# Theory and Practice in Performance Evaluation Courses: The Challenge of Online Teaching

Andrea Marin
marin@unive.it
Università Ca' Foscari Venezia
Venice, Venice, Italy

## ABSTRACT

This paper reports the experience gained over several years of teaching the course entitled *Software performance and scalability* at the University Ca' Foscari of Venice. The course is taken by perspective computer scientists and is taught at the master's level. It covers the topics of modeling and assessment of the performance properties of software systems.

In this paper, we will also devote attention to the challenge of online teaching due to the pandemic conditions.

Finally, we propose some auspices for the community to collect material for structured courses on performance and reliability evaluation topics.

## CCS CONCEPTS

• **Computing methodologies** → **Modeling and simulation**; • **Applied computing** → *Education*.

## KEYWORDS

Education, Performance Evaluation Course, Software Performance

**ACM Reference Format:**
Andrea Marin. 2023. Theory and Practice in Performance Evaluation Courses: The Challenge of Online Teaching. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3578245.3584353

## 1 INTRODUCTION

For the last decade, the Master's programme in Computer Science at the University Ca' Foscari of Venice, Italy, has offered a course entitled *Software Performance and Scalibility*. This course has replaced an excellent methodological course entitled *Performance Evaluation*. Why was this change considered necessary?

The answer of this question is based on the specific experience of this University. From students' point view, a methodological course on performance evaluation theory and methodology was mathematically very dense and, on the other hand, they have the misconception that the massive availability of resources at low price makes the problem of analyzing the quantitative aspects of computer systems (hardware of software) unnecessary. We can simplify this misconception with this question: *why should I learn a lot of Math when, if needed, I can buy more computational power, storage, bandwidth etc. for a very low price?*

Researchers in the area of quantitative analysis of computer systems have many answers to this question, and we are not going to recall them here, but they are not always easy to show in an educational context. Students hardly implement systems with performance issues during their Bachelor's or Master's programme.

This is often due to the fact that effort and the resources required to develop such systems are high, and hardly fit in a short study program that includes also many other subjects.

The group working on performance related topics at the University of Venice has then decided to take some measures: first, the application field has been restricted to software design and analysis. Second, the course has included a laboratory part that addresses the topic of benchmarking. Third, some methodological aspects have been removed from the topics (this has been probably the hardest part!) making a distinction between the students who will decide to continue their training at a PhD level (and hence will have to become 'specialists' in the field and will need deeper knowledge of the field), and those who aim to use the results produced by this area of research in their professional practice.

To make a possibly unfair comparison, the courses on Artificial Intelligence (AI) are often designed in this way. The training programs on AI can easily show applications of very efficient learning algorithms without going into the very details of their design leaving the mathematical and algorithmic insights to the students that decide to work in the area.

The measures succeeded in rising students' interest in performance modeling topics and facilitated the participation of companies to the educational process.

However, with the restrictions imposed by the pandemic, the lab part of the course has faced some issues. The physical space where the machines dedicated to the benchmarking part was not accessible, and we had to find ways to replace this crucial experience for the success of the course.

In this paper, we discuss this experience with a view of what remains, in our opinion, critical for the design of successful courses on performance evaluation topics.

The structure of the paper is as follows. In Section 2, we describe the organization of course. This description will be commented in Section 3 with some examples. Section 4 discusses the measures for overcoming the problems posed by the online teaching required by the Covid-19 restriction rules.


*ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal*

ACM ISBN 979-8-4007-0072-9/23/04…$15.00
https://doi.org/10.1145/3578245.3584353

## 2 COURSE DESCRIPTION

The course *Software Performance and Scalability* consists of 6 credits corresponding to 48 hours including lab and frontal lessons.

The overall effort required to the student is estimated in 150 hours.

The textbooks adopted are:

- Performance Modelling and Design of Computer Systems: Queueing Theory in Action by Mor Harchol-Balter [4];
- Systems Benchmarking: For Scientists and Engineers by Samuel Kounev and Klaus-Dieter Lange [7].

The course is thought to provide an introductory answer to three groups of questions:

(1) How do we schedule jobs? What is the impact on the system performance of our choice?
(2) What happens when our software system coordinates several components? How do we study the performance and scalability of these systems?
(3) How do we dispatch requests in a distributed systems? What is the impact on the system performance of our choice?

In brief, the topics addressed to answer these quesions are the following.

To answer the first question, we introduce queueing theory and show the results on M/M/1, M/M/m, M/M/$\infty$, M/G/1, M/G/1/PS, M/G/1/LCFPR, Erlang queues. We discuss the Shortest-Remaining-Processing-Time (SRPT) scheduling and the age-based scheduling with two level processor sharing.

The second question requires us to introduce queueing networks. We only address single class networks and classical bottleneck analysis. In this context, we show Jackson and Gordon-Newell's product-form results and the Mean Value Analysis.

The third and last question is introduced with the classical comparison between a shared queue with $m$ processors (M/M/$m$) and a random dispatcher to $m$ independent queues. We show the round-robin dispatcher and discuss its performance. Finally, we present join-the-shortest queue and join-the-idle-queue without their mathematical analysis.

