Performance evaluation teaching in the age of cloud computing

Giuliano Casale
Department of Computing
Imperial College London
gcasale@ic.ac.uk

ABSTRACT

Cloud computing has been one of the most significant developments in computer science of the last two decades, fostering sharp changes in performance engineering practices across the computing industry and, at the same time, profoundly steering research trends in academia. A distinctive trait of this paradigm is that cloud engineers can programmatically control application performance, raising an expectation for computing graduates who find employment in software and system development to have basic performance engineering skills. This, in my view, calls for a broader and deeper education on software and system performance topics as part of the computing curriculum, while at the same time requiring a rethink of the syllabus of a classic performance evaluation module. This abstract presents my personal experience in doing so, including a discussion on the educational strengths and weaknesses of performance engineering emerging from cloud computing practice.

1. INTRODUCTION

If we wish to reflect on how we should teach performance-related topics in university modules, I believe a good place to start is to ask ourselves what professional profiles can leverage the methods and results developed in performance evaluation and engineering (PE), and whether the way the community teaches the discipline is appropriate in educating students for these roles. Indeed, university education ultimately has a mission to prepare students for their careers, therefore such a reflection is in my view needed to avoid designing modules based on personal preferences alone.

Several professions can surely make good use of the tools, insights, and fundamental laws developed over the years by the performance evaluation community. For example, network traffic engineers can benefit from a deep understanding of queueing theory and point processes. Similarly, data scientists are often confronted with understanding complex stochastic phenomena from data, a challenge that shares similarities with performance measurement and workload characterization. Simulation and modelling tools are widely used by professionals in enterprise management and planning (e.g., logistics). Consultants need to design appropriate workloads to find bottlenecks and determine the performance limits of systems built by their customers. Classic PE topics, such as stochastic modelling, performance measurement, and operational analysis, can therefore still undoubtedly play a role in shaping the minds and strengthening the preparation of our students for all these professions and tasks. Yet, in this paper, I take a different view on the suitability of classic PE teaching topics for educating students who wish to understand performance in the context of software systems. This is an important student group, since software engineering is one of the topics that regularly motivates students to choose computing degrees. In particular, I believe that academics who wish to tailor their courses to prepare students to deal with software performance engineering in the age of cloud computing should consider to:

1. reduce the prominence in PE modules of stochastic modelling topics, part of which may still be taught within other modules, such as probability and statistics courses;

2. refresh the PE module syllabus with novel topics, problems, and examples emerging from cloud computing, striking a balance with classic PE concepts and theory.

3. feature in the PE module a “hands-on” component to foster a better understanding of real-world performance issues that arise in cloud systems and techniques available nowadays to monitor and dynamically address such problems while the system is in operation.

The above recommendations are subjective views that reflect my personal experience in teaching PE for the last 15 years. The paper intends to expose my opinions and suggestions in an informal way. I give in Section 3 an overview of the evolution of PE teaching within the Department of Computing at Imperial College London to illustrate an instance where the above changes were applied in practice and positively impacted student participation.

Concerning recommendation 1), stochastic modelling has often been justified in the context of PE with a need to predict system behavior under unseen, or uncertain, operational conditions. Problems such as capacity planning and design-time software performance analysis have been put forward to generations of students to motivate the need for PE. However, in my view, the whole idea of predicting the behavior of complex software and services in advance of their deployment is less viable and widespread today than it used to be at the time those earlier PE modules were designed. At the heart of the issue lay various technological developments, mostly related to cloud practices, which the paper discusses in Section 2.

Stemming from the above, several classic stochastic tools still taught in long-running PE modules, ranging from Markov chains to queueing theory and Petri nets, appear somewhat less relevant to me for professional practice in nowadays cloud and software engineering practice. Recommendation 2) suggests that other subjects may be used to refresh the PE module syllabus. As explained in Section 4, a number of performance-related problems and topics emerging from cloud computing may be used to replace, fully or partially, the stochastic analysis subjects of a traditional PE module. In many cases, such topics still offer space to integrate classic PE notions into the lectures, allowing the module to strike a balance between new and old PE content. Section 4 also elaborates on my teaching experience with these new topics and their perceived value and shortcomings from an educational standpoint.

