An Analysis of Distributed Systems Syllabi With a Focus on Performance-Related Topics

Cristina L. Abad
cabadr@espol.edu.ec
Escuela Superior Politecnica del Litoral
Guayaquil, Ecuador

Alexandru Iosup
a.iosup@vu.nl
VU Amsterdam
Amsterdam, the Netherlands

Edwin F. Boza
eboza@espol.edu.ec
Escuela Superior Politecnica del Litoral
Guayaquil, Ecuador

Eduardo Ortiz Holguin
leortiz@espol.edu.ec
Escuela Superior Politecnica del Litoral
Guayaquil, Ecuador

ABSTRACT

We analyze a dataset of 51 current (2019-2020) Distributed Systems syllabi from top Computer Science programs, focusing on finding the prevalence and context in which topics related to performance are being taught in these courses. We also study the scale of the infrastructure mentioned in DS courses, from small client-server systems to cloud-scale, peer-to-peer, global-scale systems. We make eight main findings, covering goals such as performance, and scalability and its variant elasticity; activities such as performance benchmarking and monitoring; eight selected performance-enhancing techniques (replication, caching, sharding, load balancing, scheduling, streaming, migrating, and offloading); and control issues such as trade-offs that include performance and performance variability.

ACM Reference Format:
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 2021 ACM/SPEC International Conference on Performance Engineering (ICPE '21 Companion), April 19–23, 2021, Virtual Event, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3447545.3451197

1 INTRODUCTION

At the start of the roaring (twenty-)twenties, higher education faces the important challenge of scaling to unprecedented numbers of students in Computer Science (CS) and related fields. But scaling is not the only important goal; equally, higher education must focus on how well the students acquire knowledge, skills, and experience relevant to the problems they will face in their professional life. Performance is a relevant topic of general importance but with specific challenges in CS. Though the WEPPE workshop has published various studies on the design and experience in teaching a standalone performance course [2, 5, 24], a declining number of curricula include even a single course focusing entirely on performance [5]. Instead, performance aspects can appear in several courses in the curriculum, and especially in computer systems (e.g., [7–9]).[1] The scattered approach hampers identifying common topics and sharing best teaching practices related to them. Addressing this problem, this work identifies and analyzes performance-related topics in a popular class of courses on Distributed Systems (DS).

The choice of focus on DS courses is not arbitrary. First, DS courses appear often in curricula sanctioned by the IEEE/ACM guidelines [13]. Other popular IEEE/ACM curricula, on Computer Engineering, Information Technology, and Software Engineering, also include important DS topics [14–16].

Second, among the courses in which performance aspects can be discussed, Distributed Systems is a good target. DS courses have a focus on the system view of the world, which is holistic by nature. Distributed systems present naturally a diverse set of levels and components where performance can be considered, e.g., from multi-node pools of execution threads to massive global-scale deployments, from processors to storage and to diverse sensors. Last, distributed systems have a broad range of performance-aware applications, e.g., online gaming, video streaming, public transportation and high-performance scientific processing, so performance aspects can be explained with prominent and practical examples.

We analyze 51 current Distributed Systems syllabi from top CS programs around the world and study whether or not they discuss issues related to performance in them. Specifically, we seek to answer two main questions: *(1) To what extent are performance-related topics being listed in the topic lists of these syllabi?* and *(2) Do these syllabi include papers that have a strong focus on performance?*

Our work complements prior research in Computer Science education that has looked into performance engineering or teaching Distributed Systems (Section 2). Closest to our work, is our previous study of these DS syllabi [1], where we sought to answer four questions: What are the most frequent DS course names?, Which topics are commonly included in the syllabi?, Which books and papers appear in the reading lists?, and, Do DS courses have a strong theoretical focus? In contrast, this article focuses on the *performance* aspects of DS courses.

