Characterizing Service Level Objectives for Cloud Services: Realities and Myths

Jerry Ding

Undergraduate Research Thesis Oral Defense

Data is available: http://go.osu.edu/ slodataset-v1-xls

Intro

- Service level objectives (SLOs) outline quality of service goals.
- Over 15 years, performance goals have become pervasive.

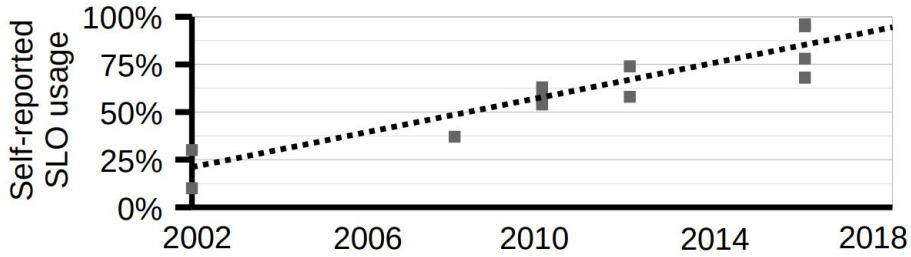
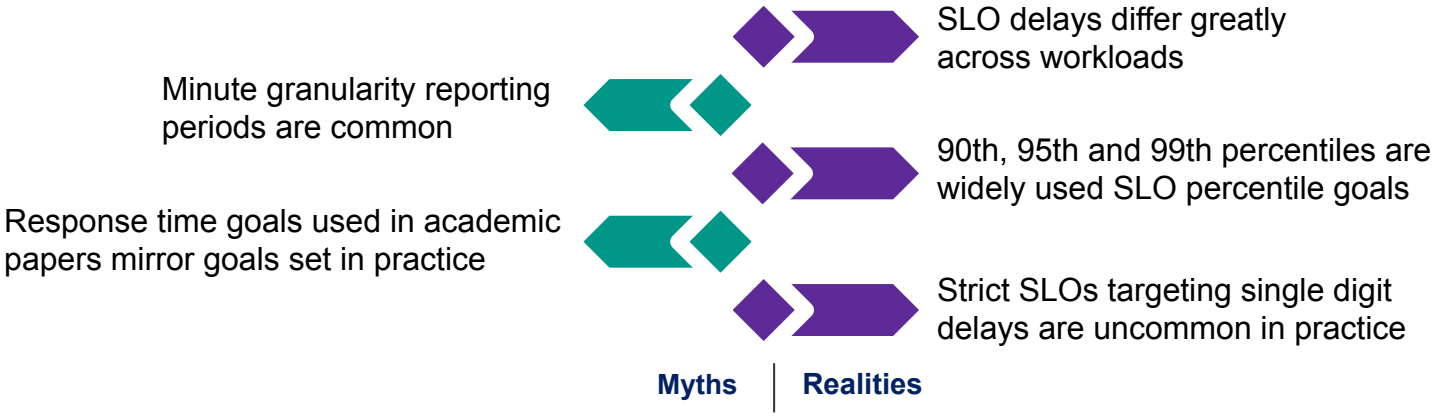


Fig. 1. Service level objectives (SLOs) have been widely adopted. Each dot represents the percentage of system administrators (Yaxis) adopting performance-oriented SLOs in the surveyed year (Xaxis) [24], [5], [15], [10], [7], [6].

