Characterizing Service Level Objectives for Cloud Services: Realities and Myths

Jianru Ding, Ruiqi Cao, Indrajeet Saravanan, Nathaniel Morris and Christopher Stewart
The Ohio State University

***Abstract*—Service level objectives (SLOs) stipulate performance goals for cloud applications, microservices, and infrastructure. SLOs are widely used, in part, because system managers can tailor goals to their products, companies, and workloads. Systems research intended to support strong SLOs should target realistic performance goals used by system managers in the field. Evaluations conducted with uncommon SLO goals may not translate to real systems. Some textbooks discuss the structure of SLOs but (1) they only sketch SLO goals and (2) they use outdated examples. We mined real SLOs published on the web, extracted their goals and characterized them. Many web documents discuss SLOs loosely but few provide details and reflect real settings. Systematic literature review (SLR) prunes results and reduces bias by (1) modeling expected SLO structure and (2) detecting and removing outliers. We collected 75 SLOs where response time, query percentile and reporting period were specified. We used these SLOs to confirm and refute common perceptions. For example, we found few SLOs with response time guarantees below 10 ms for 90% or more queries. This reality bolsters perceptions that single digit SLOs face fundamental research challenges.**

I. Introduction

Service level objectives (SLOs) outline quality of service goals. Service providers must satisfy these goals or risk penalties. For example, an SLO may require high availability goals, i.e., clients can request services at any time. Failure to meet this goal may reduce service provider compensation.

Cloud computing workloads integrate SLOs deeply. SLOs govern power and resource delivery in data centers [9], [29], [31], networking infrastructure [34], [28], microservices [24], [27], and user-facing cloud applications. SLOs allow cloud applications to safely embed content from third parties [24], [20], integrate proprietary and open-source software, and account for resource usage [22], [11].

Increasingly, cloud services include response time and percentile goals in SLOs [18], [30]. Figure 1 plots self-reported usage of SLO performance goals in system manager surveys. Over 15 years, performance goals have become pervasive. In a recent survey, 4 in 5 system managers report setting up performance goals [23]. Satisfying performance goals requires software and hardware resource provisioning— a capability afforded in cloud computing environments. However, performance goals can differ greatly across products, companies, and workloads. As performance goals become commonplace, it is key to understand how they are applied in practice.

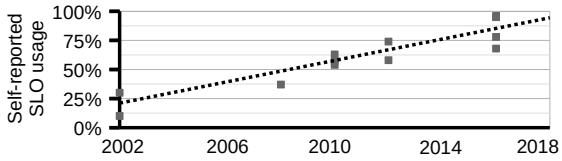

Fig. 1. Service level objectives (SLOs) have been widely adopted. Each dot represents the percentage of system administrators (Y-axis) adopting performance-oriented SLOs in the surveyed year (X-axis) [23], [5], [15], [10], [7], [6].

Some cloud computing textbooks discuss the structure of SLOs, including performance goals [4]. However, they only sketch goals and rely heavily on anecdotal examples. These sources introduce students to SLOs, but details regarding SLO goals could suffer bias. Further, performance goals are an emerging trend. Details change often. Textbook authors will struggle to keep content up to date, especially as new workloads and expectations emerge.

This paper studies real SLOs, quantifies their performance goals and labels common perceptions about their goals as realities or myths. We found 9,634 web documents related to SLOs, but most contained vague descriptions and excluded key performance details. We used a systematic literature review (SLR) to prune results. First, we modeled the structure of cloud SLOs, pruning documents that didn't include response time (delay), percentiles, and reporting period parameters. Then, we used outlier detection to study and remove anomalous SLOs. In total, we collected 75 SLOs from 34 academic sources and 41 industry sources (e.g., white papers and manuals). Our SLOs cover a wide range of cloud workloads, including databases, file systems, networking infrastructure, web tier workloads, and whole applications.

Without concrete studies, researchers have relied on common assumptions about SLO performance goals. While 75 SLOs do not comprehensively represent all SLOs, our dataset does corroborate 3 common assump-

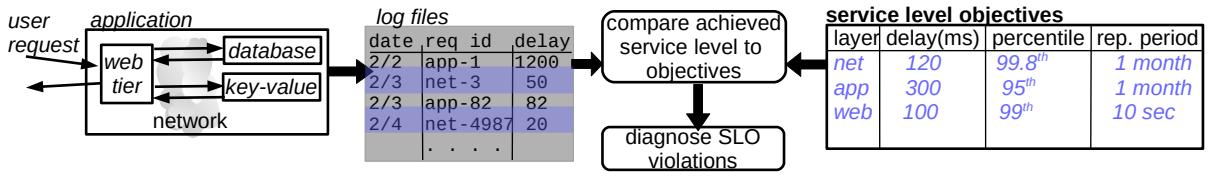

Fig. 2. The structure of SLO performance goals (right) reflects the workflow of cloud services (left). Logs highlighted in blue are aggregrated during a reporting period.

tions: (1) SLO delays differ greatly across workloads, (2) 90^{th}, 95^{th} and 99^{th} percentiles are widely used SLO percentile goals, and (3) strict SLOs targeting single-digit delays are uncommon in practice. Our analysis does not corroborate the following assumptions: (1) Minute granularity reporting periods are common (2) response time goals used in academic papers mirror goals set in practice. These realities and myths have implications for systems and autonomic computing research. Resource scheduling techniques should be evaluated on multiple strict percentiles. Novel infrastructure should be evaluated across settings that affect response time goals. These heuristics would align research evaluations and industry practices. Further, our analysis reveals research opportunities for ultra-low latency SLOs and for studying performance over long reporting periods.

