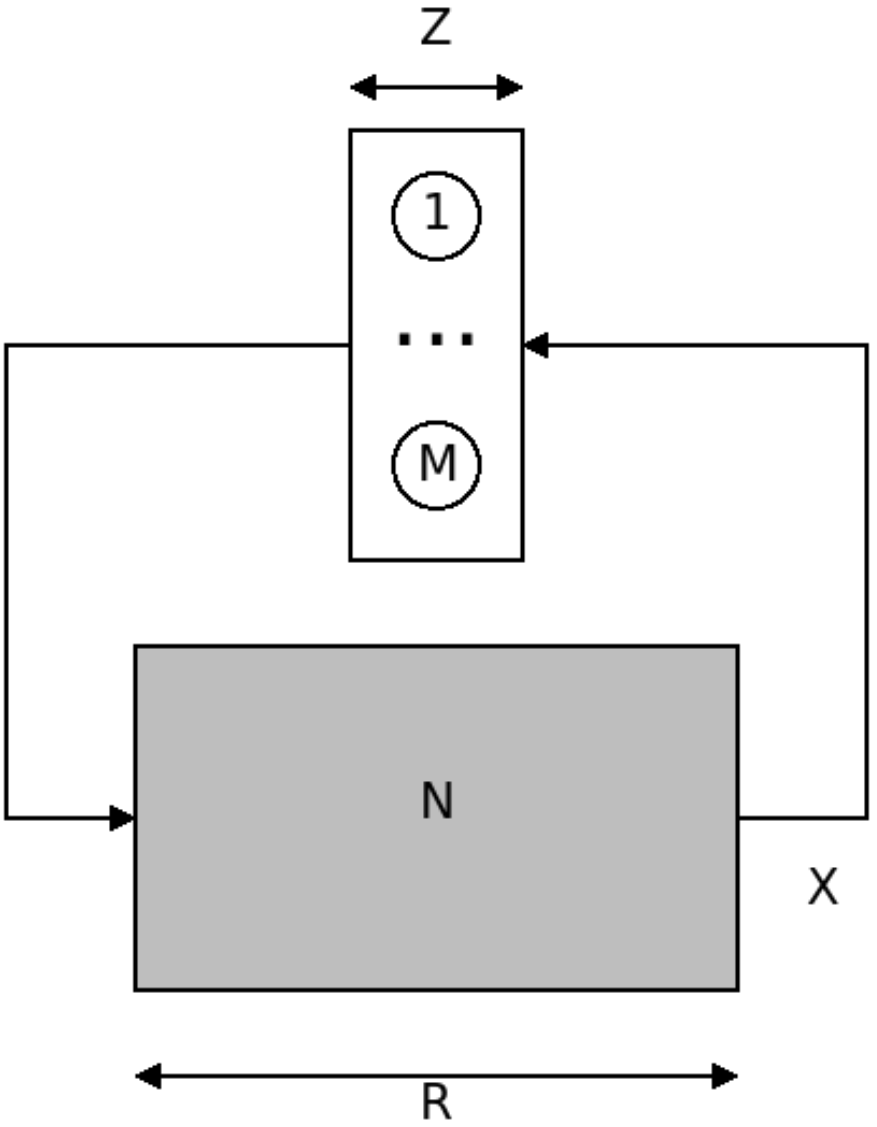




Response time law

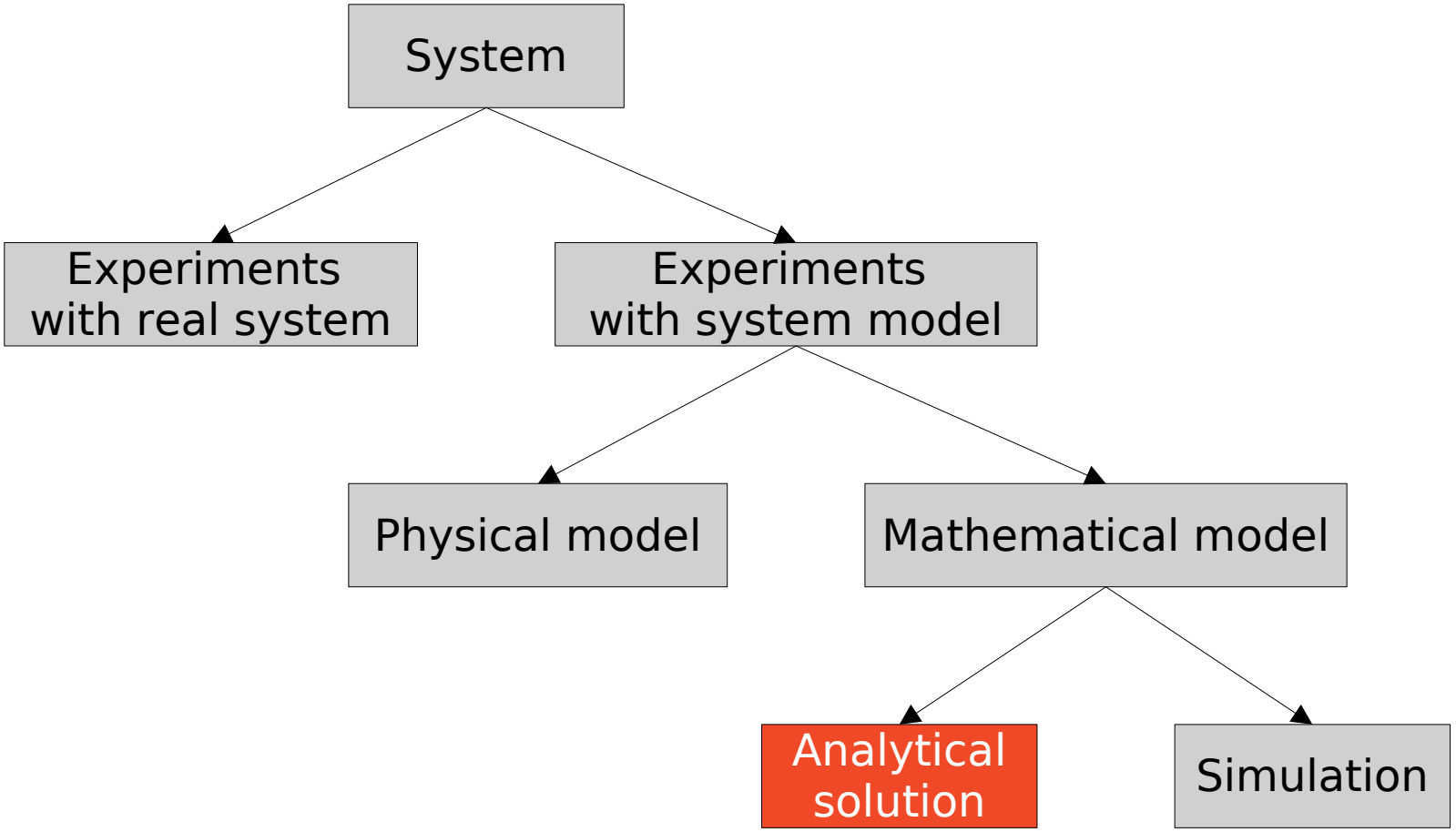
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Overview:
  - System with a population of M users
  - Think time Z
- Closed loop formulation of little's law
  - R+Z = M/X