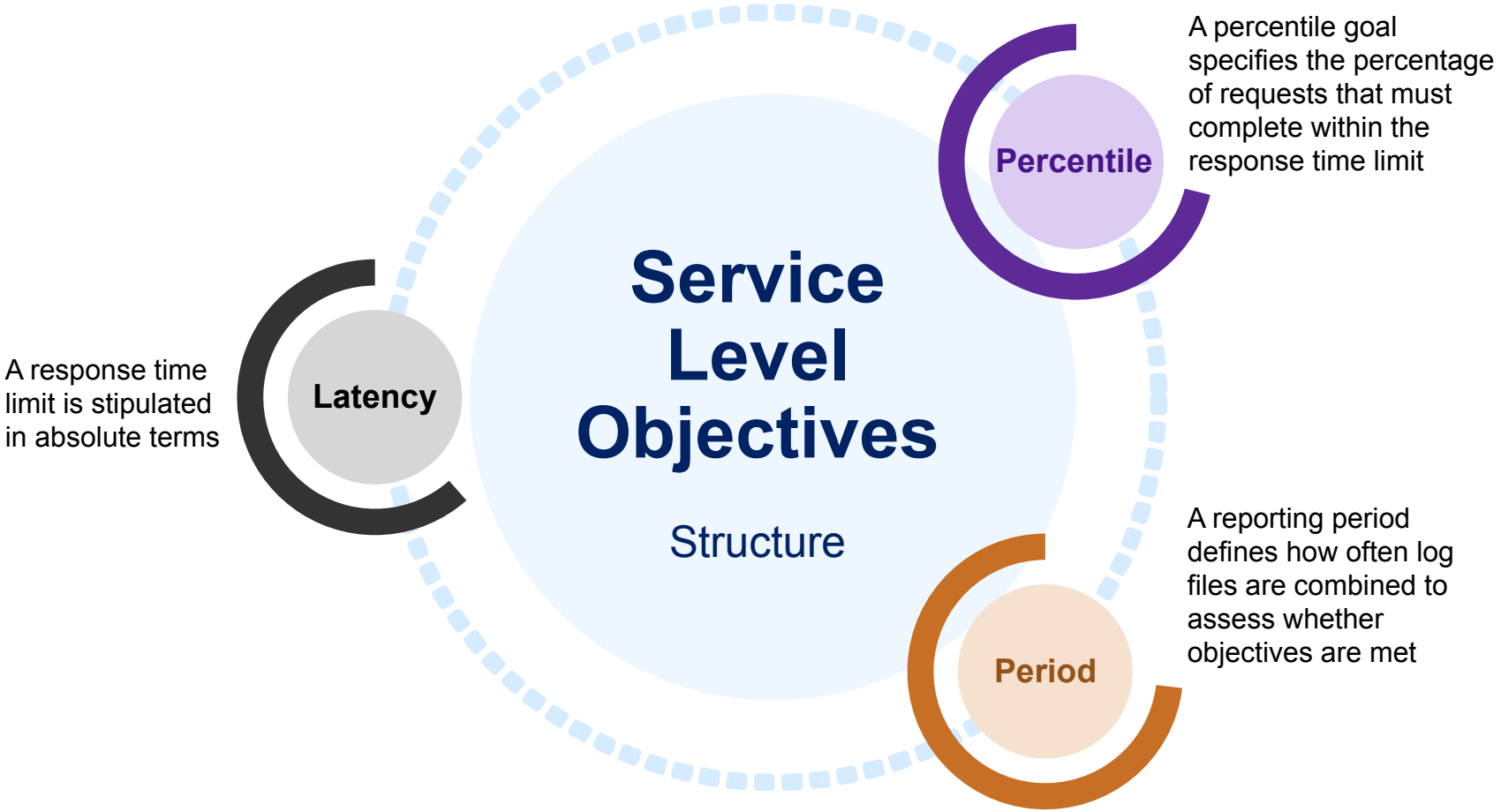
Intro

- However, textbooks only sketch goals and rely heavily on anecdotal examples.
- We studied real SLOs, quantified their performance goals and labeled **common perceptions** about their goals as **realities** or **myths**.