To see how the lab is organized, we briefly describe a typical project. Students have been asked to implement a web app that had to compile a C++ source file sent by a form and return the results of the compilation in a sandbox. The goal is to propose a scheduling discipline to handle the pending requests with the idea of minimizing the expected response time. Then, they had to establish the scalability properties of their application and benchmark it to validate the results. The choice of the compiling task is due to the fact that it is easy to create source files with variable compiling times especially with a smart use of the C++ template system.

Regarding the tools used during the course, we adopt Java Modelling Tools[1] (JMT) [1] for the analysis of queueing networks and Tsung for the benchmarking experiments[2].

[1] https://jmt.sourceforge.net
[2] http://tsung.erlang-projects.org

## 3 OBSERVATIONS ON THE EXPERIENCE

### 3.1 Comments on the topics

In a recent work [2], De Nitto Personè describes the experiences of courses on performance modeling around the world. In this contribution, she emphasizes an interesting question about the role of University in preparing students not only to the economical and technological skills but also to a correct, scientific, approach to reasoning and problem solving.

We believe that this trade off is the crucial point that courses on performance evaluation topics have to address. The risk of resulting too abstract and with mathematical knowledge difficult to spend in the 'real-world' is concrete. This has been observed also in an interesting recent keynote speech [6]. But, on the other hand, a practical, problem-driven approach motivates the students.

We believe that the correct balance between these needs depends on many factors. First of all, our performance modeling courses are part of a much wider study program that differs from others in the mathematical and statistical skills developed by the students and for the complementary courses of the programs.

It should be clear that this reasoning does not intend to undermine the importance of analytical modeling which is pivotal in the work of practitioners of performance engineering as noted, e.g., in [10, 11], but it aims at make us think about the correct balance between a problem-driven approach and a methodological one.

*Example 3.1.* The decision of not addressing the problem of workload characterization in the course of *Software Performance and Scalibility* is due to the fact that there exists a parallel course on time series analysis where, among the examples, the teacher studies some traces of workload measurements.

*Example 3.2.* The mathematical skills of students are such that many proofs require them a lot of efforts to be understood. What are the contribution of these proofs in the preparation of the students? Can these skills be reached in other courses?

Consider, for example, the Pollaczek–Khinchine formula for the stationary probabilities of the M/G/1. These are given in terms of probability generating function of the stationary queue length distribution. The definition of this transform requires to express the distribution of the service time with its Laplace transform. Although there are several ways to prove the results, they all carry a huge complexity for the students of our Master's programme.

Conversely, if we limit our analysis to a mean value reasoning, we need to address the inspection paradox which is much more accessible.

Is the balance between efforts and rewards of proving the formula for the detailed state probabilities positive?

*Example 3.3.* There are some subjects that we love more than other in our field. These are difficult to reconsider. Personally, I have been working for many years on product-form theory [5] and I do really love this field. The traditional presentation of this subject begins with Burke's theorem and continues with Jackson and Gordon-Newell theorems. If there is time, one concludes with BCMP theorem. What is the problem of this approach? The proofs of these results based on the analysis of the global balance equations are mainly algebraic (with the exception of Burke's theorem that is more elegant). Any attempt to simplify them requires us

to introduce new notions that will be used essentially just to this aim (local balance, quasi-reversibility, Reversed Compound Agent Theorem (RCAT)). Is it worth to show the proofs of product-form results? We have decided to state the results without proofs.

However, some principles are necessary and we cannot ignore them. For example, we should be sure that all students understand that there is not a linear relation between the system load factor and most performance metrics like the expected response time. The lab experiences really help to translate the mathematical notions into skills. For example, students are usually really surprised to observe the reduction on the expected response time of their benchmarking experiments when they add a core to their computations by increasing the level of parallelism of their application.

### 3.2 Comments on the tools and model library

In our experience, the successful realization of a performance evaluation course greatly depends on the availability of user-friendly tools. JMT and Tsung are, in our experience, among the most usable tools. Similarly to the considerations proposed in the previous section, we do not want students to implement their own MVA algorithm but our aim is that they can use and interpret the results provided by JMT with awareness.

Unfortunately, we know that the development and maintenance of tools is extremely demanding in terms of resources. Moreover, many tools are mainly thought for academics or researchers. Students may have difficulty to use such tools during a course. Another important aspect is the documentation and portability of the tools.

However, in our experience, the comparison of the results of a tool with the measurements done by the benchmarking experience is extremely educational for the students.

A last difficulty is the availability of ´stable models' for some systems such as those presented in [8]. Some interesting research papers can be presented in a course as a case study (see, e.g., [3]) but the examples are not many. The models should be sufficiently simple to be understood by students of an introductory course and general to characterize a popular class of systems. It could be an interesting task to collect the case studies presented in the courses of performance evaluation to form a shared library.

### 3.3 Comments on the lab experience

The implementation of a lab part of the course has been extremely important both to increase the interest of the students in the subject and to allow them to understand some principles with a hand-on experience.

