How can we teach workload modeling in CS systems classes?

@ TeaPACS 2024

Cristina L. Abad

Escuela Superior Politécnica del Litoral (ESPOL)

About me

Cristina L. Abad

Associate Professor at ESPOL (EC)

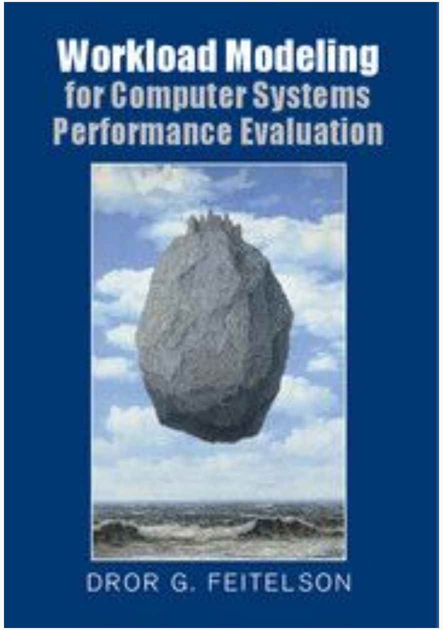
- Deputy Dean of Research
- Lead the Distributed Systems Lab

- PhD in CS, University of Illinois Urbana-Champaign
  - Thesis: Big Data Storage **Workload Characterization, Modeling and Synthetic Generation**
- Member of the Steering Committee of the SPEC RG
  - Elected Secretary
- Have taught
  - Distributed Systems
  - Advanced Operating Systems
  - Cloud Computing
  - Big Data Architectures
  - Operating Systems
  - Computer Networks
  - Computer Communications
- Research at intersection of PE & DS + CS Education
  - Learning through creating learning objects: Experiences with a class project in a distributed systems course @ ITiCSE 2008
  - Have We Reached Consensus? An Analysis of Distributed Systems Syllabi @ SICGSE 2021
  - An Analysis of Distributed Systems Syllabi With a Focus on Performance-Related Topics @ WEPPE 2021

Teaching workload modeling is important

Importance of workload modeling

- Incorrect assumptions about workload → Sub-par designs and evaluations
  - Garbage in → Garbage out
- Workloads <u>cannot</u> be an afterthought
- Understanding workloads can help us reason about results
- Not trivial to do:
  - Which features of the workload should be modeled?
  - How do we model the features? Independently?
  - Should we use the full real workload (trace) instead?



Feitelson, Dror G. Workload modeling for computer systems performance evaluation. Cambridge University Press, 2015.

Is this importance reflected in curriculum guidelines?

What do curriculum guidelines say? (ACM, 2020)

2020 guidelines:

- Moved away from knowledge area, knowledge unit, learning outcome mindset of the 2013 and prior guidelines to **competency-based learning**
- Result: Guidelines <u>removed</u> (1) mention of workloads
  - And, queueing theory (and prob, stoch)
- Kept (a few) performance and simulation references
  - These mentions are at the same time too general and too specific to be useful for educators (Examples in next slide)

From ACM 2020 (2021)

https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf

C.2.2: Computer Science Draft Competencies

- NC-Networking and Communication
  - Design and implement a simple reliable protocol for an industry network by considering factors that affect the **network's performance**.
- PD-Parallel and Distributed Computing
  - Implement a parallel divide-and-conquer (and/or graph algorithm) for a client by mapping and reducing operations for the real industry problem and **empirically measure its performance relative to its sequential analog**.
- SF-Systems Fundamentals
  - Design a simple parallel program for a corporation that manages shared resources through synchronization primitives and **use tools to evaluate program performance**.
  - **Design and conduct a performance-oriented, pattern recognition experiment** incorporating state machine descriptors and simple schedule algorithms for exploiting redundant information and data correction that is usable for a local engineering company and use appropriate tools to measure program performance.
  - Calculate average memory access time and **describe the tradeoffs in memory hierarchy performance** in terms of capacity, miss/hit rate, and access time *for a local engineering company*.
  - **Measure the performance of two application instances running on separate virtual machines** at a local engineering company and determine the effect of performance isolation.
- CN-Computational Science
  - Create a simple, formal mathematical model of a real-world situation and use that model in a **simulation** for a local technology company.