Concerning recommendation 3, it is now routine for software and service engineers to control software and system performance programmatically, for example leveraging cloud APIs to scale the compute and network resources available to an application. Moreover, leveraging the diffusion of DevOps practices [1], changes to a cloud system are increasingly often performed after its deployment, shifting the emphasis of quality-of-service engineering from "correct design" to "problem fixing at runtime". This state of play raises, in my view, a need for computing graduates to receive, as part of their education, some basic training on how to address performance problems for a live cloud system while in operation. Configuring cloud auto-scaling may be an example of a basic skill required by many employers. This requires the ability in a student to reason on how the capacity allocated to a software system should be varied dynamically to ensure that the desired level of performance is maintained despite workload fluctuations. Auto-scaling systems vary among different cloud platforms but, at least in their basic implementations, they often require a limited theoretical preparation, making them well suited for self-contained lab exercises. Topics such as distributed tracing and benchmarking are also well-suited to increase student familiarity with these problems. In my view, lab components can therefore enrich PE teaching and help convince students to opt in for such modules through a better alignment with cloud engineering practice.

2. EDUCATING FOR THE PROFESSION

PE is a continuously evolving discipline, allowing us to look back at its trajectory over the years to spot how today's PE differs from the one for which classic PE modules were designed. As mentioned, since the majority of students seek university education to prepare for a career in companies, I believe it is important to look at the application of performance concepts in a practical setting to steer the educational offering proposed in academic PE modules.

One possible way to do so is to look at the notion of performance discussed in practitioner events. The focus of this section is therefore on observations on the evolution of the Computing Measurement Group (CMG) [12], a prominent community of performance practitioners. This community featured at the height of its popularity several international branches, which organized local events and brought together several hundreds of people worldwide with a keen interest in the topic of performance evaluation. The community also ran the popular CMG conference for many decades, which gathered contributions from industrial practitioners and scholars in the PE field. In a typical year, the CMG conference program would include presentations from experts and capacity management teams, for example operating in banks or in ICT consulting companies. Notably, despite its practical scope, the event included presentations from research scholars, demonstrating the presence of a healthy interaction between the research and the practitioner communities.

For a historical perspective on the notion of performance discussed at these events, the reader may look at the survey in [12], which focuses on themes touched upon in the 1970s and 1980s. Taking CMG 2003 as a more recent example, the conference mostly featured papers on mainstream technology (e.g., Windows, DBMS, z/OS mainframes, . . .), capacity management methodologies (planning, ITIL, sizing, measurement, QoS), and stochastic modeling (queueing, simulation, forecasting), demonstrating, at least in the last two topic areas, an overall good alignment with the themes of both PE research and education of those days.

Let us now move forward 20 years to the present day. The same organization now brands itself as a technology community, featuring a more industrial composition. Topic-wise, the conference program now focuses on more recent infrastructure technologies (cloud databases, cloud servers, containers, ...), observability (monitoring, distributed tracing, ...), and AI (automated system configuration, operationalization, ...). Interestingly, aside for monitoring and mainframes, few contributions overlap significantly with the topics of older conference editions. From an educational and research standpoint, it is striking in particular to see a near absence of tools from classic PE theory, such as Markov chains, queueing theory, but also a modest adoption of simulation-driven analysis. Clearly, a dramatic shift in PE practice has occurred over the last two decades. In my opinion, this suggests that PE educators should take notice of this change along the lines of recommendation 2 in the introduction.

The reasons for such a shift in PE practice are surely multiple. It may still be possible to conjecture on what some of these might be so as to inform how PE education should change as a result. To begin with, computer systems are generally much more complex than in the early days of PE, and basic analytical and simulation models tend to perform less accurately in complex systems than data-driven AI models. AI models also take a black-box approach to the problem that broadens their applicability, whereas stochastic networks typically require some knowledge of how the system works, and in this sense they are much more time-consuming to develop, besides requiring skilled professionals.