Queuing theory

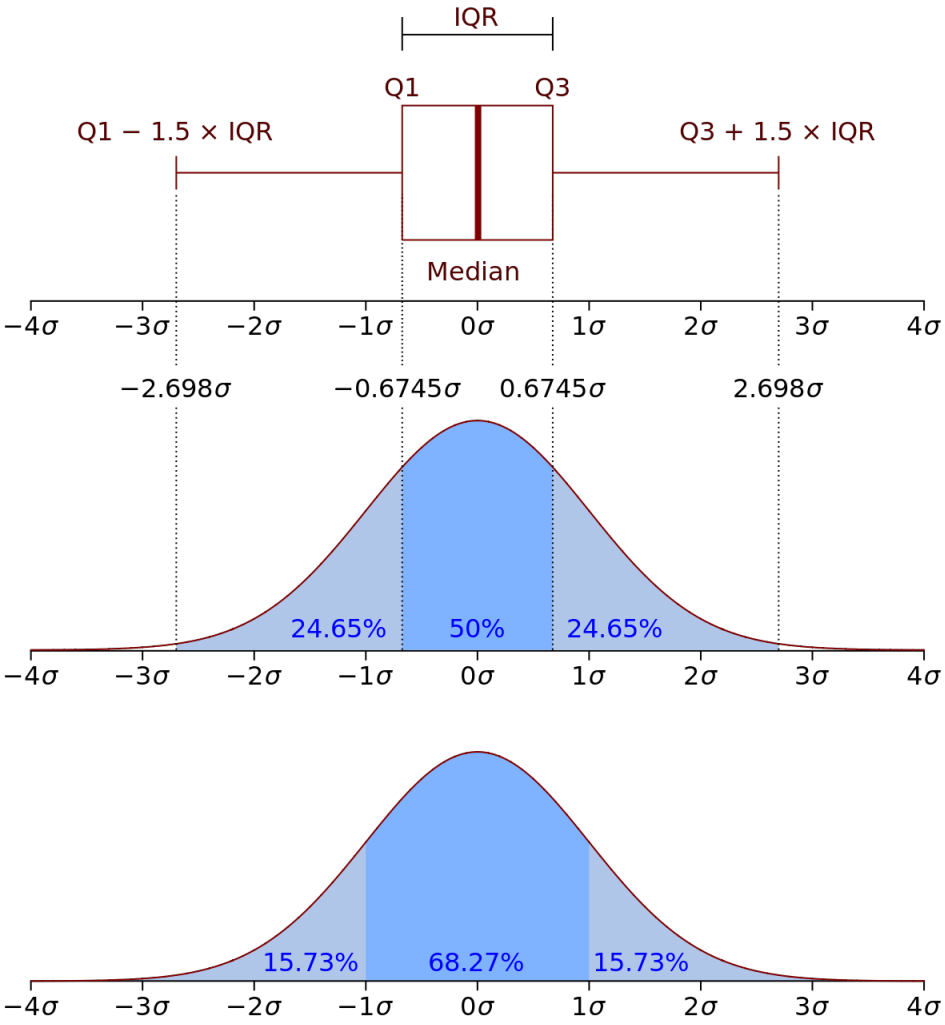
Possible approaches



Stochastic process

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

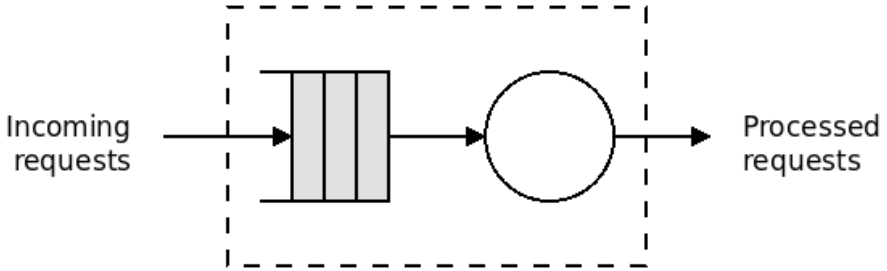
- Stochastic process
  - A process driven by random behavior
  - Defines a family f random (aleatory) variables
- Aleatory variable
  - Can be described by its statistical properties
  - e.g., PDF



System overview

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Open loop
- Incoming job
  - Stochastic process
  - Inter-arrival time is a random variable
- Processed requests
  - Stochastic process
  - Processing time is another random variable



Some definitions...

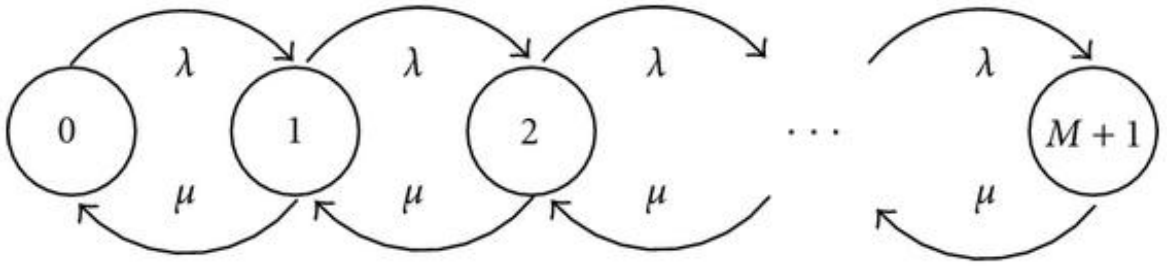
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- System definition
  - {Arrival}/{Process}/{# Servers}/{Queue Len}…
  - e.g.: M/M/1/∞ (M/M/1 for short)
- Some stochastic processes:
  - M → Poisson (memory-less)
  - D → Deterministic
  - E_K → Erlang distribution
  - G → Generic
- Poisson process are easy to model
  - Use of Markov Chain (the M in the process)