Our main contributions in addressing the main questions are:

(1) We define a set of research questions to analyze the presence of performance aspects in DS courses (Section 3). We cover common goals in performance engineering, such as improving performance and scalability, common activities, such as benchmarking and monitoring, and common techniques,

[1]Furthermore, the Body of Knowledge of the IEEE/ACM 2013 CS Curriculum Guidelines [13] lists topics and learning outcomes related to performance throughout several of its knowledge areas: Algorithms and Complexity (AL), Architecture and Organization (AR), Computational Science (CN), Information Assurance and Security (IAS), Information Management (IM), Intelligent Systems (NC), Operating Systems (OS), Platform-Based Development (PBD), **Parallel and Distributed Computing (PD)**, Software Engineering (SE), Systems Fundamentals (SF). (The emphasis on PD is ours.)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICPE '21 Companion, April 19–23, 2021, Virtual Event, France
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8331-8/21/04…$15.00
https://doi.org/10.1145/3447545.3451197

such as replication and scheduling. We also look at the scales of systems covered in DS courses, from small (e.g., cluster) to massive (e.g., datacenter) to global (e.g., Internet-scale).

(2) We present the main results of our data-driven analysis (Section 4). We summarize a set of main findings, e.g., that, although DS courses cover often performance issues, performance itself is not a topic commonly covered by these courses. For each question, we quantify the presence of specific performance issues among the course topics, and among the papers proposed as course reading material.

(3) We make a set of recommendations for the community teaching DS, aiming to improve the presence and diversity of performance topics (Section 5). For example, we recommend that performance should be an explicit topic in DS courses.

2 RELATED WORK

Prior historical and recent work on performance education within the CS curricula has looked into the design and experience in teaching courses that focus on performance, like Performance Evaluation [23], Analytical Performance Modeling [24], and Performance Analysis of Computer Systems and Networks [2]. For a comprehensive look at Performance Modeling (PM) courses, we refer the reader to the work of de Nitto Personè, who looked into 75 courses on PM in the world and highlighted insights about their approach in a talk at WEPPE 2017 [5] and a later technical report [6].

Issues about performance could also be added across the curricula to courses like Parallel Programming [9], a third-year programming course [7], and Computer Organization and Design [8]. This paper complements these works by looking into the current practice of teaching Distributed Systems, identifying which issues related to performance are more commonly being taught.

The current work complements our prior study analyzing common patters in DS syllabi; we refer the reader to [1] for a summary of the related work on education practices in teaching DS.

3 METHOD FOR ANALYZING PERFORMANCE ASPECTS IN DS COURSES

We propose a method for analyzing performance aspects in Distributed Systems (DS) courses. Our method is data-driven: starting from a previously collected dataset, we ask specific research questions. For each question, we describe how to obtain its answer; this part of the method leverages the domain knowledge and expertise of the first two authors in teaching *both* performance topics and distributed systems courses, a combined expertise of teaching such courses during over 30 academic years.

We analyze a dataset of DS syllabi that we manually collected and curated.[2] The dataset contains 51 syllabi from institutions in the top-100 of the 2019 Times Higher Education World University Rankings for Computer Science (CS).[3] To ensure that these syllabi are representative of current teaching practices, we selected syllabi from course offerings in 2019 and 2020. When universities had more than one Distributed Systems course (e.g., Distributed Systems and Advanced Distributed Systems), we chose the most basic course (lower level); the reason for this is that the dataset was collected specifically with the intention of analyzing entry-level DS courses [1] and it is our view that performance issues should be part from the start in thinking about DS topics. We manually curated the syllabi and built a table with thirteen fields: Rank, University, Country, Course name, Instructor, Course code, Semester/year, Links, Topic list, Textbook, other Recommended books, Papers listed as required reading, and Papers listed as optional or recommended reading. We refer the reader to the related SIGCSE publication [1][4] for more details on the collection and curation procedure.

