Imperial College London

Performance evaluation teaching in the age of cloud computing

Giuliano Casale

Department of Computing

Imperial College London

TeaPACS workshop, FCRC, Orlando
June 19, 2023

Myself, My Teaching & This Talk

- >15-year teaching experience in Italy and UK
  - Computing degrees at Imperial
    - Extensive training in Softw., Programming Lang., AI/ ML
    - Project- and coursework-oriented teaching
  - Some teaching also at Politecnico di Milano, Italy
    - Very different, my observations mostly from UK system
- Most of this talk focuses on my performance evaluation and engineering (PE) teaching
- Goal: reflections on updating of performance topics at Imperial over a period of 10 years
  - Deeply subjective reflections, let's discuss!

Typical module @ Imperial/Computing

Module structure:

- 28 hours of frontal teaching (lectures and tutorials)
  - Relatively short, 4 hours a week for < 2 months
  - Lab time, where present, needs to fit the 28 hours
- 1-2 assessed coursework (cumulatively 15% of marks)
- 1 final exam (no oral examinations)

Teaching aids:

- Video recording
- EdStem for offline Q&A
- Some modules use flipped teaching

My Teaching @ Imperial College London

"Historical" performance module:

- Performance Analysis (1980s-2014)
  - Probabilistic modelling, queueing theory
  - Module ended due to staff retirement and lowering student numbers
- Performance Analysis (Half module, 2015)
  - Mean-value analysis only variant of earlier module

Issues with the above traditional modules:

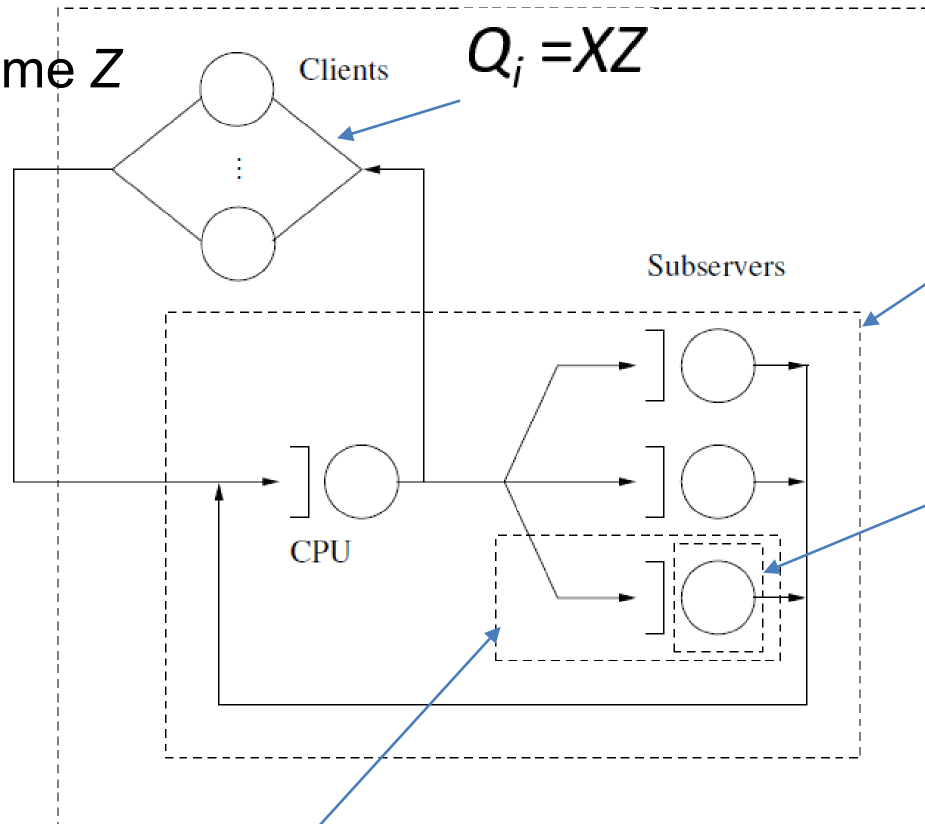
- Decreasing student interest (optional classes)
- Decreasing ability to make applicability case
- Other taught modules in the degree getting more "hands-on"
- Started experiments on what worked and what did not