Overview

- Service Level Objectives
- Systematic Literature Review
- Characterizing Realities and Myths
- Discussion


Service Level Objectives
Structure
Latency
A response time limit is stipulated in absolute terms
Percentile
A percentile goal specifies the percentage of requests that must complete within the response time limit
Period
A reporting period defines how often log files are combined to assess whether objectives are met

Service Level Objective

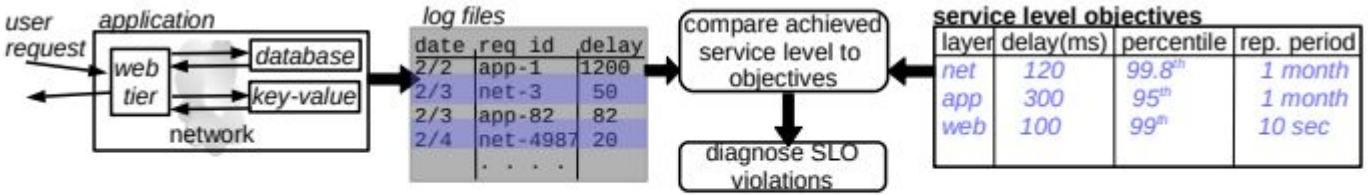


Fig. 2. The structure of SLO performance goals (right) reflects the workflow of cloud services (left). Logs highlighted in blue are aggregrated during a reporting period.

Service Level Objective Examples

CXENSE Service Level Agreement [8]

95% of all ad requests will achieve a response time of 300 milliseconds or less.

WorldCom/UUNET VPN [21]

99.8% of VPN traffic should transfer between customer premise equipment (CPE), e.g., routers, in less than 120 ms.

Site Reliability at Google [4]

99% of the RPC microservices should be processed in 100ms.

Systematic Literature Review

- It provides structure, bias mitigation, rigor and repeatable results [1].
- Researchers can repeat strategies and compare alternative strategies to assess completeness [1].
- SLR consists of 4 steps.

Systematic Literature Review

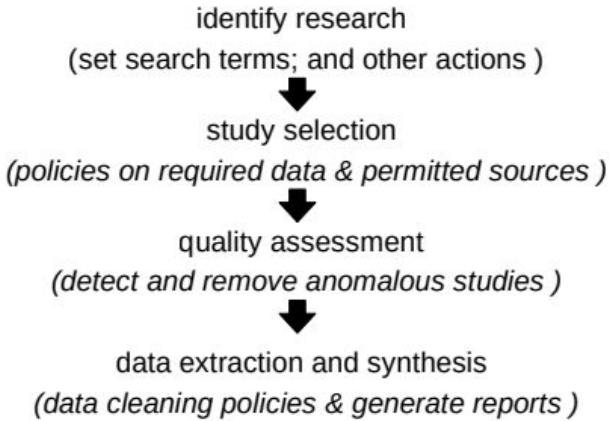


Fig. 3. Systematic literature review procedure

SLR – Search Strategy

- PICOC criteria (Population, Intervention, Comparison, Outcomes, Context) constrains search structure [26].

$$\text{string} = \text{population} + \text{intervention} + \text{outcome} + \text{context} \quad (1)$$

- Population refers to the types of computer systems under review.
- Intervention provides the types of SLO features sought.
- Outcome describes the target metrics from intervention.
- Context is the types of documents used in our search such as primary studies and company handbooks

$$\text{String} = \text{VPN service } AND \text{ latency } AND \text{ ms } AND \text{ University} \quad (2)$$

SLR – Literature Selection & Quality Assessment

8,530 academic papers

1,104 other documents

↓

300 documents

↓

75 SLOs

from 34 academic sources and 41 industry sources

SLO features	req.	SLR layer
reporting period	yes	study selection
delay	yes	study selection
uptime/availability	no	study selection
type of cloud service	no	study selection
percentile	yes	study selection
anomalous workload	---	quality assessment
affliation	---	data extraction
source profile	---	data extraction

TABLE I
INCLUSION AND EXCLUSION CRITERIA AND ADDITIONAL FEATURES EXTRACTED.