There may also be several other reasons why classic PE modeling tools are not as widespread as they used to be. For example, if cloud engineers can quickly and cheaply correct performance problems on-the-fly, allocating and removing capacity in a matter of minutes, and thus the cost of resource allocation mistakes is low and easily fixable, reactive methods in use in many cloud systems, albeit not perfect, are sufficient for many companies. Moreover, if monitoring is cheap, ubiquitous, and allows the observation at a high-frequency of live distributed systems, steady-state analysis via a model may be a less appealing way to characterize performance than directly doing performance troubleshooting on the running system. And even in situations where steady-state predictions can be helpful, with applications that are densely layered, built on top of a stack of platform services and software-defined infrastructure (e.g., IaaS, containers, JVMs, serverless, etc), such predictions may become brittle, as the models can only characterize a small slice of the overall system.

Summing up, the technological evolution driven by cloud computing has shifted the attention of PE practitioners in the software domain to observing systems in a live environment and to using AI models to reason on their performance. The cost of applying hand-crafted performance models remains high and companies face skill shortages of staff experts in building such models. As suggested in the recommendations in the introduction, this warrants in my view a major rethink of which PE topics should be taught in academia in the perspective of better preparing students in computing degrees towards careers in software and service engineering.

3. PERSONAL TEACHING EXPERIENCE

In many discussions I have had over the years with colleagues in the PE field, the subject of a perceived change of tastes of the students in recent years has come up many times. This circumstance is one that I have also observed myself in the PE courses taught within the Department of Computing at Imperial College London. The goal of this section is therefore to give an example of how my PE teaching has evolved in response to the challenges described in the previous sections.

In the Department of Computing at Imperial College London taught modules normally involve 28 hours of frontal teaching, including both lectures and tutorials. The latter may be either in-class tutorials or lab sessions. Classes in the last two years of the curriculum, where PE topics are typically taught, include a mix of both undergraduate and Master's students. A short module duration of just 28 hours, which maps to a mere 4 hours a week for 7 weeks, is not uncommon in the UK. It implies that module lecturers need to carefully select their syllabus, as it would be difficult for example to present the content of an entire book in such a short time span. Assessment is typically based on coursework and a written exam.

A *Performance Analysis* module was taught for decades in our department, which students could optionally select in the last year of their studies. The Performance Analysis module was focused on probabilistic modelling, with an emphasis on Markov processes, queueing theory, and numerical exercises. The course ran smoothly for many years and students were generally keen (if not fascinated) to explore the mathematical techniques available for computer system analysis, such as Markov processes, queueing systems, queueing networks, Petri nets, stochastic process algebras, and others. However, the recent generations of computing students showed a decreasing interest in stochastic theory, possibly as a result of the factors outlined in the previous sections and the concurrent rise of AI, which has shifted the interest of those keen on mathematical modelling onto subjects related to machine learning. A decision was taken to end the long-running *Performance Analysis* module in 2014, replacing it with a module with the title but, due to other staffing needs, with half credits (14 hours of frontal teaching) and an entirely new syllabus.

For the new course, which started in 2015, I decided to soften the mathematical density of the module by switching to a mean-value analysis (MVA) based PE teaching, centering the presentation on materials covered in the classic PE book by Lazowska *et al.* [10], refreshed with real-world examples from cloud computing research and industry. A distinctive feature of Lazowska *et al.*'s book is the ability to present essential performance analysis theory without entering either into a probabilistic description of the system or formal proofs, but rather focusing only on operational analysis [3]. However, students kept demonstrating a limited appetite for queueing theory even if this was presented in a mathematically simpler form.