Performance Analysis Evergreens

- Operational analysis, queueing theory, Little's law

Idle jobs
$Q_i = XZ$

Think time Z



Response time law
$R = N/X - Z$

Utilization law
$U_i = X D_i$

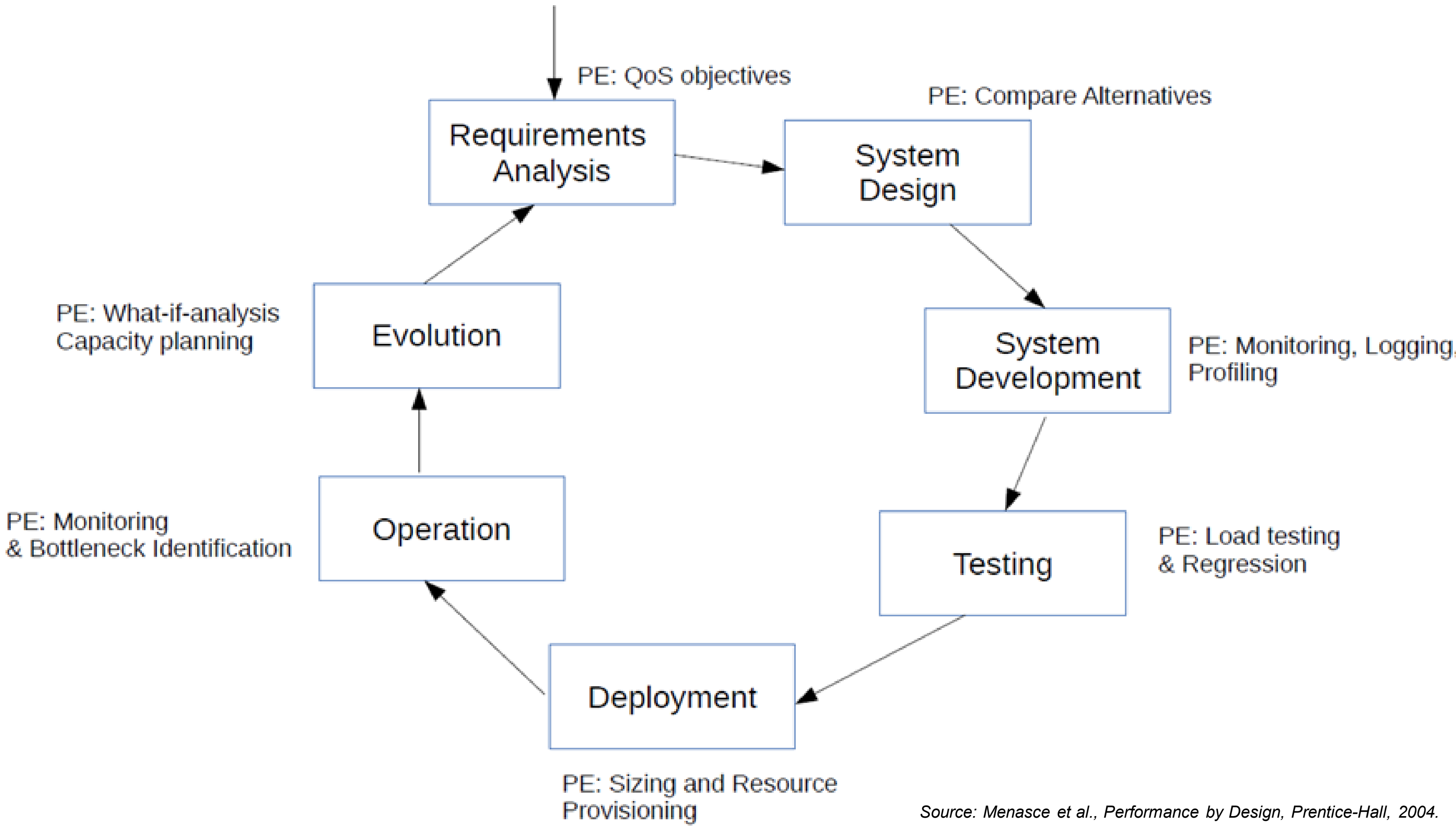
Cycle time
$C = N/X$

Queue-length/backlog
$Q_i = X_i W_i = X R_i$

Image: Lazowska et al.

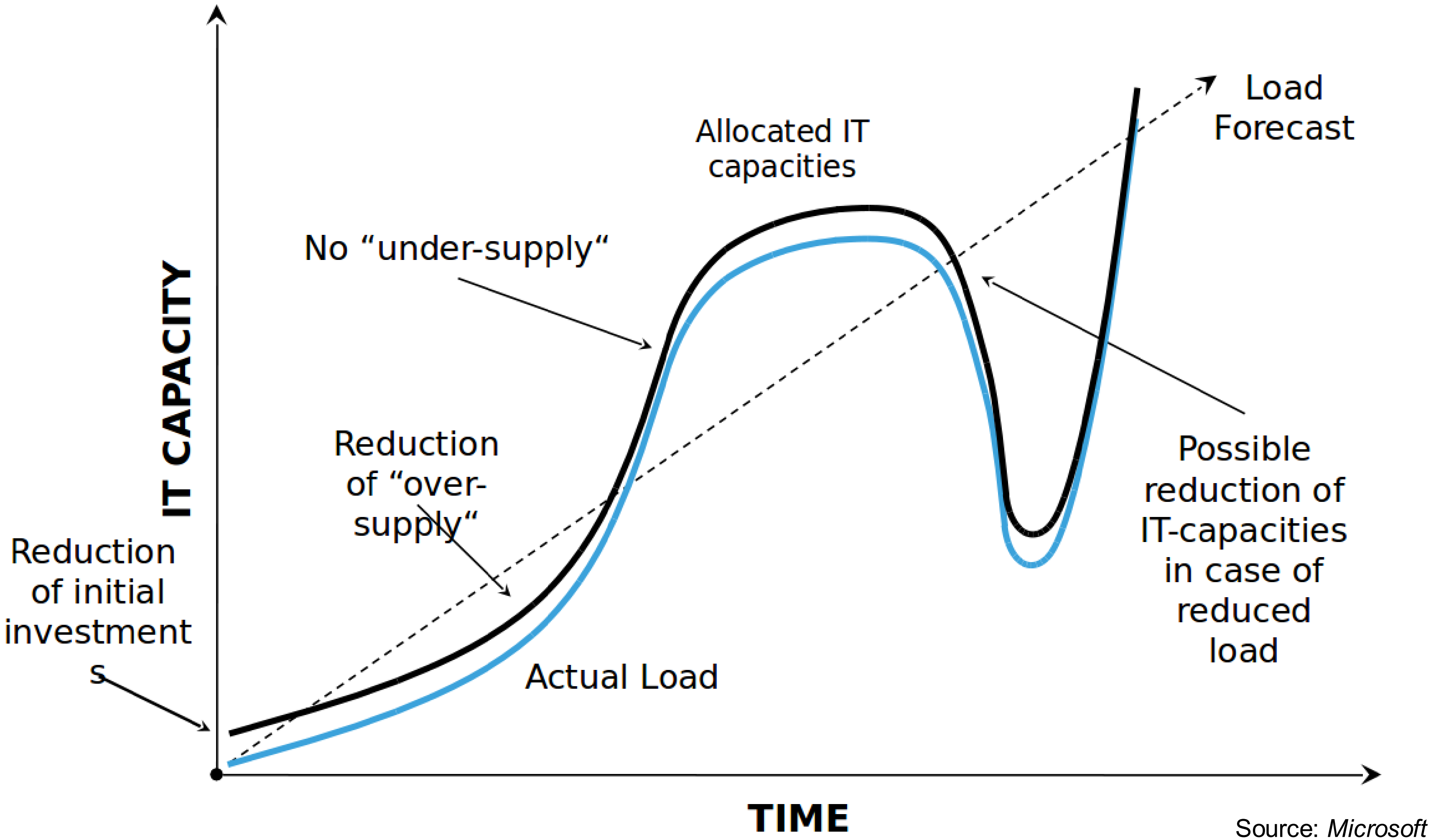
Classic PE Teaching

PE = Performance Engineer's Responsabilities



Source: Menasce et al., Performance by Design, Prentice-Hall, 2004.