What do curriculum guidelines say? (ACM, 2023 Beta)

2023 appears to be bringing these concepts back, giving them a higher importance

- They brought back the same wording from 2013; made them core knowledge
- Even added explicit pre-requisites that mention queueing theory and stochastic processes / probability

SF/Performance Evaluation
[2 CS Core hours and 2 KA Core hours]
Topics:

- Performance figures of merit
- Workloads and representative benchmarks, and methods of collecting and analyzing performance figures of merit
- CPI (Cycles per Instruction) equation as tool for understanding tradeoffs in the design of instruction sets, processor pipelines, and memory system organizations.
- Amdahl's Law: the part of the computation that cannot be sped up limits the effect of the parts that can
- Analytical tools to guide quantitative evaluation
- Order of magnitude analysis (Big O notation)
- Analysis of slow and fast paths of a system
- Events on their effect on performance (e.g., instruction stalls, cache misses, page faults)
- Understanding layered systems, workloads, and platforms, their implications for performance, and the challenges they represent for evaluation
- Microbenchmarking pitfalls

Relevant changes from 2013:

- Added a new unit of system performance, which includes the topics from the deprecated unit of proximity and the deprecated unit of virtualization and isolation;
- Added a new unit of performance evaluation, which includes the topics from the deprecated unit of evaluation and the deprecated unit of quantitative evaluation;

Other curriculum initiatives?
NSF/IEEE-TCPP Curriculum Initiative on PDC

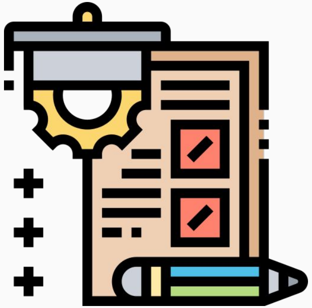
NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing - Core Topics for Undergraduates Version 2.0

- Architecture topics → Performance metrics

Memory bandwidth	Be able to explain significance of memory bandwidth with respect to multicore access, and different contending workloads, and challenge of measuring	Arch2
Network bandwidth	Know how network bandwidth is specified and explain the limitations of the metric for predicting performance, given different workloads that communicate with different contending patterns	Arch2

Back to the question: Is this importance (of WM) reflected in curriculum guidelines?

- No explicit mentions of workload modeling
- Few mentions workloads about using them in evaluations
  - BUT, in some cases, only within some topics/areas but not others
- Competency/skills-based guidelines approach makes it harder to explicitly include importance of workloads in performance



Where (in the CS curriculum) can we teach workload modeling?

Where (in the CS curriculum) can we teach workload modeling?

- Standalone courses (?) → Too specialized
- Applied statistics courses → Unlikely, as other applications are more commonly taught
- Software eng. courses → Unlikely, unless course focuses on testing
- Simulation courses
- Performance courses → Briefly discussed in this talk
- Monitoring courses → Briefly discussed in this talk
- Systems courses ← This talk!

Where (in the CS curriculum) can we teach workload modeling?

- Standalone courses (?) → Too specialized
- Applied statistics courses → Unlikely, as other applications are more commonly taught
- Software eng. courses → Unlikely, unless course focuses on testing
- Simulation courses
- Performance courses → Briefly discussed in this talk
- Monitoring courses → Briefly discussed in this talk
- Systems courses ← This talk!

Where (in the CS curriculum) can we teach workload modeling? Performance courses

- Performance evaluation (and benchmarking)
  - Workloads are essential; no other way of doing this
- Performance modeling
  - Depends on preferences of instructor

Where (in the CS curriculum) can we teach workload modeling? Performance courses

- Performance evaluation (and benchmarking)
  - Workloads are essential; no other way of doing this
- Performance modeling
  - Depends on preferences of instructor