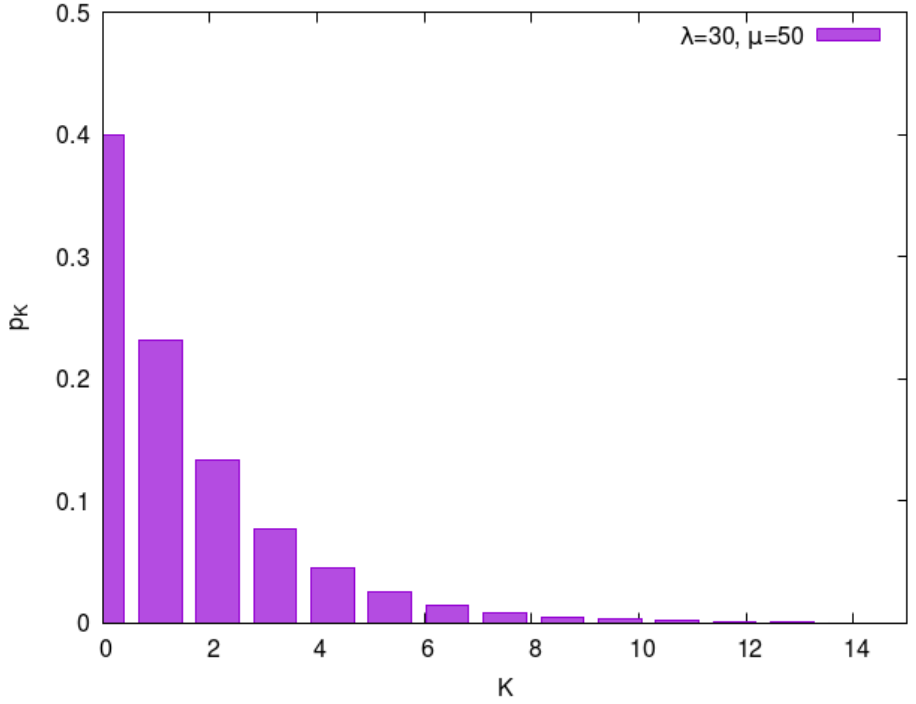
Poisson process

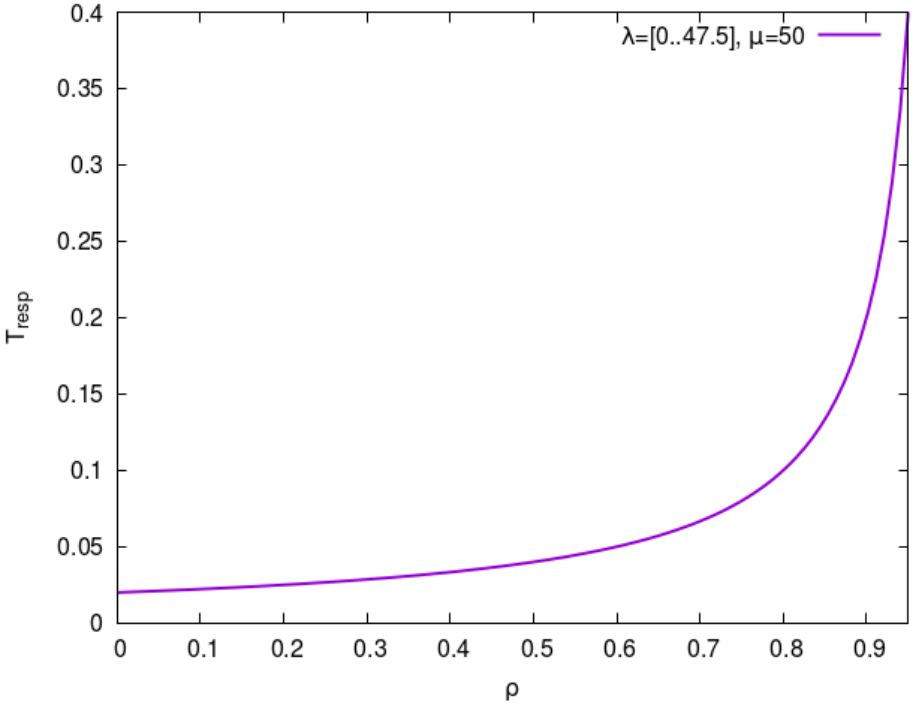
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Status:
  - Elements in system
- Poisson process:
  - Rates do not depend on status
  - Arrival rate: λ
  - Leave rate: μ
  - Utilization: $\rho = \lambda / \mu$
  - Probability to be in state k: $p_k = (1 - \lambda/\mu)(\lambda/\mu)^k = (1-\rho)\rho^K$



Poisson process





- Overview of Poisson process:
  - Infinite queue size
  - Probability of being state K
  - Response time as function of arrival rate: $T_{resp} = 1/(\mu - \lambda)$

Other useful formulas

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- PASTA Theorem (M/G/1 systems)
- Pollaczek-Khinchin Formula

$$\begin{cases} \mathrm{E}[W] = \dfrac{\lambda \mathrm{E}[S^2]}{2(1-\rho)} = \dfrac{1+C_v^2}{2} \cdot \dfrac{\rho}{1-\rho} \cdot \mathrm{E}[S] \\ \mathrm{E}[T] = \mathrm{E}[S] + \dfrac{\lambda \mathrm{E}[S^2]}{2(1-\rho)} = \left(1 + \dfrac{1+C_v^2}{2} \cdot \dfrac{\rho}{1-\rho}\right) \cdot \mathrm{E}[S] \end{cases}$$

- E[W]: average waiting time
- E[T]: average response time
- E[S]: average service time (1/μ)