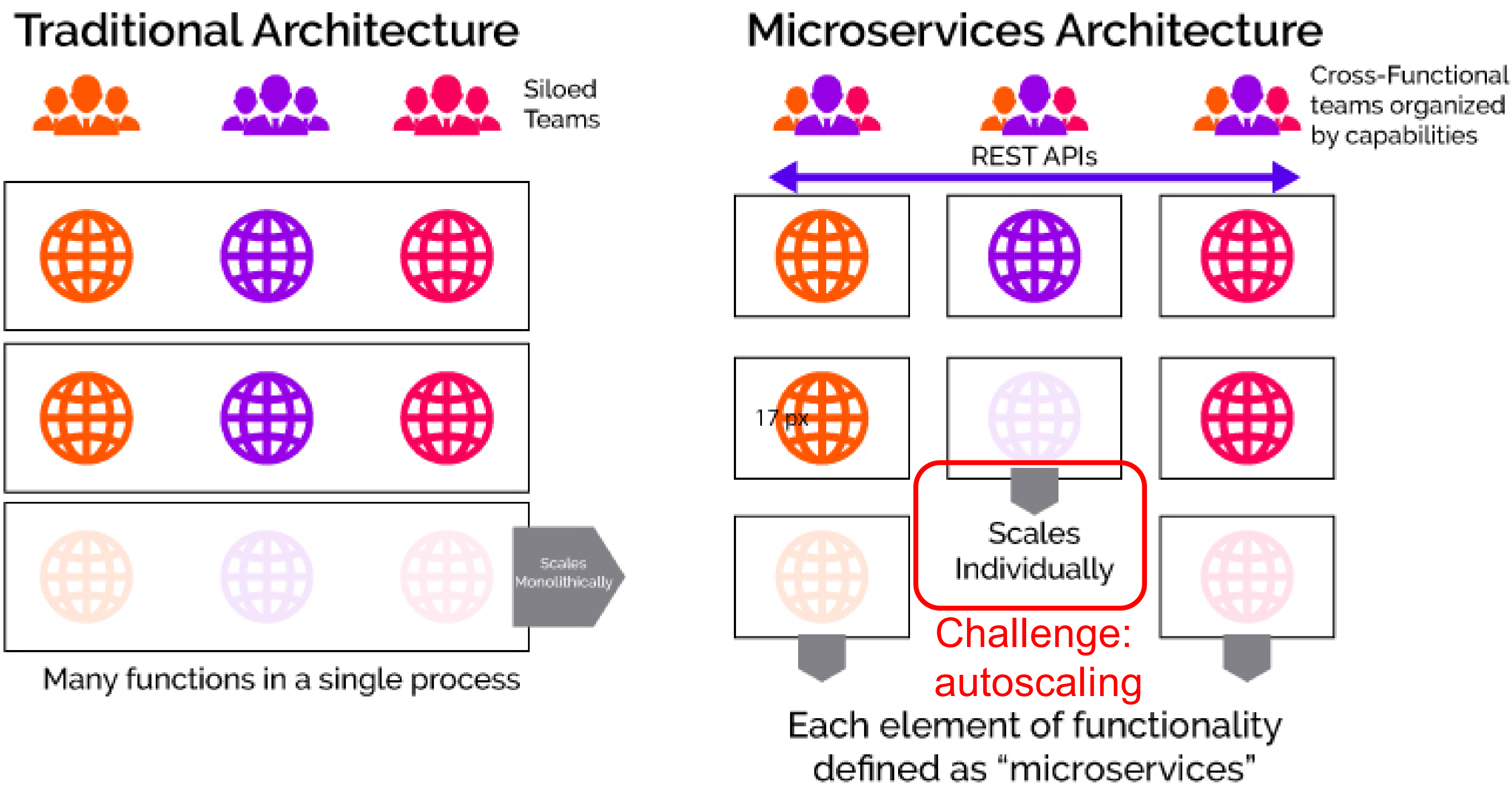
How did the cloud change this?



→ Capacity planning/sizing much easier now

The fading line between Dev & Ops

- DevOps practices (e.g., CI/CD)
- Microservices based architecture



Source: *xenonstack.com*

ICT Skill Evolution: the CMG case study

- CMG: historical PE practitioner conference

2003

- Las Vegas, >100 talks, crowded
- Technology
  - Windows, DBMS
- Capacity management
  - Sizing, ITIL, and planning
  - Load testing
- Stochastic modeling
  - Queueing & simulation
  - Forecasting techniques

2023

- Virtual (now called IMPACT), 33 talks
- Technology
  - Cloud services
  - Containers
- Observability
  - Distributed tracing
  - Continuous testing
- AI-based methods
  - System configuration
  - AI operationalization

Operational analysis & queueing brilliant, but what are they for now in ICT?

- Steady state? State is observed "continuously" & loads are highly volatile
- Fundamental laws (e.g., Little Q=XR)? We can monitor Q,X,R, etc. with ease!
- Forecasting? Capacity on demand & reactive methods are often good enough

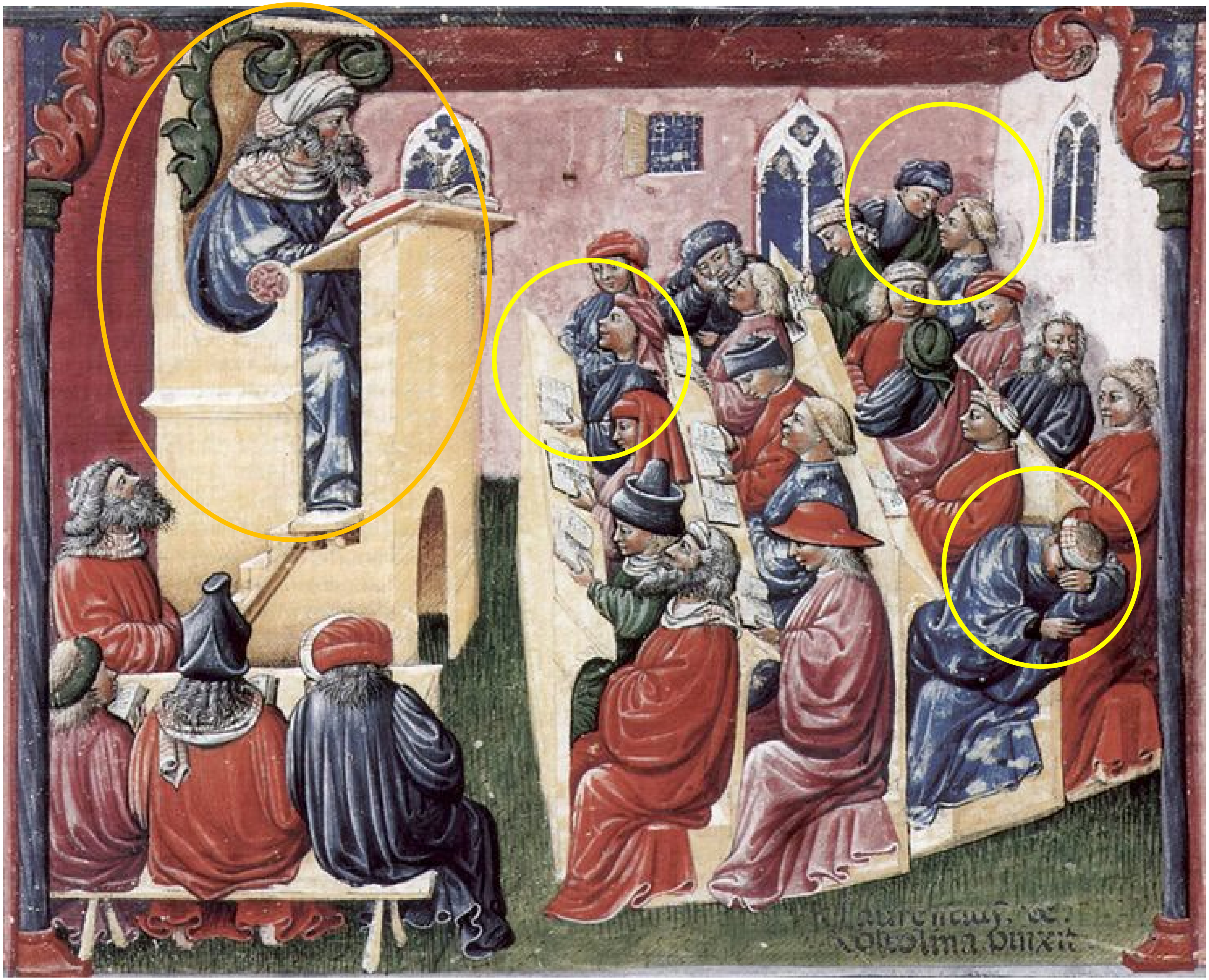
This prompted big dilemmas...