The present study expands on our prior work by seeking to answer the following, detailed, research questions (RQs):

RQ1: How frequently (and in what context) is the goal of *performance* mentioned in the topic lists? We search for the string *perf* in the topic lists and identify the number of syllabi in which it appears, as well as provide specific context in which it is mentioned in the topic lists; for example Princeton's COS 418 Distributed Systems, lists the topic "Reasoning about system performance" in its syllabi.

RQ2: How frequently (and in what context) is the goal of *scalability*, and its variant *elasticity*, mentioned in the topic lists? We conduct a search similar to RQ1, using the strings *scale/scalable/scalability* for scalability and *elast* for elasticity.

RQ3: How frequently (and in what context) is the activity of *performance evaluation* mentioned in the topic lists? We search for the strings *benchmark/test/eval* in the topic lists and analyze the results, seeking to determine if issues related to system performance benchmarking, testing, or evaluation are mentioned in the syllabi.

RQ4: How frequently is the activity of *performance monitoring* mentioned in the topic lists? As for RQ3, we search for the string *monitor*, which is applicable at various levels, from monitoring single variables to entire, globally distributed systems.

RQ5: How frequently (and in what context) are *techniques used to improve performance* mentioned in the topic lists? Several performance-enhancing techniques can be discussed in the context of building distributed systems and their applications. Among them, we consider eight commonly used such techniques: replication, caching, sharding, load balancing, scheduling, streaming, migrating, and offloading. Unless otherwise noted, we use primarily these terms to search for matching items in the topic lists. For the case of sharding, we also looked for the term *partition* and consider as sharding if it refers to the same type of solution (e.g., data partitioning) but ignored it if it referred to network partitions (e.g., in the CAP theorem). For scheduling, we considered both the term *scheduling* and further *queue/queueing* (the latter did not return any results); we manually removed one instance related to transaction scheduling. For streaming, we considered the terms *stream/streaming* and also *[message] queue* and *publish[-subscribe]*.

RQ6: Which *infrastructure scale* appears discussed in the DS course? The scale of distributed systems ranges from a

[2] Available at: https://doi.org/10.5281/zenodo.4290622

[3] Available at: https://www.timeshighereducation.com/world-university-rankings/2019/subject-ranking/computer-science/.

[4] Available at: https://arxiv.org/abs/2012.00552

pair of computers acting in client-server roles, to massive global deployments of Internet-scale services. We consider the scales (and *terms*): (i) *client-server*; (ii) *cluster*, and the variant *rack* (no hits); (iii) *datacenter*, with as variant spellings *data center* and *data[ ]centre*; (iv) *grid* and *cloud* computing; (v) *peer-to-peer* (*p2p*), and the variant *decentralized*; and (vi) *global* scale, the variant *Internet[-scale]* (for which we manually exclude references to Internet protocols), and *Spanner*.

RQ7: Which *control* aspects appear in the DS course? Systems engineers try to control various performance-related aspects, among which we focus on trade-offs (term *trade*) and reducing performance variability (lists for terms *variability/variable* and *stability/stable/stabilize*, from which we eliminate topics that refer to functionality rather than performance, e.g., self-stabilizing algorithms).

RQ8: Which of the most commonly included academic papers mention performance or scale in their title? More than half (29) of the syllabi in the dataset list academic papers as recommended or required readings. While we expect a high percentage of these papers to discuss issues related to performance in their evaluation sections, RQ8 attempts to identify papers that focus very significantly on performance or scalability issues, such that these terms are explicitly mentioned in the paper titles.

3.1 Threats to dataset validity

The results in Section 4 provide an analysis of the courses in our dataset; we do not make broad claims about the overall state of the field. Our analysis seeks to find trends and patterns within good syllabi, as indicated by their inclusion in ranked programs. This is a common approach in syllabi studies, e.g., see [3, 10]. However, any biases in the rankings may bias the results (e.g., by biasing towards research universities [3]). Our data likely over-samples from anglophone countries (searches were in English), and from instructors who are comfortable sharing their syllabi. The manual collection and curation approach has its limitations and we could have missed some courses, specially if they do not contain "distributed" in their name. There may be reinforcement bias among the selected syllabi, as many syllabi are designed by consulting model curricula or are inspired from more established syllabi at other universities [3].