Other useful formulas

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Allen-Cuneen approximation (G/G/N systems)

$$W_M = \frac{P_{cb,N}}{\mu N(1-\rho)}\left(\frac{C_S^2 + C_D^2}{2}\right), \qquad P_{cb,N} \approx \begin{cases} (\rho^N + \rho)/2 & \text{if } \rho \geq 0.7 \\ \rho^{\frac{N+1}{2}} & \text{otherwise.} \end{cases}$$

- W_M Average waiting time
- $P_{cb,N}$ Probability that all servers are busy
- C_S, C_D, Coefficient of variation
  - arrival (C_S)
  - Service (C_D)
  - Coefficient of variation = Std. Dev / Average

A dirty trick (reprise)

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Let's turn Mr. Hyde for an example
- We want to sell a new CPU
  - Slightly faster than competitors
  - Expensive

- Need good advertisement
  - ~~Pay 50% more for a 10% performance increase~~
- We can *tune* our benchmark
  - When system is heavily loaded small changes in power make the difference in response time
  - Pay 50% more for 2× faster

A dirty trick (reprise)

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- We need to find the right load λ^*
  - Goal: 2× gain on response time
- Real advantage
  - Throughout speedup is K
  - Competitor processor: 50 req/s
  - Our processor: 55 req/s → K=1.1
- Assume a simple setup:
  - Poisson request arrival
  - Poisson service

Challenge

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- How to model the problem?
- How to find λ^*?
- Volunteers?

A dirty trick (reprise)

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

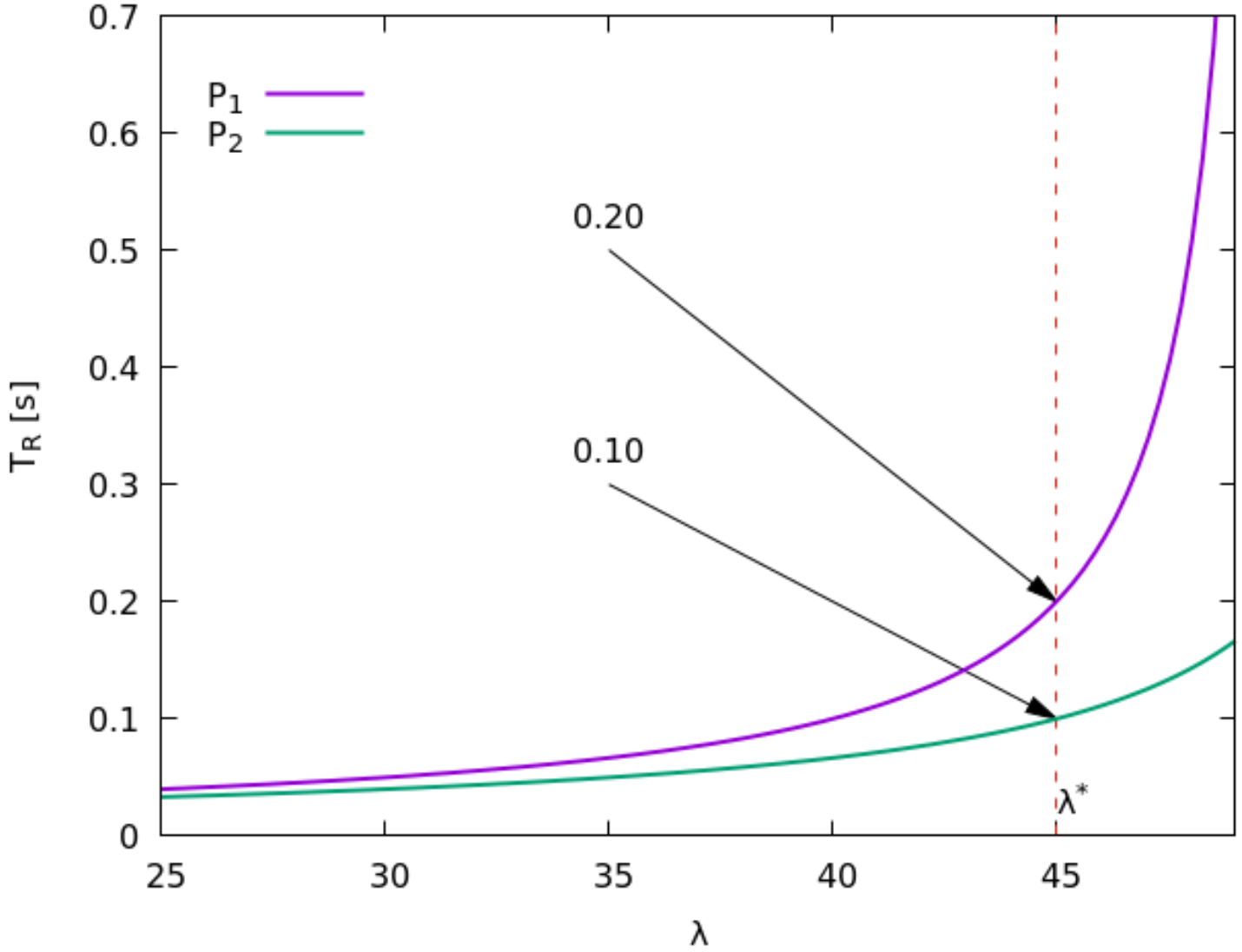
- We model the systems as M/M/1 systems
- Processor P1 (slow):
  - $\mu_1 = \mu$
  - $T_1(\lambda) = 1/(\mu-\lambda)$
- Processor P2 (our, fast):
  - $\mu_2 = K\,\mu$
  - $T_2(\lambda) = 1/(K\mu-\lambda)$
- P2 is 2x faster at λ^*
  - $T_1(\lambda^*) = 2\,T_2(\lambda^*)$

A dirty trick (reprise)

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Solution is easy:
  - $\lambda^* = (2-K)\mu$
- We can now plot the results