Shall we stick to classic performance evaluation or change the way we teach the subject?

Shall we form to train the next generation of researchers or follow trends/market?

Let's look at history, surely there must have been times where teaching was "dogmatically" focused on the discipline rather than on the market...

Academia in the Middle Ages



Role of module lecturers

- Middle ages (U. Bologna, 1100 AD c.a.):
  - Lecturers funded by societies of students
  - Students paid to learn skills, they drove the definition of module contents (!)
  - Students could fine academics for low teaching performance (!)
  - Even then teaching was for the students not for the discipline!
- My take:
  - Student education should take precedence over passion for the research field, we should be ready to drop classic subjects!
  - Still important to shape the mindset through deep problems and methodological teaching

PE Teaching @ Imperial College London

How my teaching changed as a result:

- Performance Engineering (2016-)
  - Hands-on exposure to cloud topics
  - Benchmarking, autoscaling, workload characterization
  - Hands-on lab experiments on Azure cloud

Methodological PE topics embedded in other modules:

- Simulation & Modelling
  - Discrete-event simulation, CTMCs, Poisson process, QNs, ...
- Probability & Statistics
  - DTMCs, Probabilistic modelling, Pareto, ...
- Scheduling & Resource Allocation
  - Deterministic scheduling (e.g., SRPT), workflows
  - Game theory, auctions

Some Cloud & PE teaching experiences

1. Configuration optimization
2. Resource allocation
3. Autoscaling
4. Workflow scheduling

Some Cloud & PE teaching experiences

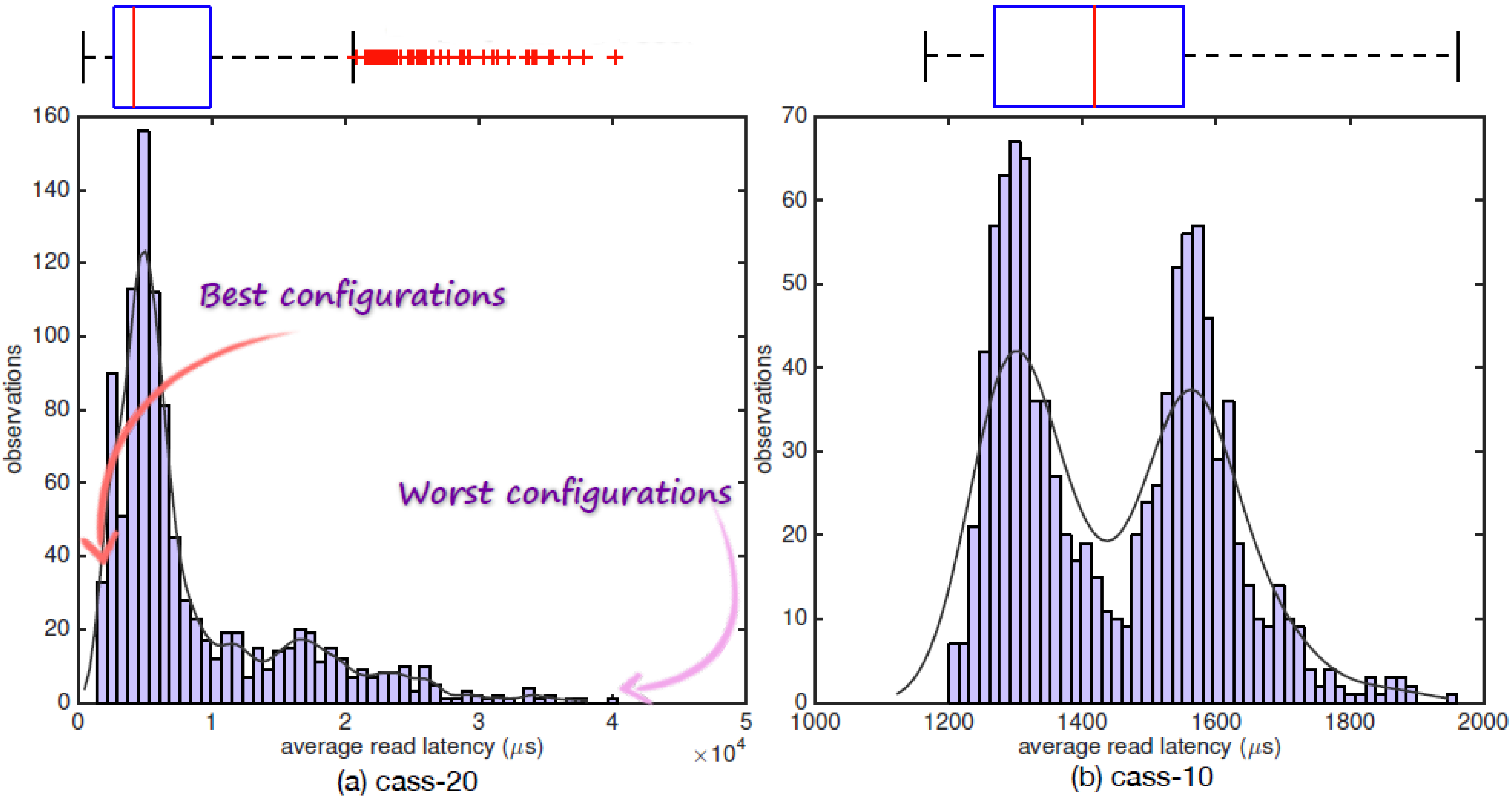
1. **Configuration optimization**
2. Resource allocation
3. Autoscaling
4. Workflow scheduling