In the years that followed, I then looked to change my PE teaching more fundamentally, significantly reducing the focus on stochastic modelling, and seeking a stronger case for applicability and relevance to professional practice. This resulted in much-improved participation and an overall higher level of engagement from the students. The change consisted mainly of the following modifications. Firstly, a new *Performance Engineering* 28-hours module started in 2016 with the syllabus focused on the intersections between PE with cloud computing, but taking primarily a systems engineering and measurement view. The module wanted to retain some methodological elements of classic PE (e.g., Markov chains to describe user workload patterns). Moreover, this was the first course in our department to teach hands-on cloud computing basics to our students, including a lab-based coursework that leveraged an educational sponsorship from Microsoft Azure. In these lab sessions, the students could learn the basics of instantiating and sizing VMs and conducting benchmarking and workload characterization experiments, followed by coursework centered on measurement and autoscaling. This change introduced a practical side to PE teaching that students really enjoyed. In a handful of years, the module grew from 14 students in 2016 to 96 students in 2020. Throughout this period, a colleague joined me in teaching the module, introducing, among others, topics related to cache performance and monitoring hardware performance counters, further strengthening the exposure of the students to systems performance.

At the same time of the above developments, I decided to spread the more theoretical aspects of PE (e.g., Markov chains, scheduling, point processes) horizontally across other modules I taught. These included modules on probability and statistics (2nd year), scheduling (3rd year), and simulation (3rd year). My experience was that presenting topics such as stochastic processes and queueing theory within theoretically-focused modules that targeted a broader problem space then PE worked better in terms of the student reception. I also noticed that exposing some performance concepts only in a determining setting (e.g., deterministic scheduling) worked better for many students than looking at similar problems through a stochastic lens.

Summarizing, the recommendations provided in Section 1 may be seen as the key takeaways of the evolution of my teaching discussed in this section. Evolving the *Performance Engineering* module format with the inclusion of topics from cloud computing allowed me to address what seemed a progressive decline in student numbers and interest in PE. Several important topics that were traditionally taught in that module, such as Poisson processes and queueing theory, have then been spread horizontally across other modules in the degree, obtaining a warmer reception, possibly also due to a different composition of the student body.

4. TEACHING TOPICS

Leveraging the experience gained in teaching the modules described in the last section, this section presents some recommendations for cloud-related topics that may be included in a modern PE syllabus. My goal, in particular, is to review topics that integrate well with classic PE materials, allowing the lecturer to still teach several classic notions of performance evaluation but in a renewed context.

4.1 Configuration optimization

The topic of configuration optimization deals with elaborating strategies to automatically tune the configuration parameters of a software system so as to maximize some performance measures [7]. For example, in a Big data platform such as Apache Storm, hundreds of configuration parameters need to be assigned. Such parameters can profoundly influence the performance of cloud systems, which commonly rely on these open-source platforms.

As traditional queueing theory models focus on a narrow set of system parameters (e.g., number of service stations, server multiplicities, service rates, ...), this topic allows the students to develop a broader view of factors that affect performance. The topic can also help them develop data-driven modelling skills by looking at system response surfaces. The topic is also useful to foster reasoning on how to find optimal solutions from such surfaces.

Teaching benefits. In my experience, the configuration optimization topic is generally interesting to students, as it is easily understandable and a useful skill to gain due to its generality. A benefit of its teaching is that it has a tight link to benchmarking, which can be used to gather the data to fit the response surface. This then naturally leads to cover in the module associate topics such as design of experiments, the structure of a modern benchmark (such as the SPEC benchmarks[1]), surrogate modelling, and regression model fitting. Books such as [9, 11, 8] offer excellent source materials to develop lectures on the topic.

Another benefit of the topic is that it is fairly easy to set up coursework or lab-based exercises. For example, students may be asked to optimize a set of on/off options for a system, e.g., enabling and disabling hyper-threading on a CPU while observing the resulting performance changes of a service, or changing the flags passed to a compiler, followed by performance profiling of the compiled executable.

Teaching challenges. On the downside, this topic is rapidly evolving in the state of the art, and thus may require frequent updates to the slides. The analysis of surrogate response surfaces to determine optimal system configuration may also be best conducted if the students have some machine learning background (e.g., Gaussian processes) and a basic understanding of related optimization methods (e.g., Bayesian optimization). The topic may also be somewhat difficult to test in an exam setting, where limited data and mathematical calculations can be expected.