A dirty trick



Other exercises

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

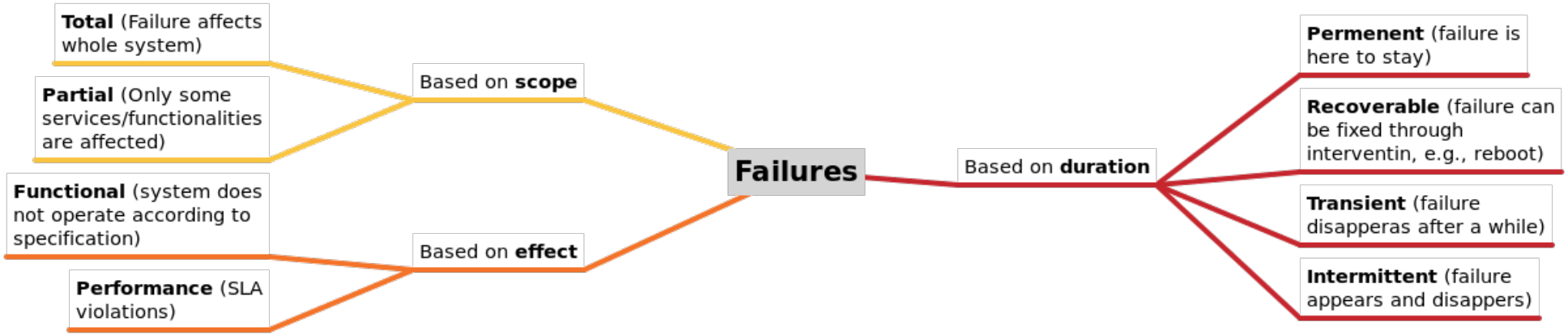
- We have an incoming load $\lambda > \mu$
- We want to scale-up our system → K servers
- Assumption:
  - Requests can be distributed uniformly over servers
- How many servers do I need to avoid overload?
- Assumption
  - I have a maximum response time T_{SLA}
- How many servers do I need to meet the SLA?
- Volunteers welcome

Failures

Availability

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Being available → not being in a failure state
- Classification of failures
  - Several criteria



Failures

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Classification based on Duration
- Permanent
  - Failure is here to stay
- Recoverable
  - It is possible to return online after intervention (e.g., restart, reboot)
- Transient
  - Failure disappears after a while
- Intermittent
  - Failure that appears and disappears

Failures

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Classification based on Effect
- Functional
  - The system does not operate according to its functional specifications
  - “System is not working right”
- Performance
  - System works but not in a timely fashion
  - SLA violations

Failures

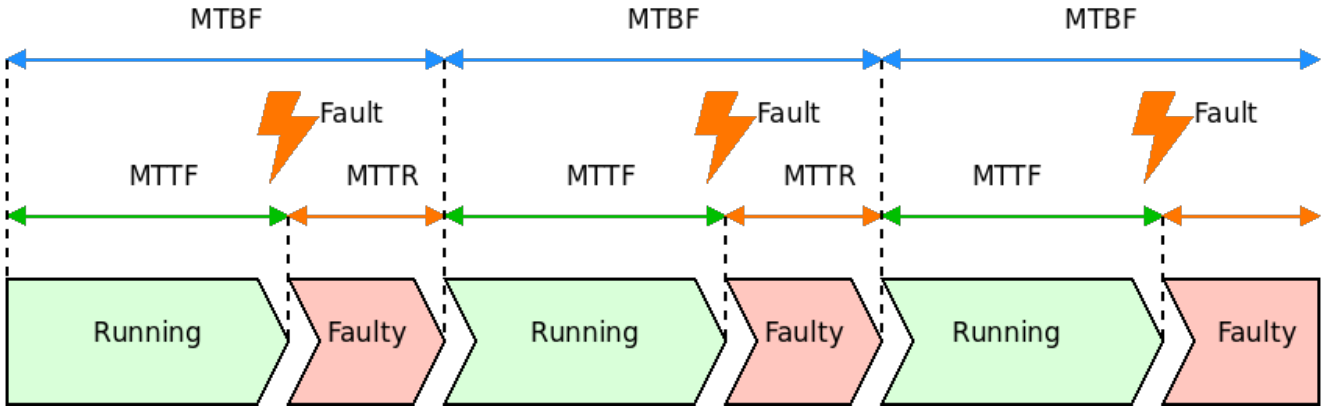
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Classification based on Scope
- Total
  - Failure affects the whole system
- Partial
  - Only some services/functionalities are affected

Failure metrics

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- MTBF
  - Mean Time Between Faults
- MTTF
  - Mean Time To Fault
- MTTR
  - Mean Time To Repair
- MTBF=MTTF+MTTR



Availability classes

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Availability Class	Availability	Unavailable [min/y]	System Type
1	90.0%	52560 (36d)	Unmanaged
2	99.0%	5256 (3.6d)	Managed
3	99.9%	526 (8.8h)	Well-managed
4	99.99%	52.6	Fault-tolerant
5	99.999%	5.3	Highly Available
6	99.9999%	0.53	Very Highly Available
7	99.99999%	0.053	Ultra Available

- Availability
  - Fraction of time when the system is operational
  - Availability A=MTTF/MTBF