Topic: Configuration optimization

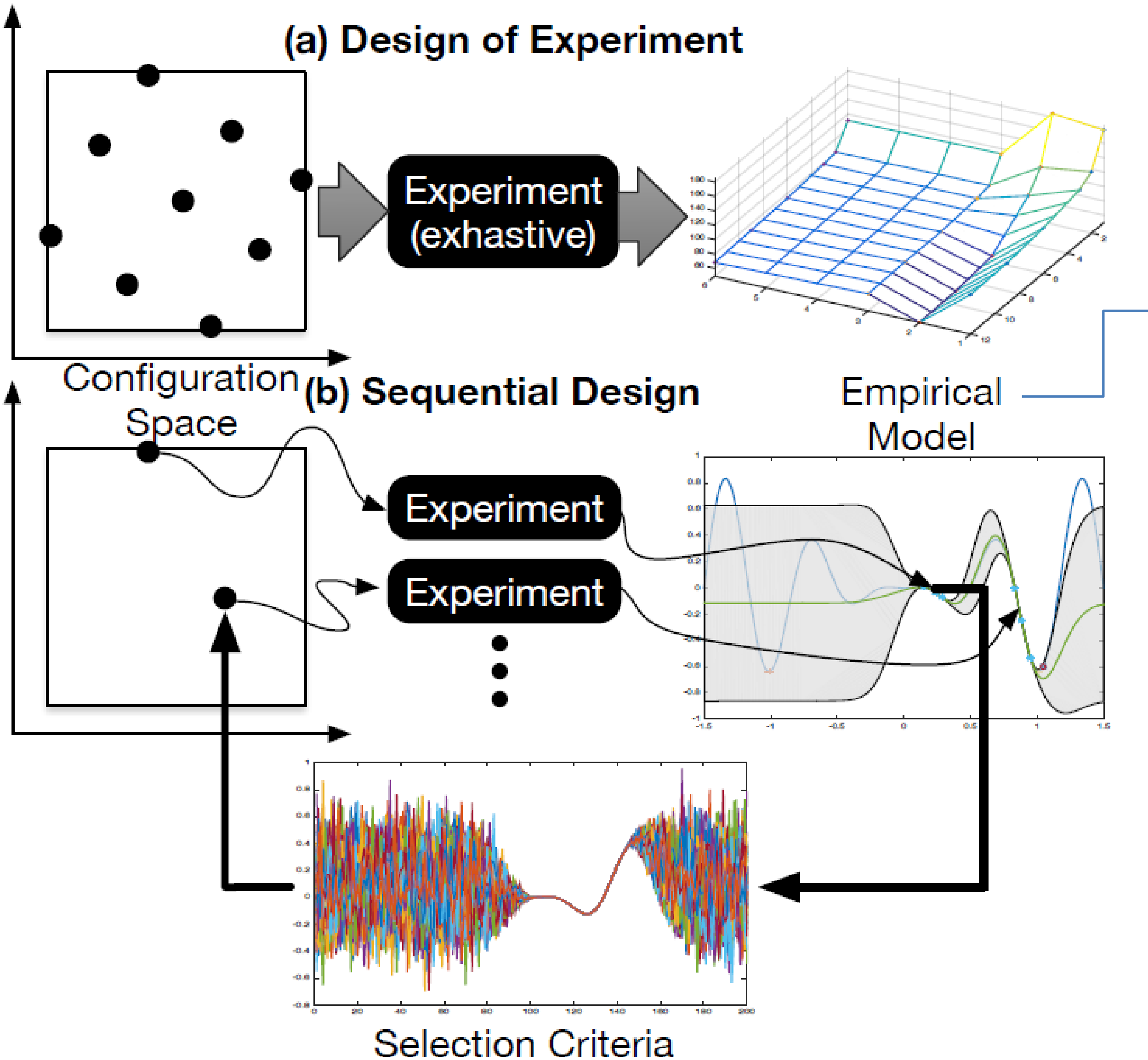
```
drpc.port: 3772
drpc.worker.threads: 64
drpc.max_buffer_size: 1048576
drpc.queue.size: 128
drpc.invocations.port: 3773
drpc.invocations.threads: 64
drpc.request.timeout.secs: 600
drpc.childopts: "-Xmx768m"
drpc.http.port: 3774
drpc.https.port: -1
drpc.https.keystore.password: ""
drpc.https.keystore.type: "JKS"
drpc.http.creds.plugin: org.apache.storm.security.auth.DefaultHttpCredentialsPlugi
drpc.authorizer.acl.filename: "drpc-auth-acl.yaml"
drpc.authorizer.acl.strict: false

transactional.zookeeper.root: "/transactional"
transactional.zookeeper.servers: null
transactional.zookeeper.port: null

## blobstore configs
supervisor.blobstore.class: "org.apache.storm.blobstore.NimbusBlobStore"
supervisor.blobstore.download.thread.count: 5
supervisor.blobstore.download.max_retries: 3
supervisor.localizer.cache.target.size.mb: 10240
supervisor.localizer.cleanup.interval.ms: 600000
```

Topic: Configuration optimization

- E.g.: Apache Cassandra avg read latency (1024 configs)



Topic: Configuration optimization



Gaussian process (GP):

- Non-parametric model, essentially a distribution over functions
- Effective modelling of known information (or lack thereof)