SLR - Data Extraction and Synthesis

SLO features	req.	SLR layer
reporting period	yes	study selection
delay	yes	study selection
uptime/availability	no	study selection
type of cloud service	no	study selection
percentile	yes	study selection
anomalous workload	---	quality assessment
affliation	---	data extraction
source profile	---	data extraction

TABLE I
INCLUSION AND EXCLUSION CRITERIA AND ADDITIONAL FEATURES EXTRACTED.

CHARACTERIZING REALITIES AND MYTHS

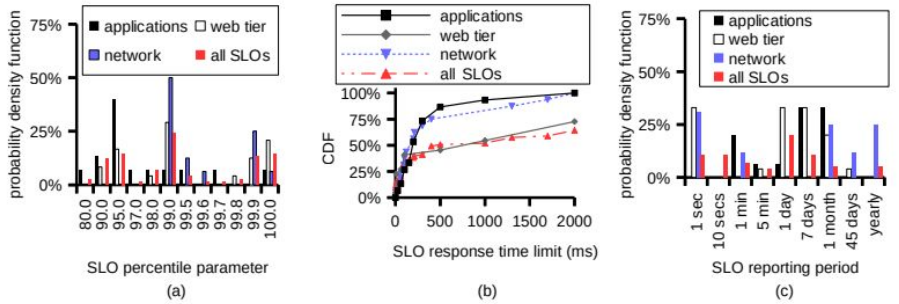


Fig. 4. Distributions of (a) SLO percentiles, (b) delay goals and (c) reporting periods. CDF stands for cummulative distribution function.

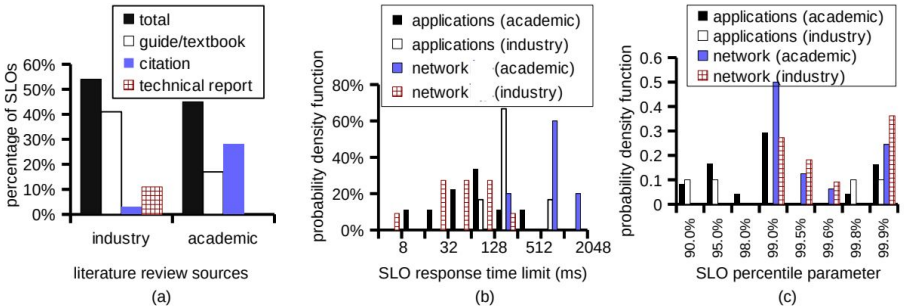


Fig. 5. (a) distribution of sources in our study, (b) distribution of SLO response time goals grouped by industry and academic sources, (c) distribution of SLO percentile goals grouped by source.

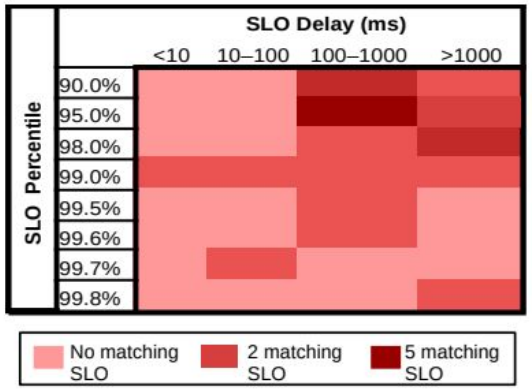
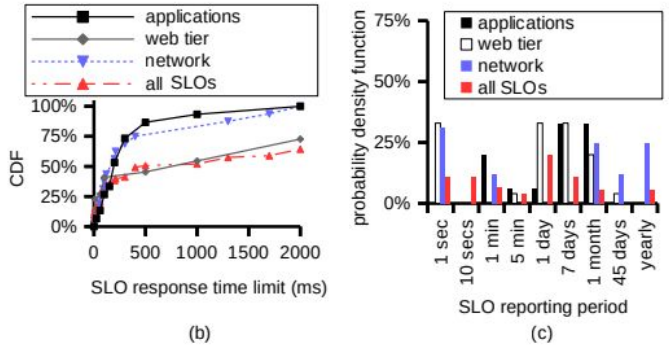