4.2 Cloud deployment

Cloud computing systems are increasingly deployed by means of declarative models of infrastructure resources, as in AWS CloudFormation[2] or OASIS TOSCA models[3]. An orchestrator then takes in input such specifications and deploys the application on the requested resources, instantiating them on-the-fly on the cloud. Yet, the amount of resources requested for an application is a controllable parameter that the application owner needs to decide. This opens interesting questions on how the performance engineer should decide which and how many resources to allocate to the application. This mindset brings into play questions similar to capacity planning, but from a different angle.

Teaching benefits. Analytical models may be justified with this problem as allowing to model performance within a computational optimization program used to reach a resource allocation decision. In its simplest form, to avoid the complications of stochastic models, this may be just a linear program that depends on the utilization levels of the resources, since operational analysis requires for these only simple laws [3]. Where desired, methods to select specific resource types (queue service rates), to respect service-level agreements (e.g., response time distributions), or to account for bare metal contention (e.g., multi-tenancy/multi-class) may also be studied, if the chosen modelling formalism is sufficiently expressive to address these problems.

Teaching challenges. Similarly to the configuration optimization topic, also this topic has the drawback of assuming some familiarity in the students with computational optimization methods, such as mixed-integer linear programming. Describing multiclass systems may require additional time and place additional complexities in explaining how to parameterize the relevant models (e.g., regression-based estimation of service demands). Also in this case, setting up exam questions that deal with computational optimization formulations presents challenges, as the student cannot derive by hand the solution at the exam. Assessment may be therefore lab-based or limited to writing the optimization program formulations without an explicit solution.

4.3 Autoscaling

As mentioned earlier, autoscaling is an essential topic for cloud engineering [13, 6]. The topic can be integrated within a PE module at varying degrees of sophistication (e.g., reactive vs. proactive autoscaling). The topic also requires the lecturer to introduce concepts of system transient and steady-state, since the time for the system to settle after a scaling decision is a parameter that affects the configuration of some autoscaling rules. Autoscaling is also easy to couple with load balancing topics, allowing again to integrate well with materials from classic PE theory.

Teaching benefits. Several students displayed significant excitement for the autoscaling topic in my modules and generated a follow-up demand for thesis supervision. The topic is easily linked with mathematically-rich topics such as forecasting, scheduling, and control theory. Methods from forecasting such as autoregressive and moving-average processes are appropriate for computing students as they require just basic elements of conditional expectations and Gaussian distributions, and the fitting of small models (e.g., AR(1)) requires simple algebraic formulas. Exposure to forecasting is also helpful to students who seek to build a career in software services for the financial industry. Such students are common across university degrees in cities like London.

[1]https://www.spec.org/benchmarks.html

[2]https://aws.amazon.com/cloudformation/

[3]https://www.oasis-open.org/committees/tosca/

Teaching challenges. A first issue with this topic is that if the content is presented using examples or lab exercises based on specific cloud providers, then slides may need yearly updates as autoscaling technologies and their interfaces evolve rapidly. A second issue is that specific elements of the theory are possibly too simple for a third- or fourth-year student (e.g. rule-based autoscaling), also presenting a low level of challenge in assessment. Lastly, a problem is that if the lecturer wishes to integrate elements from control theory upon teaching the topic, for example as in [5], these require theoretical baselines that may not be available to all computing students (e.g., Laplace and Z-transforms).

4.4 Serverless workflows

Serverless computing has been a prominent cloud computing trend for a number of years. Within it, Function-as-a-Service (FaaS) allows users to execute functions remotely in the cloud [4]. By enabling fine-grained autoscaling, FaaS offers a systematic advantage over traditional web services for enabling a scalable execution of scientific and business workflows in the cloud.