In our course, we set up a lab with dismissed machines from our datacenter and we created a LAN isolated from the rest of the department. In this way, students can deploy their applications and make tests without worrying about traffic interference, security etc. They can even work with the BIOS, e.g., to enable/disable hyperthreading.

*Example 3.4.* A successful example of the importance of the lab is how the groups have tried to manage the queue of pending programs to compile. They soon discovered that a FCFS policy was far from being optimal. The group that provided the solution with the fastest scheduling has tried to implement a SRTP with a guess on the compiling time based on previous statistics of similar source files previously compiled. The idea was justified by the observation that in a real online service of this type users would probably compile a source file many times after a few lines of code added or modified. Hence, the compiling time should not change much in the majority of the cases.

The lab part is very important in the development of the students' skills, but it requires a lot of time. In fact, not all the topics recommended by Smith in [9] are covered by this course and probably at least 9 credits would be necessary to maintain the lab part. However, we noticed that our students have a hard time to understand theory of performance without an appropriate observation of the main principles on real systems.

## 4 THE EXPERIENCE OF ONLINE TEACHING

During the first lockdown of 2020, the course of Software Performance and Scalability had just begun. The following two editions faced restrictions on the amount of students that could fit in a room and hence the lab was not accessible.

We used the datacenter of the University to reserve some virtual machines and changed the project into a task of assessing the performance of a certain database. We paid attention to avoid shared resources and to allow students to use testing machines physically close to the system under test. However, we noticed that there have been more difficulties in carrying out the tasks with respect to the physical labs. This was due to several factors, such as the difficulty in synchronizing the presence of the teacher/tutor with all groups. Moreover, students seem to have an easier time to work with physical machines rather than in the cloud.

Beside these problems, we realized that the possibility of conducting experiments of benchmarking in the cloud can be extremely interesting if well prepared. It could be interesting to have an application shared for all groups and just focus on its performance modeling and evaluation.

We think that it could be interesting o have such an environment shared among the communities teaching performance modeling and analysis of software architectures. In a certain sense, this could mean to keep the best part of the negative experience of the online teaching during the pandemic.

## 5 CONCLUSIONS

We reported some considerations on the experience of teaching the course of *Software Performance and Scalability* at the University Ca' Foscari of Venice. We discussed the balance between a model-driven and problem-driven approach to the didactic of the course. We share with the view of [9] the necessity of having, as community, a set of real-world case studies to present in the courses. Possibly, these should teach generally valid lessons and use standard approaches rather than ad-hoc ones. Moreover, we believe that this wish could be extended also to a lab environment where students can develop practical skills. A successful case study, in a different domain, is Kathará[3] where labs of networking can be done. With such a tool, labs of benchmarking could be done on physical labs or on the cloud with a nice variety of scenarios.

[3] https://www.kathara.org

# REFERENCES


[1] Marco Bertoli, Giuliano Casale, and Giuseppe Serazzi. 2009. JMT: performance engineering tools for system modeling. *ACM SIGMETRICS Performance Evaluation Review* 36, 4 (2009), 10–15.

[2] Vittoria de Nitto Persone. 2020. Teaching Performance Modeling in the era of millennials. *CoRR* abs/2001.08949 (2020). https://arxiv.org/abs/2001.08949

[3] Abhishek Dubey, Rajat Mehrotra, Sherif Abdelwahed, and Asser N. Tantawi. 2009. Performance modeling of distributed multi-tier enterprise systems. *SIGMETRICS Perform. Evaluation Rev.* 37, 2 (2009), 9–11.

[4] Mor Harcol-Balter. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action.* Cambridge University Press.

[5] Peter G. Harrison and Andrea Marin. 2014. Product-Forms in Multi-Way Synchronizations. *Comput. J.* 57, 11 (2014), 1693–1710.

[6] Boudewijn R. Haverkort. 2021. Performance Evaluation: Model-Driven or Problem-Driven?. In *Quantitative Evaluation of Systems - 18th International Conference, QEST 2021, Paris, France, August 23-27, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12846)*. Springer, 3–11.

[7] Samuel Kounev and Klaus-Dieter Lange. 2021. *Systems Benchmarking: For Scientists and Engineers.* Springer.

[8] Daniel A. Menascè and Virgilio A. F. Almeida. 2005. *Capacity planning for web performance: metrics, models and methods.* Prentice Hall Iberia.

[9] Connie U. Smith. 2021. Software Performance Engineering Education: What Topics Should be Covered?. In *ICPE '21: ACM/SPEC International Conference on Performance Engineering, Companion Volume, WEPPE Workshop*. 131–132.

[10] Y. C. Tay. 2019. Lessons from Teaching Analytical Performance Modeling. In *ICPE '19: ACM/SPEC International Conference on Performance Engineering, Companion Volume, WEPPE Workshop*. 79–84.

[11] Y. C. Tay. 2021. The Role of Analytical Models in the Engineering and Science of Computer Systems. In *ICPE '21: ACM/SPEC International Conference on Performance Engineering, Companion Volume, WEPPE Workshop*. 107.