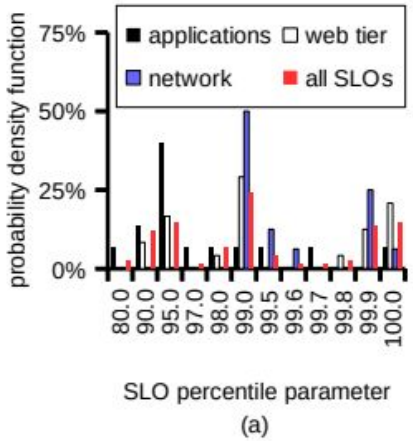
Fig. 6. Heat map of delay and percentile goals from industry sources.

Reality 1: SLO delays and reporting periods differ greatly across workloads.



- Approximately 40% of SLOs across the 3 workload types define their response time limits in the range 0-100ms.
- Network and software applications rarely set their SLOs greater than 500ms.
- Network SLOs set their reporting periods less than 1 minute or more than 1 month but nothing in between.
- Web services and applications have reporting periods concentrated in the range of minutes to 1 month.

Reality 2: 90th , 95th and 99th percentiles are widely used SLO percentage goals.



- Software applications and web services have a wide range of percentiles varying from 80% to 99.9%.
- Over 35% of SLOs for software applications use the 95th percentile while the majority of web services use 99th.
- A large fraction of SLOs for network services use the 99th percentile as well, but its distribution differs from the other workloads.

Reality 3: Strict SLOs targeting single-digit delays are uncommon in practice.

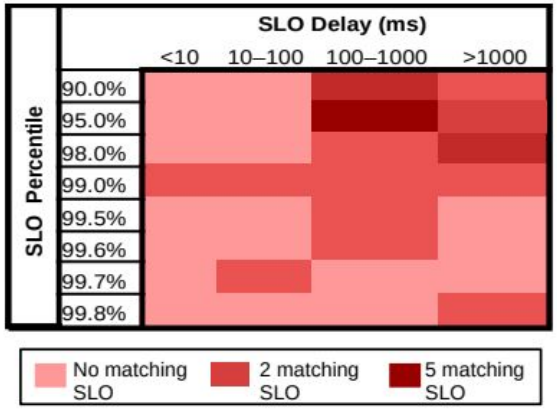
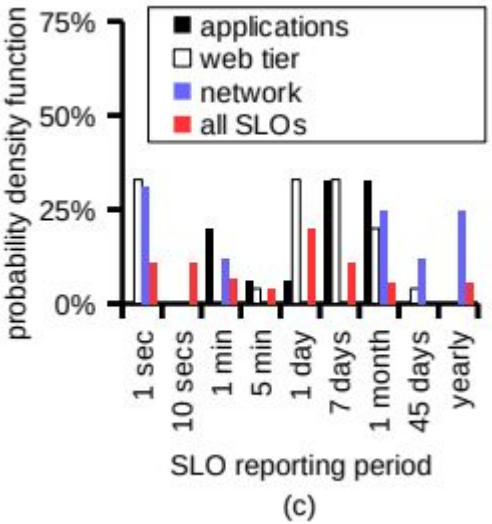


Fig. 6. Heat map of delay and percentile goals from industry sources.