Introduction:			
1	16/2/14	Motivation and problems The issues: techniques, metrics, and workloads Using measurements, simulations, and analysis Jain chap. 2, 3	ex0: looking at graphs
Queueing analysis:			
Workloads:			
6	23/3/14	Workload analysis and characterization Summary statistics such as mean and median Feitelson, Workload modeling for performance evaluation. Performance 2002 tutorials (or the long version)	
	25/3/14	Creating a variate from a distribution Jain chap. 12, 28; Law/Kelton chap. 6, 8	ex5: generating random variates
7	30/3/14	Useful distributions Parameter estimation techniques and goodness of fit Comparing distributions using quantile-quantile plots Law/Kelton chap. 8; Jain chap. 29	
	1/4/14	Heavy tails and long tails Power laws and the Pareto distribution Mass-count disparity and conditional expectation Popularity and the Zipf distribution Crovella, Performance evaluation with heavy tailed distributions. JSSPP 2001	ex6: fitting a distribution with Q-Q plots
8	29/4/14	Case study: load balancing Oblivious balancing Balancing based on workload characteristics Harchol-Balter and Downey, Exploiting process lifetime distributions for dynamic load balancing. *ACM Trans. Comput. Syst.* **15(3)** pp. 253-285, Aug 1997	ex7: characterizing requests from a web server
9	11/5/14	Feedback in workloads (slides) The daily cycle of activity User-based workload modeling Shmueli and Feitelson, Using site-level modeling to evaluate the performance of parallel system schedulers. 14th MASCOTS, Sep 2006	project: extracting feedback data
	13/5/14	Self similarity The Hurst parameter	
Simulation:			

Performance Evaluation Example: Topics in Performance Evaluation (Prof. Dror Feitelson)

Example 2: Performance Evaluation and QoS (Prof. Maria Carla Calzarossa)

1. Introduction
2. Measures
3. Workloads →
4. Metrics and fundamental laws
5. Service centers; queue networks
6. Other performance topics
7. Simulation

Definitions. Load types, levels of detail, parameters. Quantitative and qualitative parameters, measured and derived. Methodological approach. Exploratory analysis: basic statistics, frequency distributions, percentiles, scatter plots and correlations. Web server log analysis. Static and dynamic properties of the workload. Parameter scaling. Statistical techniques: clustering. Hierarchical agglomerative algorithms: dendrogram. Hierarchical algorithms: k-means. Principal component analysis. Correspondence analysis. Linear and non-linear regression methods. Examples of studies requiring detailed knowledge of workload intensity.

Where (in the CS curriculum) can we teach workload modeling? Performance courses

- Performance evaluation (and benchmarking)
  - Workloads are essential; no other way of doing this
- Performance modeling
  - Depends on preferences of instructor

Performance Modeling: Different course flavours

Groups of PM courses [Personè 2020]:

- General PM
- PM for Communications Systems
- PM for Software

Gen PM	*PM for CS*	*PM for SW*
Operational Laws, Queueing Systems, Statistics		
Fluid models	Combinatorics	LQN
Optimization	Control	Petri nets
Petri nets	Fluid models	Simulation
Probability	Game Theory	Workload charact
Process algebras	Graph Theory	
Simulation	Net Calculus	
Stochastic processes	Optimization	
Timed automata	Probability	
Workload charact	Simulation	
	Stochastic processes	

Personè, Vittoria de Nitto. "Teaching Performance Modeling in the era of millennials." arXiv preprint arXiv:2001.08949 (2020).

PM Example 1: Analytical PM (Prof. Y.C. Tay)

- Use real systems papers to illustrate usefulness of different analytical models (incl. in relation to workloads)
  - E.g., "GPUs and disks, routers and crawling, databases and multimedia, worms and wireless, multicore and cloud, security and energy, etc."

2.6 Bottleneck analysis

No matter how complex a system is, estimating its performance is usually easy at two extremes: when workload is light and no time is wasted on queueing for resources, and when workload is heavy and performance is determined by a bottleneck resource that is rarely idle. These extremes determine two straightline asymptotes that meet at a *knee*, and performance is a nonlinear curve around this knee, but converges to these asymptotes for light and heavy workloads. Sometimes, it suffices that the model locate these two straight lines via a bottleneck analysis.