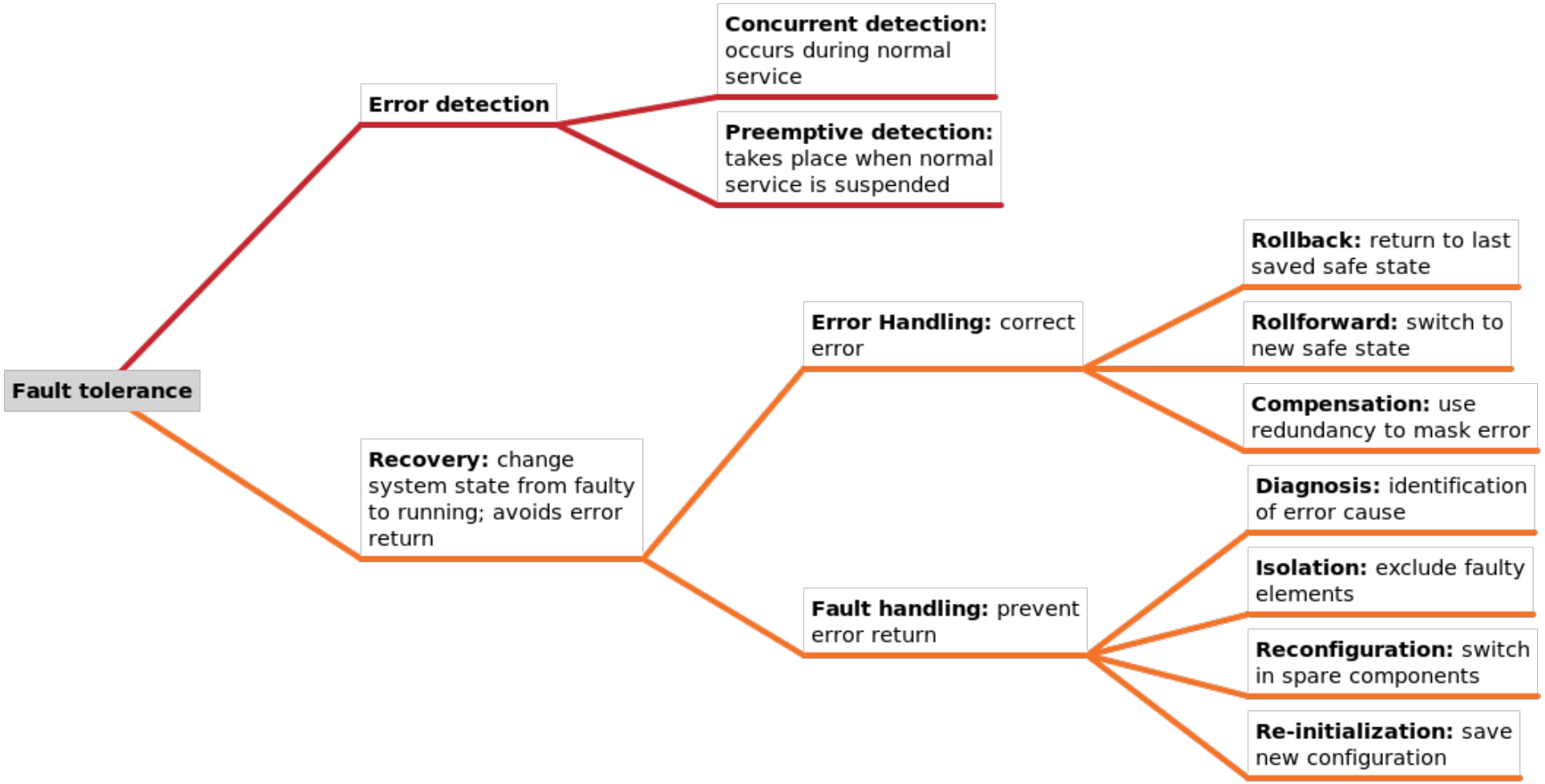
Availability and reliability

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Availability
  - Probability that a system is available at time t
- Reliability
  - Probability that a system works correctly during a period τ
- If the observation period τ is long enough Availability and Reliability converge to the same value

Fault-tolerance checklist

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA



Some figures

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Data Center problems in one year
  - ~0.5 overheating (power down most machines in <5 mins, ~1-2 days to recover)
  - ~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hrs to come back)
  - ~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hrs)
  - ~1 network rewiring (rolling ~5% of machines down over 2-day span)
  - ~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
  - ~5 racks go wonky (40-80 machines see 50% packet loss)
  - ~8 network maintenances (might cause ~30-minute random connectivity losses)
  - ~12 router reloads (takes out DNS and external vIPs for a couple minutes)
  - ~3 router failures (have to immediately pull traffic for an hour)
  - ~dozens of minor 30-second blips for dns
  - ~1000 individual machine failures (2-4% failure rate, machines crash at least twice)
  - ~thousands of hard drive failures (1-5% of all disks will die)

Combining reliability

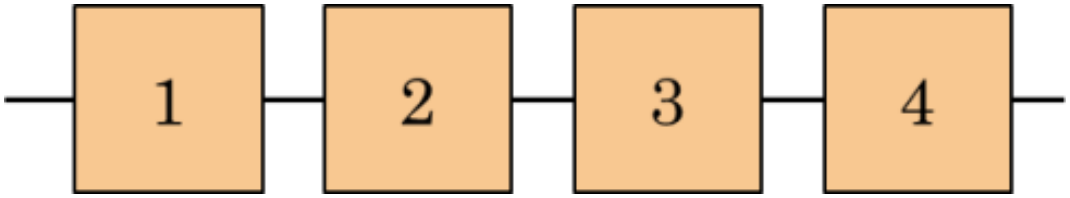
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Given N sub-systems
  - Each subsystem I has a failure probability r_i
- What is the overall failure probability?
  - Depends on how systems interacts
- Serial systems
- Parallel systems

Serial systems

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

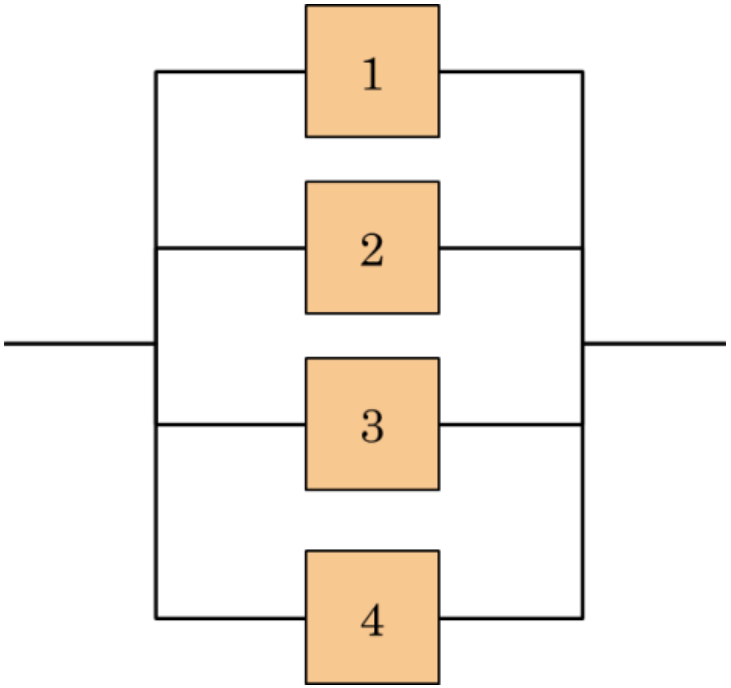
- Global failure → one sub-system failed
  - Like Christmas lights
- System reliability:
  - $R=\prod r_i$
- Systems fails if one sub-system fails
- System is available if every sub-system is available



Parallel systems

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Global failure → every sub-system failed
- System reliability:
  - $R=1-\prod(1-r_i)$
- System is available if at least one sub-system is available