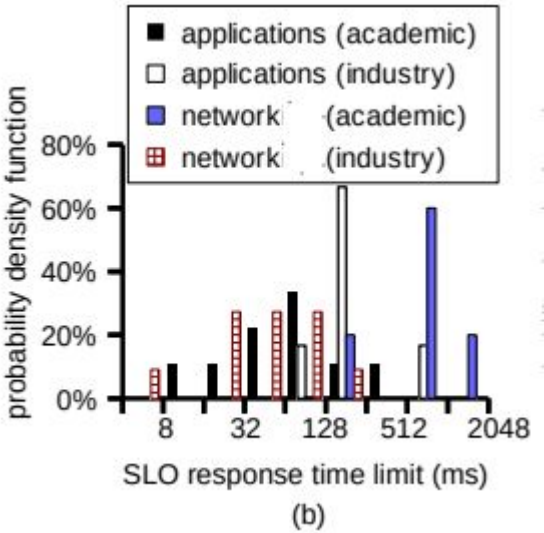
- The SLOs tend to focus on the top right corner in Figure 6.
- The majority of delays are between 100-1000 ms with most of the rest falling in the range of larger than 1000ms.
- Only 3 SLO delays are less 100ms with only 1 of them of single-digit delay in ms.

Myth 1: 5–10 minute reporting periods.



- Surprisingly, we found few SLOs with minute-granularity reporting periods. They consist of only less than 15% of our samples.

Myth 2: Response time goals used in academic papers mirror goals set in practice.



- On network services, academic sources are much more lenient.
- For whole applications, they are too strict.
- We can report that both distributions are significantly different (p=0.9).

Discussion

- Our sample size is small and limits our ability to extrapolate.
- A larger sample would require proprietary data— perhaps from large surveys of system managers.

- Performance goals in some sources reflect real case studies whereas other sources reflect vague recollections of authors.
- We mitigated this problem by distinguishing the source characteristics, e.g., industry manual versus case study.

Translational evaluation for systems research.

Conclusion - 1

- Industry adopts research slowly, replicating and validating results before integration [20].
- Our study suggests evaluation approaches used in research papers (not included in the study) may not reflect uses cases in practice.

Open challenges for SLO performance goals.

Conclusion - 2

- Response time limits below 10 milliseconds are challenging.
- Prior research has shown it is possible to achieve extreme percentiles by inefficiently replicating resources [29].
- However, cost effective approaches remain elusive in practice.
- Computational sprinting has emerged as a possible solution [12], [22], [11].

Potpourri

This work was funded by NSF Grants 1749501 and 1350941. We thank co-writers Ruiqi Cao, Indrajeet Saravanan, Nathaniel Morris and our advisor, Christopher Stewart. We also thank our shepherd, Dr. Timothy Zhu. Response time limits, reporting periods, percentiles and references for the service level objectives used in this study are available [19].

References

[1] K. BA and S. Charters. Guidelines for performing systematic literature reviews in software engineering. EBSE Technical Report, 2007.
[2] G. Banga, P. Druschel, and J. C. Mogul. Resource containers: A new facility for resource management in server systems. In OSDI, 1999.
[3] L. A. Barroso, M. Marty, D. A. Patterson, and P. Ranganathan. Attack of the killer microseconds. Communications of the ACM, 2017.
[4] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy. Site Reliability Engineering: How Google Runs Production Systems. ” O’Reilly Media, Inc.”, 2016.
[5] R. Blum. Service level management and service level agreements survey report, 2002.
[6] M. Connections. Poll finds poor communications and changing requirements can derail any government it project, 2013.
[7] I. T. I. Council. Measuring performance. https://itic-corp.com/ news-analysis/archive/attachment/06/, 2009.
[8] cxense inc. Cxense service level agreement - valid from may 3rd, 2018. https://support.cxense.com/.
[9] N. Deng, C. Stewart, D. Gmach, M. Arlitt, and J. Kelley. Adaptive green hosting. In IEEE ICAC, 2012.
[10] Enterprise Management Associates. https: //searchitoperations.techtarget.com/news/1363066/
Has-the-down-economy-driven-data-center-automation, 2009.
[11] S. Fan, S. M. Zahedi, and B. C. Lee. The computational sprinting game. ACM SIGOPS Operating Systems Review, 50(2), 2016.
[12] M. E. Haque, Y. h. Eom, Y. He, S. Elnikety, R. Bianchini, and K. S. McKinley. Few-to-many: Incremental parallelism for reducing tail latency in interactive services. ASPLOS, 2015.
[13] A. Harlap, A. Chung, A. Tumanov, G. R. Ganger, and P. B. Gibbons. Tributary: spot-dancing for elastic services with latency slos. In USENIX ATC, 2018.
[14] R. Hat. Introduction to control groups, 2019.
[15] IS Scorecard. Is scorecard: Service level management. https: //www.computereconomics.com/article.cfm?id=961, 2004.
[16] P. Janus and K. Rzadca. Slo-aware colocation of data center tasks based on instantaneous processor requirements. In Symposium on Cloud Computing, 2017.
[17] S. A. Javadi and A. Gandhi. Dial: Reducing tail latencies for cloud applications via dynamic interference-aware load balancing. In ICAC, 2017.