Y.C. Tay. 2019. Lessons from Teaching Analytical Performance Modeling. In Companion of the 2019 ACM/SPEC International Conference on Performance Engineering (ICPE '19). Association for Computing Machinery, New York, NY, USA, 79–84

PM Example 2: Perf. Engineering (Dr. Giuliano Casale)

- Course designed at the **intersection between performance engineering and systems engineering**, with a focus on cloud computing systems
  - "[...] focused on the intersections between PE with **cloud computing**, but taking primarily a **systems engineering** and measurement view"
- Sample topics:
  - Markov chains to describe user workload patterns → **workload modeling**
  - Labs with where students instantiating and size VMs and conducting benchmarking and **workload characterization** experiments

Giuliano Casale. 2023. Performance evaluation teaching in the age of cloud computing. SIGMETRICS Perform. Eval. Rev. 51, 2 (2023), 45–49.

Where (in the CS curriculum) can we teach workload modeling?

- Standalone courses (?) → Too specialized
- Applied statistics courses (or DS) → Unlikely, as other applications are more commonly taught
- Software eng. courses → Unlikely, unless course focuses on testing
- Simulation courses
- Performance courses → Briefly discussed in this talk
- Monitoring courses → Briefly discussed in this talk
- Systems courses ← This talk!

Where (in the CS curriculum) can we teach workload modeling? Monitoring courses

- “Monitoring is at the basis of workload characterization” [Calzarossa 2021]
  - When monitoring, the system must be under an artificial workload (active monitoring) or real workload (passive monitoring)
- PRO: DevOps is a hot topic/skill for students
- Arguably, monitoring courses can be considered systems courses
  - “[...] monitoring makes it possible to see in action concepts taught in classes (e.g., Internet protocol stack, software systems), thus bridging the gap between theory and practice.”
  - Or, software engineering
  - Or, performance evaluation
  - Different versions of these courses exist, depending on expertise of instructor

Maria Carla Calzarossa, Luisa Massari, and Daniele Tessera. 2021. Performance Monitoring Guidelines. In Companion of the ACM/SPEC International Conference on Performance Engineering (ICPE '21). Association for Computing Machinery, New York, NY, USA, 109–114.

Where (in the CS curriculum) can we teach workload modeling?

- Standalone courses (?) → Too specialized
- Applied statistics courses (or DS) → Unlikely, as other applications are more commonly taught
- Software eng. courses → Unlikely, unless course focuses on testing
- Simulation courses
- Performance courses
- Monitoring courses
- Systems courses

(Simulation, performance and monitoring courses:) These are usually elective courses, chosen by few students :-(

Where (in the CS curriculum) can we teach workload modeling? Systems courses

From the ACM CS Curriculum Guidelines (2023) → Systems Fundamentals KA

> In the curriculum of computer science, the study of computer systems typically spans across multiple courses, including, but not limited to, operating systems, parallel and distributed systems, communications networks, computer architecture and organization and software engineering. The System Fundamentals knowledge area, as suggested by its name, focuses on the fundamental concepts in computer systems that are shared by these courses within their respective cores. The goal of this knowledge area is to present an integrative view of these fundamental concepts in a unified albeit simplified fashion, providing a common foundation for the different specialized mechanisms and policies appropriate to the particular domain area. These concepts include an overview of computer systems, basic concepts such as state and state transition, resource allocation and scheduling, and so on.

This is the only area in the guidelines where **workloads** are mentioned

Where (in the CS curriculum) can we teach workload modeling? Systems courses

- Only way to make sure students are exposed to the concept of workload modeling (other courses are elective)
- Enables departments to teach performance even if they have stopped teaching PM courses
- Fits well with existing curriculum guidelines
- Fits with how perf. practitioners (as a community) have evolved [Casale 2023]
- Can help **motivate** students to take courses focused on PM

Where (in the CS curriculum) can we teach workload modeling? Systems courses

"To facilitate the dissemination of performance evaluation concepts thus increasing the number of students interested in this discipline, another action can be taken at the organizational level. It consists of the integration of performance evaluation concepts, with simple examples, in some popular computer engineering courses. Very few lessons are needed and an application-oriented approach should be adopted." [Serazzi 2022]

Serazzi also argues for a *learning through applications* approach .

Giuseppe Serazzi. 2022. Updating the Content of Performance Analysis Textbooks. SIGMETRICS Perform. Eval. Rev. 49, 4 (March 2022), 24–27.