Some questions

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

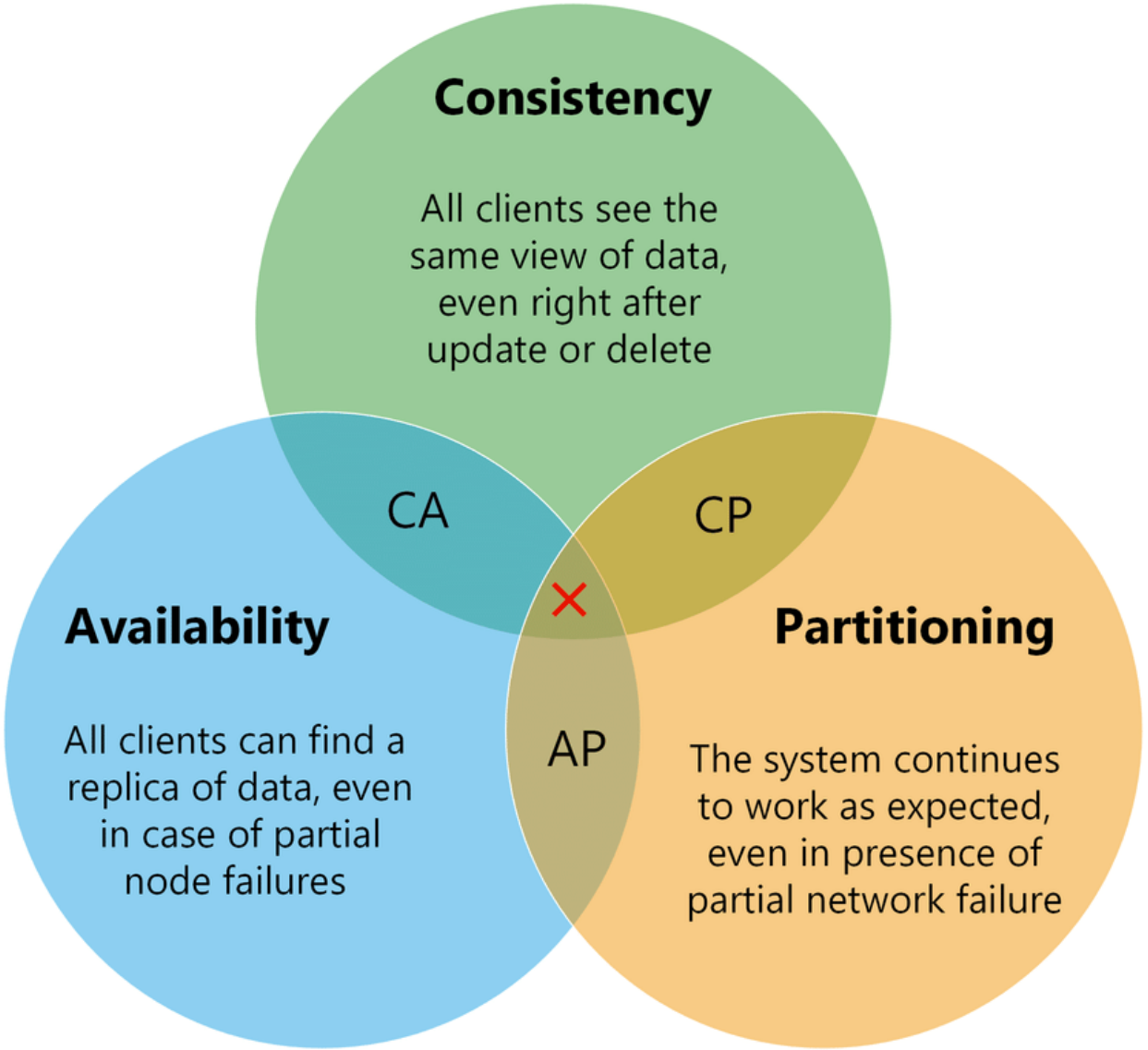
- Given subsystem with identical reliability
- In a parallel system
  - How many subsystem do I need to guarantee a reliability of X?
- In a serial system
  - What is the maximum number of subsystem to guarantee a reliability of X?
- Volunteers?

CAP Theorem

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- CAP
  - Consistency/Availability/Partition-tolerance
- Consistency: Every read return data from most recent write
- Availability: Every request returns a non-error response
- Partition-tolerance: The system works also in case of network partitions (messages lost or delayed)
- CAP theorem
  - Can choose only 2 out of 3 properties

CAP Theorem


Consistency
All clients see the same view of data, even right after update or delete
CA
CP
Availability
All clients can find a replica of data, even in case of partial node failures
AP
Partitioning
The system continues to work as expected, even in presence of partial network failure

Some useful hints

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Look out for bottlenecks
- Look out for single points of failure
- Keep it simple and stupid (KISS) 
  - Centralized control is simple!
- If it is not tested, don't rely on it
- Testing is not easy
  - Hundreds of services
  - Loosely couples services
  - Thousand of machines
  - How to deal with it?

Additional information

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Some useful readings:
- D. Menascé, V. Almeida,
  “Capacity Planning for Web Services”
- K. Trivedi, A. Bobbio,
  “Reliability and Availability Engineering”
- K. Trivedi “Probability and Statistics with Reliability, Queuing, and Computer Science Applications”

Workload forecasting

Forecasting

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Forecast:
  a statement of what is judged likely to happen in the future, especially in connection with a particular situation,
- Trying to understand how workload is evolving
- A critical part of capacity planning

A good forecast

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Not just a number but a set of scenarios
  - optimistic/average/pessimistic
- May involve many parameters of workload
  → Not just a change in workload intensity
- Examples:
  - Session arrival rate
  - Requests per session
  - Think time between interactions
  - Probability ratio in multiple services (workload mix)
  - Response size

Steps for forecasts

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Defining objectives (what are we trying to predict?)
- Select information sources
  - Interviews
  - Reports
  - Historical data
- Gather data (quantitative/qualitative)
- Build models
- Forecast
- Validation