References

[18] H. Jayathilaka, C. Krintz, and R. Wolski. Response time service level agreements for cloud-hosted web applications. In Symposium on Cloud Computing, 2015.

[19] N. M. Jianru Ding and C. Stewart. A compendium of service level objectives for networked services. http://go.osu.edu/ slodataset-v1-xls, 2019.

[20] C. Landauer and K. L. Bellman. Model-based cooperative system engineering and integration. In ICAC, 2016.

[21] J. Martin and A. Nilsson. On service level agreements for ip networks. In IEEE INFOCOM, 2012.

[22] N. Morris, S. M. Renganathan, C. Stewart, R. Birke, and L. Chen. Sprint ability: How well does your software exploit bursts in processing capacity? In ICAC, 2016.

[23] N. Morris, C. Stewart, L. Chen, R. Birke, and J. Kelley:. Modeldriven computational sprinting. In ACM European Conference on Computer Systems, 2018.

[24] H. Muscolino. 2019 automation survey: The path to digital transformation. https://documentmedia.com/, 2019.

[25] P. Nguyen and K. Nahrstedt. Monad: Self-adaptive micro-service infrastructure for heterogeneous scientific workflows. In ICAC, 2017.

[26] M. Petticrew and H. Roberts. Systematic reviews in the social sciences: A practical guide. Systematic Reviews in the Social Sciences: A Practical Guide, 2006.

[27] K. Shen, A. Shriraman, S. Dwarkadas, X. Zhang, and Z. Chen. Power containers: An os facility for fine-grained power and energy management on multicore servers. In ASPLOS, 2012.

[28] A. Sriraman and T. F. Wenisch. µ suite: A benchmark suite for microservices. In IISWC, 2018.

[29] C. Stewart, A. Chakrabarti, and R. Griffith. Zoolander: Efficiently meeting very strict, low-latency slos. In IEEE International Conference on Autonomic Computing, 2013.

[30] S. K. Tesfatsion, E. Wadbro, and J. Tordsson. Autonomic resource management for optimized power and performance in multi-tenant clouds. In ICAC, 2016.

[31] A. Wang, S. Venkataraman, S. Alspaugh, R. Katz, and I. Stoica. Cake: enabling high-level slos on shared storage systems. In Symposium on Cloud Computing, 2012.

References

[32] C. Wang, B. Urgaonkar, A. Gupta, L. Y. Chen, R. Birke, and G. Kesidis. Effective capacity modulation as an explicit control knob for public cloud profitability. In ICAC, 2016.
[33] Z. Xu, C. Stewart, N. Deng, and X. Wang. Blending on-demand and spot instances to lower costs for in-memory storage. In IEEE INFOCOM, 2016.
[34] N. Zacheilas and V. Kalogeraki. Chess: Cost-effective scheduling across multiple heterogeneous mapreduce clusters. In ICAC, 2016.
[35] W. Zhang, T. Wood, and J. Hwang. Netkv: Scalable, selfmanaging, load balancing as a network function. In ICAC, 2016.

Thanks for Listening!

Jerry Ding