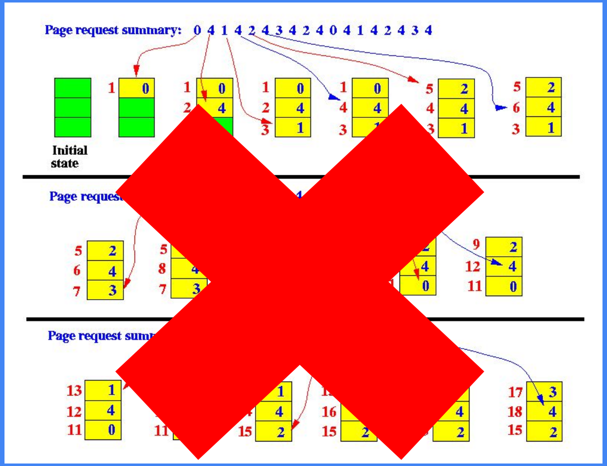
How can we teach workload modeling in CS systems classes?

How can we teach workload modeling in CS systems classes?

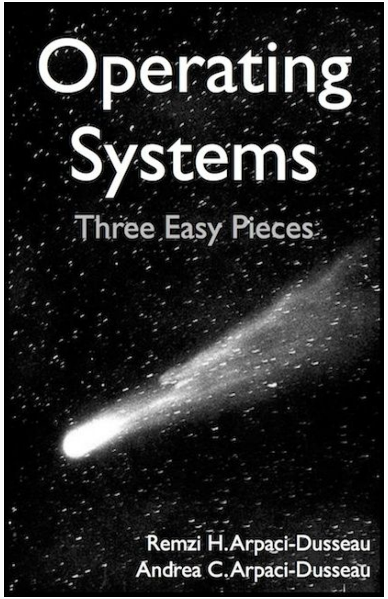
- Pen-and-paper exercises (e.g., classical caching eviction problems)
  - Hard to include modeling; clumsy; only for very simple WMs
- Make workloads a **first-class citizen**
  - Add a unit or subsection on workload modeling
  - Discuss thoroughly when applicable
  - Include code to test systems using different workloads
- Include experimentation with diverse workloads in one or more **labs**
- **Homework** where students **simulate** a networked system
- A requirement of the **final project**
- Research **paper-based analysis** (e.g., for advanced, seminar-style classes)
- Eval. of system for a client (competency-based curriculum guidelines)
  - What if their workloads are not interesting?

Make workloads a first-class citizen

An Operating Systems example



- Workload assumptions in first subsection of some chapters!
- Workloads discussed in examples
- Questions about how workloads affect results
- Additional code provided
  - Students can run simulations w/ different workloads
  - More advanced workload modeling left to suggested readings



22.6 Workload Examples

Let's look at a few more examples in order to better understand how some of these policies behave. Here, we'll examine more complex **workloads** instead of small traces. However, even these workloads are greatly simplified; a better study would include application traces.

Our first workload has no locality, which means that each reference is to a random page within the set of accessed pages. In this simple example, the workload accesses 100 unique pages over time, choosing the next page to refer to at random; overall, 10,000 pages are accessed. In the experiment, we vary the cache size from very small (1 page) to enough to hold all the unique pages (100 pages), in order to see how each policy behaves over the range of cache sizes.

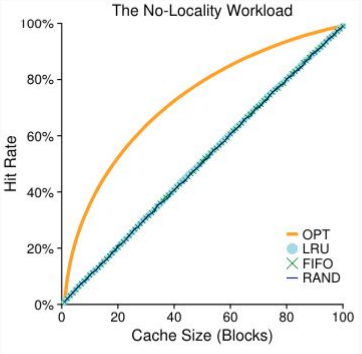


Figure 22.6: **The No-Locality Workload**

Experimentation in guided labs

Can include workload characterization or modeling components

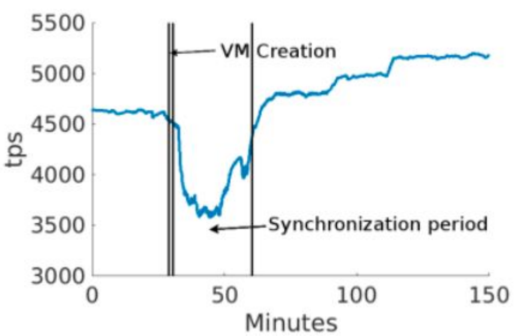
E.g., in a Distributed Systems or Cloud Computing class