4 RESULTS OF QUANTIFYING PERFORMANCE ASPECTS IN DS COURSES

Herein we present the main results we have obtained in quantifying the presence of performance aspects in DS courses. Overall, our main observations are that:

MF1: 14% DS courses mention *performance* in the topic list.

MF2: 24% mention issues related to *scale* in the topic list.

MF3: None (0%) list topics related to *performance benchmarking* or *evaluation* of distributed systems.

MF4: A small fraction (6%) of the topics list mention *monitoring*.

MF5: Three-quarters of the performance-enhancing techniques we studied appear frequently in the topic lists, in order: replication (61%), streaming (27%), caching (18%), scheduling (16%), sharding (8%), and load balancing (6%). The other two, migrating (only 2%) and offloading (zero), do not.

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

MF6: The topic lists include frequent explicit references to various infrastructure scales. Among them, the three most commonly mentioned are cloud computing (29%), Internet-scale (27%), and peer-to-peer (22%). No other scale appears above 8%.

MF7: Overall, there are few references to controlling performance issues.

MF8: The reading lists contain 8 and 34 different papers with *performance* and *scal[e/able/ability]* in their titles, respectively.

We next answer each of the research questions.

RQ1: How frequently (and in what context) is *performance* mentioned in the topic lists?

We find that 14% of the syllabi contain the word *performance* in the topic list (MF1). For context, Table 1 lists the specific instances.

RQ2: How frequently (and in what context) is the goal of scalability, and its variant elasticity, included in the topics?

We find that 24% of the syllabi mention explicitly *scal[e/able/ability]* in the topic list (MF2). We only find *elasticity* mentioned in 2% of the DS courses. For context, Table 2 lists the specific instances.

RQ3: How frequently is the activity of performance evaluation mentioned?

As finding MF3, the term *benchmark*, or the related strings *test* and *eval*, do not appear in any of the topic lists (0%).

RQ4: How frequently (and in what context) is the activity of performance monitoring mentioned in the topic lists?

We find that 6% of the topic lists include monitoring (MF4). We observe both low-level monitoring of variables, which is commonly used in distributed synchronization, and high-level monitoring of collections of servers.

RQ5: How frequently (and in what context) are techniques used to improve performance mentioned in the topic lists?

We find that 6 out of 8 of the performance-enhancing techniques are well represented in the topic lists (MF5). Among the performance-enhancing techniques we study, *replication* is the most popular; this issue is listed in 61% of the syllabi. We note, however, that replication

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

is frequently discussed only in the context of fault-tolerance and less as a performance-enhancing technique.

The second most popular technique is streaming, which appears in 27% of the topic lists. Topics dealing with caching, scheduling, sharding, and load balancing appear in 18%, 14%, 8% and 6% of the topic lists, respectively. Migrating processes and offloading (parts of) applications appear less frequently as explicit topics, only 2% and never, respectively. Table 3 lists the specific instances in which five of these techniques appear in the topic lists.

RQ6: Which infrastructure scale is discussed?

We find that the topic lists include frequent explicit references to various infrastructure scales (MF6). In increasing order of the maximum scale observed in practice,[5] the frequencies are: (i) *client-server*, only 2%; (ii) *cluster*, 6%, with the variant *rack* receiving zero explicit mentions; (iii) *datacenter*, 8%; (iv) *grid* and *cloud* computing, 4% and 29%, respectively, indicating their popularity; (v) *peer-to-peer*, 24%, including the variant *decentralized*, at 1%; and (vi) *global* scale, 27%, including the variant *Internet[-scale]*, at 4%, and *Spanner*, at 22%, the most common example of a global-scale system.