Reflections on Config. Optim. Teaching

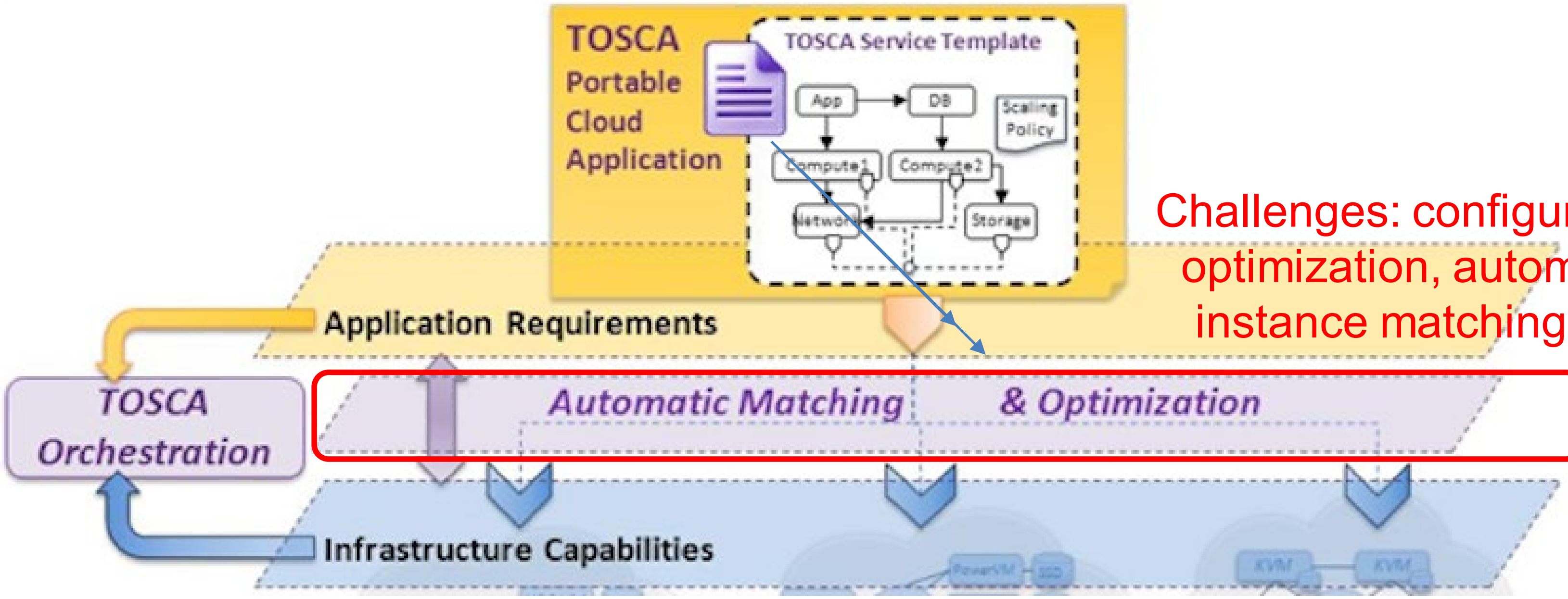
- ➕ Problem easy to understand for students
- ➕ Data-driven subject has an appeal to students
- ➕ Tight link to benchmarking
- ➕ Good topic to teach Design-of-Experiments, good books:
  - Jain, The Art of computer systems performance analysis, Wiley
  - Liljia, Measuring Computer Performance, CUP
  - Kounev et al., Systems Benchmarking, Springer
- ➕ Easy to setup coursework, e.g., optimize on/off options (eg hyper-threading ON or OFF, VM resources, program parameters)
- ➖ State-of-the-art is progressively diverging towards methods like Bayesian optimization that require ML background (e.g., GPs)
- ➖ Difficult to setup challenging exam questions, DoE often involves mechanical calculations
- ➖ More advanced exercises somewhat more AI/Stats. than CS

Some Cloud & PE teaching experiences

1. Configuration optimization
2. **Resource allocation**
3. Autoscaling
4. Workflow scheduling

PE Topic: resource allocation

- Instance matching in model-based DevOps



Challenges: configuration optimization, automatic instance matching, …

Image: xenonstack.com

Example: Cloud Load-Balancing

- Weights = Visits
- Jobs = User sessions
- Optimal weights found with an optimization program:

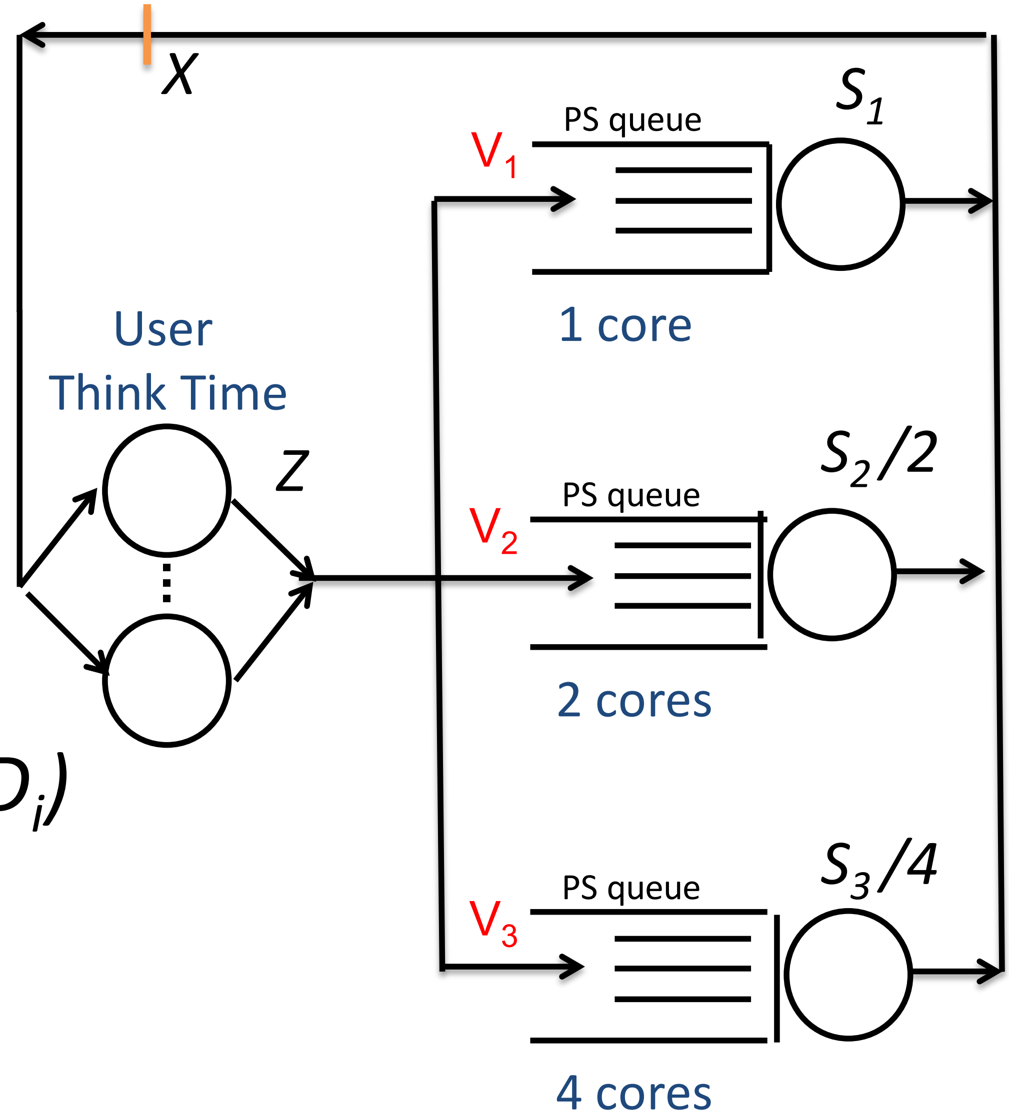
max X

s.t. $X = B\text{-}S\_approx(N, Z, D_i)$

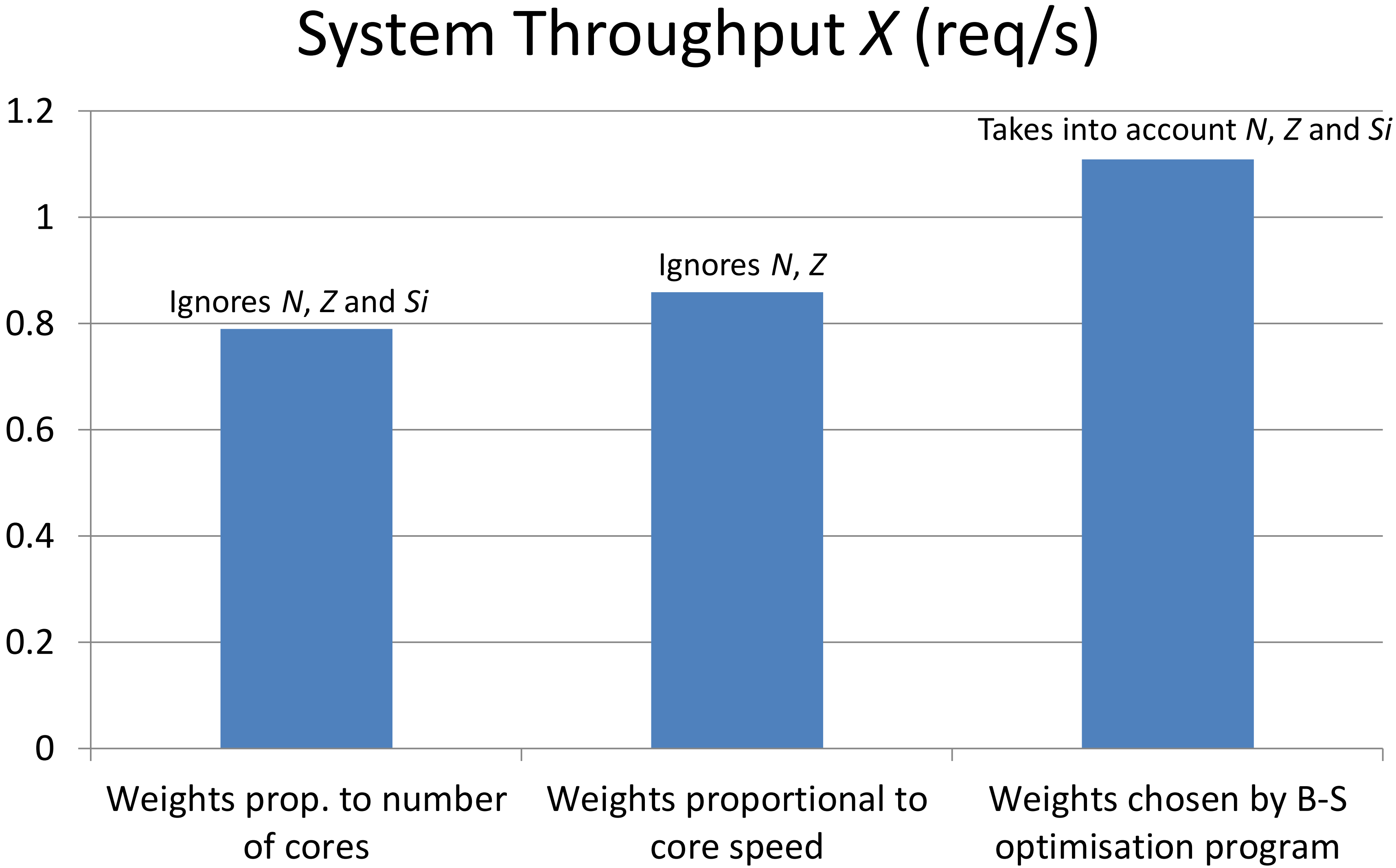
$D_i = V_i \, S_i$

$V_1 + V_2 + V_3 = 1$

$V_1, V_2, V_3 \geq 0$



Example: Cloud Load-Balancing



W. Wang, G. Casale. Evaluating weighted round-robin load-balancing for cloud web services, SYNASC 2015.

Reflections on Res. Allocation Teaching

- ➕ Analytical or simulation models intuitively needed
- ➕ Easy to define simple but rich stochastic models, e.g:
  - Heterogeneous resources (e.g., queue rates)
  - SLA percentile constraints (e.g., response time distributions)
  - Bare metal contention (e.g., multi-tenancy/multi-class)

- ➖ Numerical solution methods (e.g., AMVA) require extensive background (e.g., operational analysis, single-class/multiclass QNs)
- ➖ Requires some nonlinear optimization background (e.g., KKT conditions, metaheuristics, ...)
- ➖ May require concepts of multiclass inference (e.g., linear regression of service demands, ….)
- ➖ Difficult to setup search-based exam questions

Some Cloud & PE teaching experiences

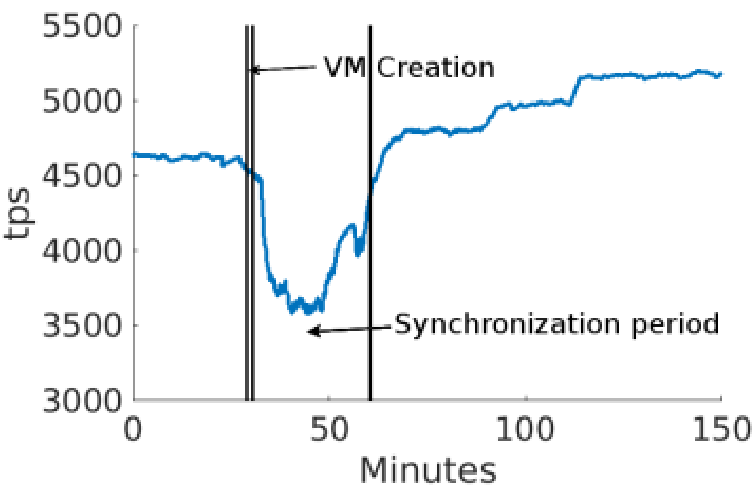
1. Configuration optimization
2. Resource allocation
3. **Autoscaling**
4. Workflow scheduling

Example: predictive autoscaling

I ask my students:

- Why do we need predictive modeling?
- Could reactive autoscaling be enough?