Example: predictive autoscaling

I ask my students:

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

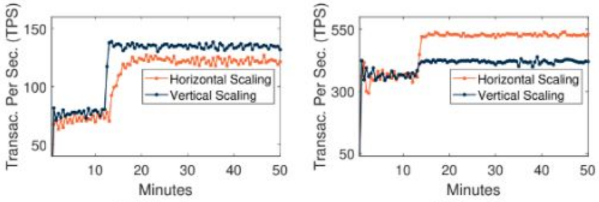
Cassandra autoscaling: with cost of data synch.



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. Users	Think Time	CPU Share	Replica
	Home	Catalog	Carts				
A	57%	29%	14%	1000	7 sec	0.2	1
B				4000		1.0	



(a) Case A (b) Case B

Giuliano Casale. 2023. Performance evaluation teaching in the age of cloud computing. SIGMETRICS Perform. Eval. Rev. 51, 2 (2023), 45–49.

Simulation homework/project

Workload **models** are a natural way of injecting load

Useful for a Networking, or DS class

Ref: Computational Forensics and Investigative Intelligence, CMU,

15-498 Project #2
Simulating a Distributed System

- **Computer Networks: A Systems Approach**, fourth edition, by Larry Peterson and Bruce Davie.
- **Operating Systems Concepts**, seventh edition, by Silberschatz, Galvin and Gagne
- **Distributed Systems: Principles and Paradigms** by Tanenbaum and van Steen
- **Distributed Systems: Concepts and Design**, fourth edition, by Coulouris, et al

The Workloads

Although your workloads can, technically, be hand-crafted – it is a much better idea to generate them automatically using some sort of statistical distribution as a guide, for example, an exponential or normal distribution. Using these as the "basics", you can then add failures or other features. Doing this will enable you to test many, many possibilities with far less work.

There are four basic things to consider in developing your workloads:

1) When should each processor request access to the critical section?
2) How long should each access to the critical section last?
3) When should failures occur?
4) How long should each failure last?

You are welcome to answer these statically by creating long lists of activities by hand. But, for reasons of your own sanity, we don't recommend this. Instead, we recommend that you use statistical models to describe these things. For example, you might want to assume that the frequency of each processor's request is governed by a uniform distribution – basically sprinkling an equal number of processors along the distribution from "never use critical section" to "sometimes use critical section" to "constantly use critical section", and everything thing along the way. Or, instead, you might want these to follow a linear distribution or an exponential distribution. The same goes for failure. You probably want to pick a model for the occurrence of and duration of failures, and then generate failure events accordingly.

Tweaking Workloads

In addition to workloads based on pure distributions, you might want to tweak some workloads to test certain edge cases, for example the "tie situation" in the majority voting scheme. This is especially important if you suspect that certain cases might have interesting results – and they don't happen to come up in your distribution-generated experiments.

You might also want to try to determine how often these interesting cases occur within for different statistical distributions at different levels of contention.

A requirement of a final project

Example, Distributed Systems class:

- Implement a key-value store
- Evaluate with configurable workloads and real traces
- **Extra credit** for most performant implementation!
- Other benefit: Students learn to integrate their solution with existing benchmarking tools