The very common presence of cloud computing is not surprising, given the wide applicability of the concept and its popularity in current practice. In contrast, Spanner [4] is surprisingly popular; we attribute this to the high-visibility publication venue, the brand of the Google engineering team, the high quality of concepts present

[5]Whereas the order server, cluster, datacenter is natural, the overuse of terms grid, cloud, and peer-to-peer makes their relative ordering more difficult to consider. We propose an ordering based on number of geographical locations. The largest grids span tens to hundreds of locations; public clouds span at most a few tens of geographically-distributed datacenters. In contrast, the largest P2P systems, for example, the large-scale deployments of Bittorrent for file-sharing [26], and of BOINC or similar for volunteer computing [11], reached at-peak hundreds of millions of locations in the world.

Table 3: Topics in the topic lists of the surveyed syllabi, in which performance-enhancing techniques (shown in bold) are mentioned. Of the 8 techniques we cover in this work, this table presents the results for replication, caching, scheduling, sharding, and load balancing. Topics that appear written in the same way in multiple topic lists appear only once in the table, with multiple instances denoted in parentheses.

Topic
Caching and **replication**
Consensus: Passive **replication** (raft), active **replication**, lazy **replication** (gossip, Bayou)
Consistency and **replication** (x6)
Data **replication**
Data **replication** and consistency: Overview, consistency models
Distributed **replication**
Fault-tolerance: **Replicated** and fusible state machine approaches
MapReduce and **replicated** state machines (x2)
Primary-backup **replication** (x3)
Replicated state machines (x2)
Replication (x4)
Replication (primary/backup, quorum protocols, sequential and causal consistency, client-centric models)
Replication (**replicated** state machine, primary/backup, quorum **replication**)
Replication control
Replication in distributed systems
Replication in the HARP file system
Replication, active **replication**, primary-backup **replication**, gossip-based **replication**
Replication, data-centric consistency
Replication: Epidemic algorithms, Bayou
Scalable services, reliability, and consistency: Scale and recovery for storage, leases, linearizable RPC for a **replicated** storage service
Cache consistency (x2)
Caching and consistency: NFS and the Web
Caching and replication
Consistency and replication: Consistency models, consistency protocols, replica management, **caching**
Distributed file systems: Architecture, **caching**, semantics
Facebook photo **cache**
Memcache
Web **caching** and consistent hashing
Distributed Computing: **Scheduling** and Resource Management
Local **Scheduling**: Scale-out threads
Multiprocessor and Distributed **Scheduling**
Scheduling (x3)
Scheduling and Load Balancing in distributed systems
Sharding
Sharding and distributed transactions
Elastic services in the cloud: [...] auto-**sharding** and **sharded** request routing (see full topic name in Table 2)
Data partitioning strategies (authors' note: same as sharding)
Load balancing: LARD, internet-scale services
Scheduling and **load balancing** in distributed systems
Elastic services in the cloud: [...] request routing and **load balancing**: into the network, [...] (see full topic name in Table 2)

in the Spanner article, and also to the maturity of both design and implementation that the team explains is the outcome of over five years of system, software, and performance engineering.

RQ7: Which control aspects appear in the DS course?

We find few references to controlling performance issues (MF7).

The systems view is often focused on trade-offs. Performance is a common element in trade-offs in practice, against functional system properties such as consistency and non-functionals such as availability (e.g., in the PACELCA view extending the CAP theorem in the last decade), but also against other non-functionals, such as energy consumption and cost. We thus find it surprising that the term trade-off appears in only 6% of the topic lists.

Given the prominence of cloud among the topic lists, and the prevalence of the phenomenon of performance variability in clouds in the past decade [12, 25], we find it even more surprising that zero (0%) of the topic lists include *variability* or related terms.

RQ8: Which of the most commonly included academic papers mention performance or scale in their title?

The reading lists contain 8 and 34 different papers with *performance* and *scal[e/able/ability]* in their titles, respectively (MF8). Table 4 contains the list of paper titles that contain these terms.