Serverless workflows may be used to teach various PE topics. Taking the user perspective, the module can introduce scheduling in the presence of workflows, resorting for example to the large body of literature available on deterministic scheduling [2], possibly in a lighter (albeit less rigorous) form than in algorithm theory courses devoted to the subject.

From the FaaS platform perspective, the PE module may touch upon heuristics, such as bin packing, that the platform may use to consolidate different serverless functions across server machines. The effects of memory constraints on performance are also an interesting topic aligned to serverless. Workflows are also useful to introduce the notion of a deadline, since they are often used to support critical end-of-month financial calculations in many businesses.
Teaching benefits. The topic has excellent appeal to students, who see its importance in the real world and the timeliness with respect to ongoing cloud computing trends. When coupled with scheduling theory concepts, the topic allows the lecturer to develop in the class a basic understanding of important general concepts such as NP-hardness, from which it is easy to pair the module with exercises involving metaheuristics (e.g., stochastic annealing, local search), which in my teaching experience generate a positive response in a computing class. It is also simple to define serverless scheduling problems in the coursework. My taught modules feature a workflow scheduling system for Azure Function that is supplied to the students. The exercises often involve studying some completion time minimization problems by developing a workflow schedule that is executed through this system. This generally has a good reception in the class as it combines both technological and theoretical elements.
Teaching challenges. One limitation of the serverless workflow topic is the limited number of optimal methods for general workflow scheduling [2]. Albeit several heuristics exist, this limits the appeal of the theory that can be developed in class. Moreover, real-world workflows can feature complex synchronization and be trigger-based and data-driven, which may introduce excessive complexity for modelling purposes. Lastly, workflow scheduling problems that are not solvable by brute force require tens of function calls. This may significantly extend the time required to solve coursework exercises by experimental means.

5. CONCLUSION

In conclusion, the advent of cloud computing offers an opportunity to refresh several contents of performance evaluation modules. In the early years of the performance community, there was general agreement on the importance of having a unified discipline centered around topic such as queueing theory. However, many more tools are available nowadays to study the performance of complex computing systems and performance evaluation teaching should evolve as a result. A review discussion has been presented on topics and techniques emerging from cloud engineering practice that may be added to the syllabus of a performance evaluation module, together with reflections on their educational merits and shortcomings.

6. REFERENCES

[1] A. Alnafessah, A. U. Gias, R. Wang, L. Zhu, G. Casale, and A. Filieri. Quality-aware devops research: Where do we stand? *IEEE access*, 9:44476–44489, 2021.

[2] K. R. Baker and D. Trietsch. *Principles of sequencing and scheduling*. John Wiley & Sons, 2013.

[3] P. J. Denning and J. P. Buzen. The operational analysis of queueing network models. *ACM Computing Surveys (CSUR)*, 10(3):225–261, 1978.

[4] S. Eismann, J. Scheuner, E. Van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup. A review of serverless use cases and their characteristics. *arXiv preprint arXiv:2008.11110*, 2020.

[5] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury. *Feedback control of computing systems*. John Wiley & Sons, 2004.

[6] N. R. Herbst, S. Kounev, and R. Reussner. Elasticity in cloud computing: What it is, and what it is not. In *10th international conference on autonomic computing (ICAC 13)*, pages 23–27, 2013.

[7] P. Jamshidi and G. Casale. An uncertainty-aware approach to optimal configuration of stream processing systems. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 39–48. IEEE, 2016.

[8] A. I. Khuri and J. A. Cornell. *Response surfaces: designs and analyses*. Routledge, 2018.

[9] S. Kounev, K.-D. Lange, and J. Von Kistowski. *Systems Benchmarking*. Springer, 2020.

[10] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., 1984.

[11] D. J. Lilja. *Measuring computer performance: a practitioner's guide*. Cambridge university press, 2005.

[12] T. L. Lo. The evolution of workload management in data processing industry: a survey. In *Proceedings of 1986 ACM Fall joint computer conference*, pages 768–777, 1986.

[13] B. Wilder. *Cloud architecture patterns: using microsoft azure*. " O'Reilly Media, Inc.", 2012.