Similar experience at OS class, implementing different page eviction policies

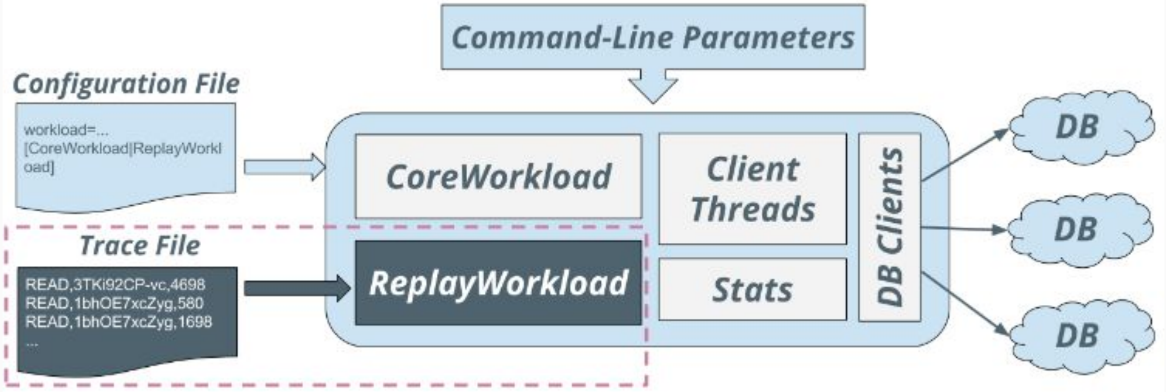


Figure 1. KV-replay's architecture, as an extension to YCSB.

E. F. Boza, C. San-Lucas, C. L. Abad and J. A. Viteri, "Benchmarking Key-Value Stores via Trace Replay," IEEE International Conference on Cloud Engineering (IC2E), 2017

Research paper-based analysis

For advanced courses, seminars

Example from Advanced Operating Systems class (ESPOL)

- Weekly paper discussions
- Most sessions include explicit discussions about workloads used in assigned reading
  - Where they adequate and thorough?
  - Any interesting characterization presented?
  - Any models used?
  - Limitations of models used?
- Final project is research-based and evaluations should be done with adequate workloads (models or traces)

Which systems class?

Ideally: All/most of them!

"To facilitate the dissemination of performance evaluation concepts thus increasing the number of students interested in this discipline, another action can be taken at the organizational level. It consists of the **integration of performance evaluation concepts**, with simple examples, in some **popular computer engineering courses**. Very few lessons are needed and an application-oriented approach should be adopted." [Serazzi 2022]

- At the very least, do for Distributed Systems
  - Performance tradeoffs are thoroughly discussed in class

Table 1: Topics, in the topic lists of the surveyed syllabi, in which the string *perf* appears.

Topic
Reasoning about system performance
Isolation and consistency semantics: Performance/usability trade-offs
Performance at scale
Performance: eRPC
Scalability vs. fault-tolerance vs. performance
No compromises: Distributed transactions with consistency, availability, and performance (paper)
NFS: Performance optimisations

Table 2: Topics, in the topic lists of the surveyed syllabi, in which *scal[e/able/ability]* or *elast* appear.

Topic
Don't settle for eventual: Scalable causal consistency for wide-area storage with COPS (paper)
Scale-out key-value storage, Dynamo
Case studies from industry: Google's Chubby fault-tolerant lock service, Google's Spanner scalable, fault-tolerant ACID database
Large-scale data processing with MapReduce
Performance at scale
Large-scale data stores
Load balancing: LARD, Internet-scale services
Scalability issues and the concept of gossip
Scalable services, reliability, and consistency: Scale and recovery for storage, leases, linearizable RPC for a replicated storage service
Quality attributes (availability/reliability, modifiability, scalability)
Scalability vs. fault-tolerance vs. performance
Scalability of blockchains
Elastic services in the cloud: Managed services, mega-services and auto-scaling, request routing and load balancing: into the network, auto-sharding and sharded request routing

  - Students are more mature; have many prior knowledge that can be leveraged
- Cloud computing classes are also a natural fit

Cristina L. Abad, Alexandru Iosup, Edwin F. Boza, and Eduardo Ortiz Holguin. 2021. An Analysis of Distributed Systems Syllabi With a Focus on Performance-Related Topics. In Companion of the ACM/SPEC International Conference on Performance Engineering (ICPE '21).

For students

Garbage-in → Garbage-out

Understanding workload key for design and evaluation

Sometimes the best research contributions come from understanding how different workloads stress the system

For teachers

Workloads typically students' last concern →Give explicit credit for WM

Show examples with unexpected results coming from diverse workloads

Competition can spark interest

How can we teach workload modeling in CS systems classes?