5 RECOMMENDATIONS ON HOW DS SYLLABI ADDRESS PERFORMANCE

Include more performance topics, and more explicitly, in the DS curriculum. Less than one fourth of the syllabi explicitly mention *performance* (14%) or *scale* (24%) in the topic lists. However, these issues may be well covered within other topics in the topic lists (e.g., remote invocation and cluster management, which appear in 67% and 24% of the topic lists [1]) or at independent performance-focused courses at the corresponding institutions [5] (for example, the Politecnico di Milano which has a DS course and a course on Performance Evaluation of Computer Systems [6]).

Focus on performance & scale as first-class concepts. The software engineering community has repeatedly noted that performance and scalability issues tend to be an afterthought in the development process [18, 20, 22]. These issues should be explicitly listed in the syllabi and not buried under other topics. Doing so would increase their visibility, solidify the idea that software performance is important, and possibly even help attract more students to an important course that is frequently listed as elective.

Add learning goals focusing on benchmarking or experimental evaluation of DS. None (0%) of the syllabi in our dataset list topics related to benchmarking or experimental evaluation of distributed systems. Given that courses focusing on this issue are even less popular than Distributed Systems courses, and given that important curricular initiatives [13] have found this to be an important topic to be included in CS programs, we think the community should consider making distributed systems evaluation a first-class topic instead of relegating it to a project or ignoring it altogether.

Consider including monitoring in DS syllabi. Another essential activity in distributed systems, monitoring, appears to be under-valued by current DS syllabi. We recommend introducing this topic, which raises important conceptual challenges and has resulted in notable practical results such as CERN's MonALISA [17], Ganglia [19], and more recently, DevOps pipelines [21] leveraging Prometheus and Grafana.[6] While monitoring issues may be discussed in a Software Engineering course, the challenges of monitoring large-scale systems are arguably better understood in a DS class. Furthermore, teaching performance-enhancing techniques without also including how to properly verify how they work (i.e., through testing, analysis, monitoring) greatly reduces the value and impact of the technique.[7]

Increase coverage of performance-enhancing techniques used in practice and their trade-offs. Many performance-enhancing techniques are already widely present in DS syllabi, starting with replication. However, we found that important techniques that are commonly used in practice, such as process migration and component offloading, have a negligible presence in the topic lists of modern Distributed Systems courses. Also neglected are aspects critical to controlling the performance of distributed systems, such as understanding the trade-offs where performance plays a prominent role and reducing performance variability, do not currently appear explicitly in the DS syllabi we study. While these trade-offs are likely discussed in class, explicitly listing the trade-offs as topics being studied could help students be more aware of their importance and can also help highlight the relevance of the DS course in engineering practice.

6 CONCLUSION

We identified the need to study the presence of performance aspects in Distributed Systems syllabi. Addressing this need, we conducted an analysis of 51 Ditributed Systems syllabi, focusing on finding the prevalence and context in which topics related to performance are being taught in these courses. Our analysis represents the most comprehensive study of its kind, to date. It covers for performance engineering a number of common goals, activities, techniques, and control aspects. We also study the scale of the infrastructure mentioned in DS courses, from small client-server systems to cloud-scale, peer-to-peer, global-scale systems. We make eight main findings, covering goals such as performance, and scalability and its variant elasticity; activities such as performance benchmarking and monitoring; eight selected performance-enhancing techniques, that is, replication, caching, sharding, load balancing, scheduling, streaming, migrating, and offloading; and control issues such as trade-offs that include performance and performance variability.

We envision the use of this work in the development of DS courses that focus more on performance topics. We also foresee our method for syllabus analysis being leveraged by directors of education in full-curriculum analysis, as element in their reporting and support in their decision-making processes.