Qualitative forecast

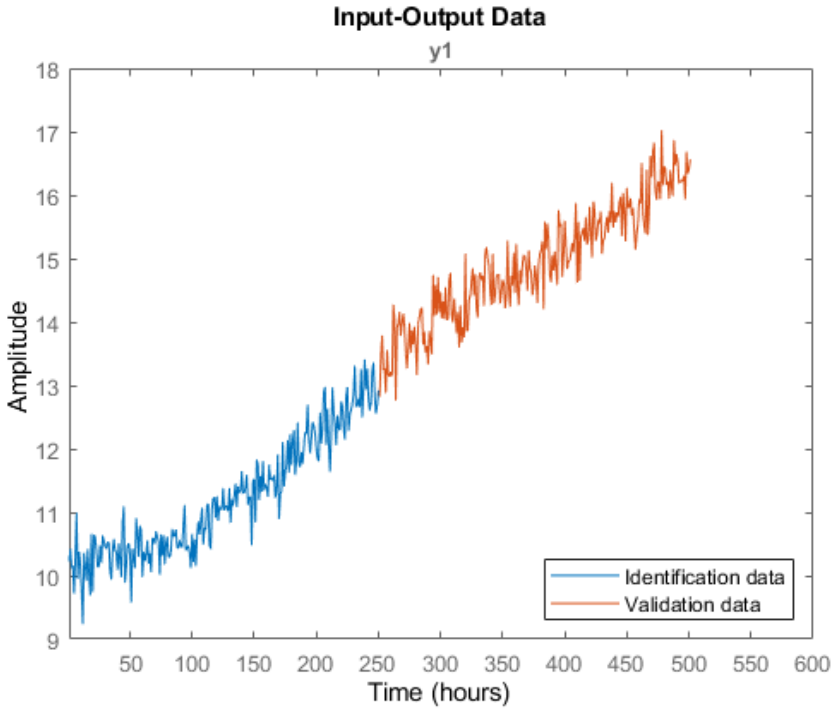
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Typically based on human sources
- Can be useful to build a reference scenario
- Examples:
- Interviews with end users
  - Questionnaires to a large set of people
- Collecting opinions of experts
  - More free-format interview
- Analysis of technological trends
  - Access to specialized reports

Validation

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

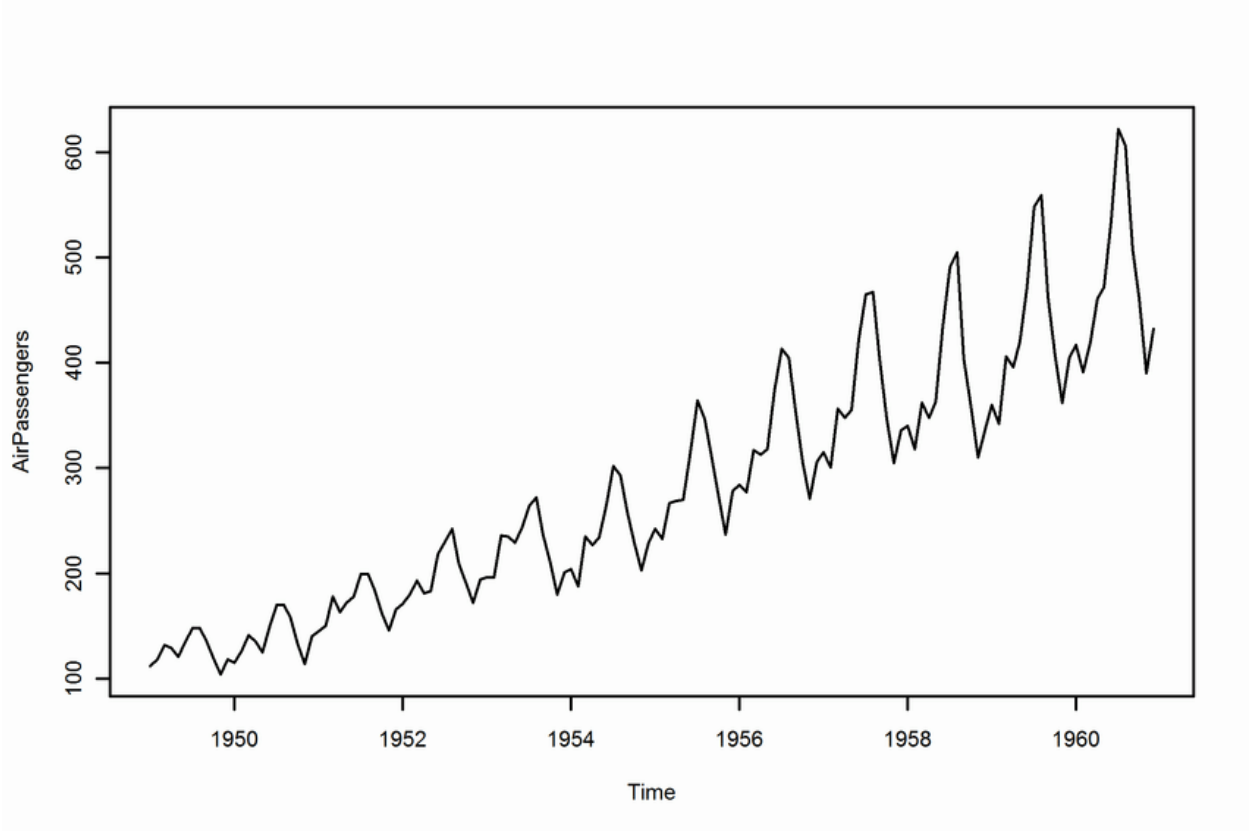
- Can use part of historical data
- Apply forecast to past
- It should describe the present
- Similar to Training/Validation process in ML



Historical data analysis

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Identify trends in time series
  - Linear trend
  - Periodic components



Error measures

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Mean error
  - $ME=\Sigma_t(Y_t-F_t) / n$
- Mean square error
  - $MSE=\Sigma_t(Y_t-F_t)^2 / n$
- Sum of square error
  - $SSE=\Sigma_t(Y_t-F_t)^2$

Linear regression methods

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Interpolation function
  - $Y=a+bX$
- Minimize square error (Least Square method)
  - $b= (n\Sigma_t(Y_tX_t) - (\Sigma_tX_t)(\Sigma_tY_t))/(n\Sigma_t(X_t)^2 - (\Sigma_tX_t)^2)$
  - $a= avg(Y)-b(avg(X)$

How predictable is a series?

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Typical test:
- Autocorrelation
  - Correlation between:
    - Time series X(t) and
    - Time series X(t-lag)
  - Evaluation as a function of lag
- Slow autocorrelation decay:
  - easy to predict
- Fast autocorrelation decay:
  - highly irregular time series
- Spikes for a specific lag value
  - Periodic behavior