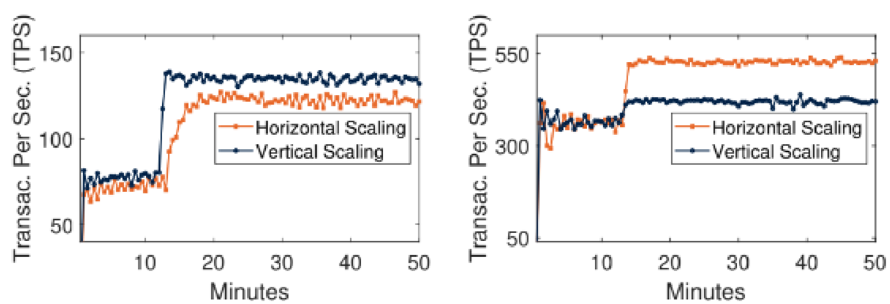
Cassandra autoscaling: with cost of data synch.



Unpredictability of horiz./vert. scaling effects in microservices

Table 1: Two cases where the front-end microservice is the bottleneck

Case	System Workload					Front-end Config.	
	Request Distribution			Conc.	Think	CPU	Replica
	Home	Catalog	Carts	Users	Time	Share	
A	57%	29%	14%	1000	7 sec	0.2	1
B				4000		1.0	



(a) Case A (b) Case B

Reflections on Autoscaling teaching

- ➕ Easy to link with forecasting, queueing modelling and control theory
- ➕ Methods from control/forecasting simple and easy to examine
- ➕ Several students displayed significant excitement

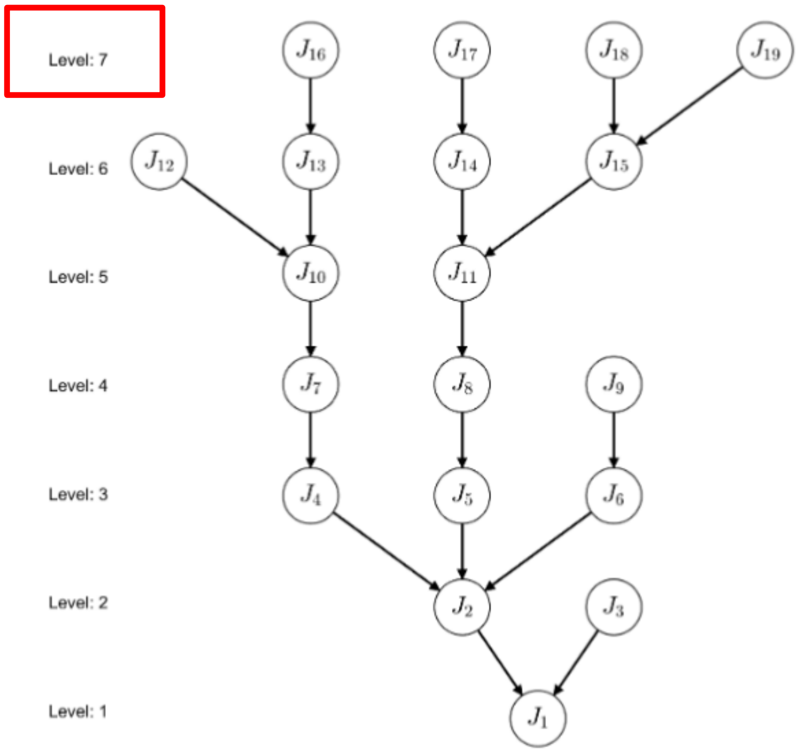
- ➖ Slides need regular updates as technologies evolve
- ➖ Difficult to setup a hands-on experience
- ➖ Specific elements of the theory are a little shallow (e.g. rule-based autoscaling)
- ➖ Forecasting (e.g., ARMA) and control theory methods somewhat more suited to EE than CS students

Some Cloud & PE teaching experiences

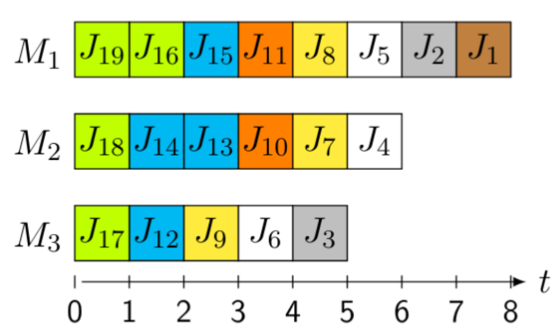
1. Configuration optimization
2. Resource allocation
3. Autoscaling
4. **Workflow scheduling**

Example: workflow scheduling

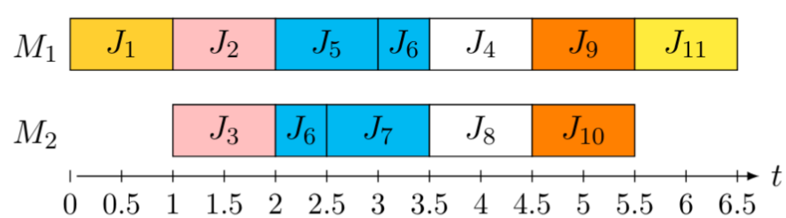
Critical paths in workflows



Parallel machine scheduling



Preemptive vs non-preemptive jobs



Reflections on Workflow sched. teaching

- ➕ Excellent appeal to students, they see they are important
- ➕ Used in the real world
- ➕ Exposure to practical NP-hardness issues
- ➕ Easy to pair with metaheuristics, which our students love
- ➕ Coursework based on Azure Functions
- ➕ Results of SIGMETRICS people (e.g. Muntz-Coffman)

- ➖ Large volume of demands for individual projects
- ➖ Mostly deterministic modelling
- ➖ Difficult to broaden to workflow management engines

Final thoughts & recap

- The performance community used to have a "unified" theory based on queueing and OA, no longer the case.
- PE is too empirical at times. Theory remains important to shape student intuitions and understanding.
- Cloud engineering problems richer and more exciting than traditional sizing problems
- Academic cloud credits an easy way to engage students in hands-on measurements
- Frontal teaching hours a hurdle to PE teaching in systems like the UK (not enough time for full background)
- Easier for me to spread PE problems and techniques in other modules than having a single PE class