ACKNOWLEDGMENTS

We thank the anonymous WEPPE reviewers that thoroughly read our paper and provided invaluable feedback that helped improve the quality of this manuscript. Part of this work was sponsored by a Google Faculty Research Award (CA, EB), and the NWO (AI), projects MagnaData and OffSense.

[6]See https://grafana.com/oss/prometheus/ and https://grafana.com/oss/grafana/
[7]We thank our anonymous referees for pointing this out.

Table 4: List of paper titles included in the DS syllabi as required or recommended readings, that contain performance (top, 8 papers) or scal(e/able/ability) (bottom, 34 papers).

Paper Title
Building Secure High-Performance Web Services with OKWS
Characteristics of Scalability and Their Impact on Performance
Implementation and Performance of Munin
No compromises: Distributed transactions with consistency, availability, and performance
Paxos Replicated State Machines as the Basis of a High-Performance Data Store
Performance debugging for distributed systems of black boxes
The Akamai Network: A Platform for High-Performance Internet Applications
The Performance Implications of Thread Management Alternatives for Shared-Memory Multiprocessors
A scalable content-addressable network
Algorand: Scaling Byzantine Agreements for Cryptocurrencies
Bitcoin-NG: A Scalable Blockchain Protocol
Characteristics of Scalability and Their Impact on Performance
Chord: a scalable peer-to-peer lookup protocol for internet applications
Cluster-based file replication in large-scale distributed systems
Dapper, a Large-Scale Distributed Systems Tracing Infrastructure
Discretized Streams: A Fault-Tolerant Model for Scalable Stream Processing
Don't Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS
Experiences with a Distributed, Scalable, Methodological File System: AnalogicFS
Exploiting a Natural Network Effect for Scalable, Fine-grained Clock Synchronization
Frangipani: A Scalable Distributed File System
Highly Scalable Algorithm For Distributed Real-Time Text Indexing
Implementing linearizability at large scale and low latency
Large-scale cluster management at Google with Borg
Lessons from Giant-Scale Services
OceanStore: An Architecture for Global-Scale Persistent Storage
Omega: Flexible, scalable schedulers for large compute clusters
On scalable and efficient distributed failure detectors
PRAM: A Scalable Shared Memory
Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems
Pregel: A system for large-scale graph processing
SVE: Distributed Video Processing at Facebook Scale
SWIM: Scalable Weakly-consistent Infection-style Process Group Membership Protocol
Scalable Application Layer Multicast
Scaling Distributed Machine Learning with the Parameter Server
Scaling Memcache at Facebook
Sinfonia: A new paradigm for building scalable distributed systems
Spotify - Large Scale, Low Latency, P2P Music-on-Demand Streaming
Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility
Tapestry: A Resilient Global-scale Overlay for Service Deployment
TensorFlow: A System for Large-Scale Machine Learning
The tail at scale
ZooKeeper: Wait-free Coordination for Internet-scale Systems

REFERENCES

[1] Cristina Abad, Eduardo Ortiz-Holguin, and Edwin Boza. 2021. Have we reached consensus? An Analysis of Distributed Systems Syllabi. In *ACM Technical Symposium on Computer Science Education (SIGCSE TS)*.

[2] Varsha Apte. 2019. "What Did I Learn In Performance Analysis Last Year?": Teaching Queuing Theory for Long-Term Retention. In *Companion of the ACM/SPEC International Conference on Performance Engineering*.

[3] Brett Becker and Thomas Fitzpatrick. 2019. What Do CS1 Syllabi Reveal About Our Expectations of Introductory Programming Students?. In *ACM Technical Symposium on Computer Science Education (SIGCSE TS)*.

[4] James Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. 2013. Spanner: Google's Globally Distributed Database. *ACM Transactions of Computer Systems* 31, 3 (2013).

[5] Vittoria de Nitto Personè. 2017. Teaching Performance Modeling in the Era of *140characters Information*. In *Companion of the ACM/SPEC International Conference on Performance Engineering*.

[6] Vittoria de Nitto Personè. 2020. Teaching Performance Modeling in the era of millennials. arXiv:2001.08949 [cs.CY]

[7] Saumya Debray. 2004. Writing Efficient Programs: Performance Issues in an Undergraduate CS Curriculum. *SIGCSE Bulletin* 36, 1 (March 2004).

[8] Daniel Ernst. 2011. Preparing Students for Future Architectures with an Exploration of Multi- and Many-Core Performance. In *Conference on Innovation and Technology in Computer Science Education (ITiCSE)*.

[9] Allan Fisher and Thomas Gross. 1992. Teaching Empirical Performance Analysis of Parallel Programs. *SIGCSE Bulletin* 24, 1 (March 1992).

[10] Nadjim Fréchet, Justin Savoie, and Yannick Dufresne. 2020. Analysis of Text-Analysis Syllabi: Building a Text-Analysis Syllabus Using Scaling. *Political Science & Politics* 53, 2 (2020).

[11] Daniel Lázaro Iglesias, Derrick Kondo, and Joan Manuel Marquès. 2012. Long-term availability prediction for groups of volunteer resources. *Journal of Parallel Distributed Computing* 72, 2 (2012).

[12] Alexandru Iosup, Nezih Yigitbasi, and Dick Epema. 2011. On the Performance Variability of Production Cloud Services. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*.

[13] ACM Joint Task Force on Computing Curricula and IEEE Computer Society. 2013. Computer Science Curricula 2013. Final Report.

[14] ACM Joint Task Force on Computing Curricula and IEEE Computer Society. 2015. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering 2014. Final Report.

[15] ACM Joint Task Force on Computing Curricula and IEEE Computer Society. 2016. Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering 2016. Final Report.

[16] ACM Joint Task Force on Computing Curricula and IEEE Computer Society. 2017. Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology 2017. Final Report.

[17] Iosif Legrand, Harvey Newman, Ramiro Voicu, Catalin Cirstoiu, et al. 2009. MonALISA: An agent based, dynamic service system to monitor, control and optimize distributed systems. *Computer Physics Communications* 180, 12 (2009).

[18] Henry Liu. 2011. *Software performance and scalability: A quantitative approach*. Vol. 7. John Wiley & Sons.

[19] Matthew Massie, Brent Chun, and David Culler. 2004. The Ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Comput.* 30, 7 (2004).

[20] Daniel Menasce, Virgilio Almeida, Lawrence Dowdy, and Larry Dowdy. 2004. *Performance by design: Computer capacity planning by example*. Prentice Hall.

[21] Marco Miglierina and Damian Tamburri. 2017. Towards Omnia: A monitoring factory for quality-aware DevOps. In *ACM/SPEC on International Conference on Performance Engineering Companion (ICPE)*.

[22] Kay Ousterhout, Christopher Canel, Max Wolffe, Sylvia Ratnasamy, and Scott Shenker. 2017. Performance clarity as a first-class design principle. In *Workshop on Hot Topics in Operating Systems (HotOS)*.

[23] Charles Shub. 1989. Performance Experiments for the Performance Course. *SIGCSE Bulletin* 21, 1 (Feb. 1989).

[24] Y. Tay. 2019. Lessons from Teaching Analytical Performance Modeling. In *Companion of the ACM/SPEC International Conference on Performance Engineering*.

[25] Alexandru Uta, Alexandru Custura, Dmitry Duplyakin, Ivo Jimenez, Jan Rellermeyer, Carlos Maltzahn, Robert Ricci, and Alexandru Iosup. 2020. Is Big Data Performance Reproducible in Modern Cloud Networks?. In *USENIX NSDI*.

[26] Maciej Wojciechowski, Mihai Capota, Johan A. Pouwelse, and Alexandru Iosup. 2010. BTWorld: Towards observing the global BitTorrent file-sharing network. In *ACM Symposium on High Performance Distributed Computing (HPDC)*.