Our contributions are:

1. We collected diverse and realistic cloud SLOs using systematic literature review.

2. We group and compare SLOs by source, target workload and performance goals. Our analysis confirms common assumptions and provides reason to reconsider others.

The remainder of this paper is as follows: Section II provides background and describes the structure of SLOs. Section III describes our SLR study. Section IV quantifies our SLO performance goals and uses them to confirm realities and refute myths. Section V discusses the impact of our study on systems and autonomic computing research.

II. Service Level Objectives

Cloud services are networked, distributed workflows. Simple cloud services only retrieve and return data. Complex cloud services invoke simpler services and combine results; this is called a microservice architecture [24], [27]. In this context, request executions are the primary workload. Figure 2 highlights a common workflow: A web tier processes requests and manages data aggregation while databases, key-value stores, and distributed file systems store and retrieve data.

Cloud services can now link system events and their corresponding request executions. Linux Control Groups are widely used to monitor and allocate resources, such as CPU time, system memory and network bandwidth, among processes executing a request [14]. Such request tracking is the translation of prior systems research [26], [2]. Today's systems generalize and advance early prototypes. Monitoring produces log files local to the execution context. Offline scripts merge them and create global views, tracking complex and simple services. Figure 2 depicts a global view of monitored events during request executions. The events are sorted by arrival. Such request tracking provides response time and percentile data and enables SLO performance goals. Note, networking services and other heavily used components may sample response time, e.g., by measuring short round-trip protocols.

A. SLO Structure

Complex cloud services and simple microservices share common performance goals: *Achieve low response time for almost all requests.* SLO performance goals are structured to quantify these goals while allowing system managers to tailor details. Figure 2 depicts the core structure of SLO performance goals: (1) a response time limit is stipulated in absolute terms, (2) a percentile goal specifies the percentage of requests that must complete within the response time limit and (3) a reporting period defines how often log files are combined to assess whether objectives are met.

Cloud services have diverse response time expectations. Interactive cloud applications used by humans can allow several seconds of delay. In contrast, microservices accessed repeatedly during a single request execution allow tens to hundreds of millisecond delays.

SLO violations indicate response time or percentile goals were not satisfied during a reporting period. That is, log files reveal actual performance falls below goals. In contractual agreements where SLOs govern business partners, violations can trigger monetary charges. For example, Amazon Web Services refund clients if instances start and fail within 5 minutes [32], [13]. Cloud services that interact with humans can correlate profits and SLO violations. Walmart reports a 1% loss in profit with 10% degradation in response time [28]. Microservices also

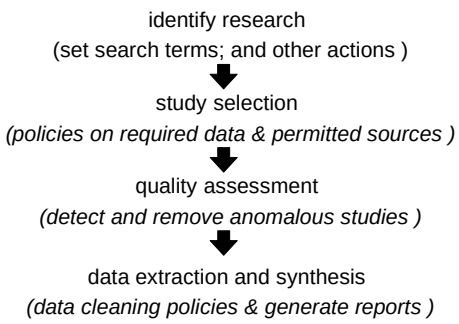

Fig. 3. Systematic literature review procedure

face consequences, because delays in simple services spread to complex services.

B. SLO Examples

The blue text in Figure 2 reports performance goals found through systematic literature review discussed in Section III. They reflect an industry case study, an SLO shared in a research paper and a widely used textbook. They are representative of the sources used in our study. We describe them here.

1. CXENSE Service Level Agreement [8]. Cxense provides software as a service for advertising, data analytics, and search. Their service level agreement covers all subscribers. 95% of all ad requests will achieve a response time of 300 milliseconds or less. Here, response time reflects time spent executing the request on Cxense servers or within their network (subscriber latency is excluded). It also excludes up to 90 minutes per month of downtime. Logs are aggregated monthly to determine violations. In Figure 2, this SLO is labeled *app*.

2. WorldCom/UUNET VPN [21]. This paper discusses SLOs for IP traffic managed by the WorldCom VPN service. 99.8% of VPN traffic should transfer between customer premise equipment (CPE), e.g., routers, in less than 120 ms. Transfer times are reported monthly. This SLO provided generous headroom for the VPN service provider, and the paper discusses approaches to make the SLO more strict. In Figure 2, this SLO is labeled *net*.

3. Site Reliability at Google [4]. This textbook explains the structure and underlying concerns for service level objectives. In the section Defining Objectives, the authors provide performance goals for RPC microservices. Figure 2 labels this SLO *web*.

III. Systematic Literature Review

Systematic literature review (SLR) helps researchers quantify data from qualitative, literary sources. It provides structure, bias mitigation, rigor and repeatable results [1]. Systematic reviews have well defined search strategies. Researchers can repeat strategies and compare alternative strategies to assess completeness [1]. Later studies can improve completeness. As described in Figure 3, SLR comprises 4 phases: search strategy, literature selection, quality assessment, and data extraction and synthesis. Below, we describe each phase in our study.

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.

A. Search Strategy

PICOC criteria (Population, Intervention, Comparison, Outcomes, Context) constrains search structure [25]. These concepts are independent qualifiers on search phrases. Combined, they form a search string used to retrieve documents, as shown in Equation 1.

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

The terms to the right of string represent different sets of phrases. A phrase was selected from each set and joined by OR/AND to construct the string. Population refers to the types of computer systems under review. Our study targeted interactive workloads, using phrases like cloud, networked systems, VPN service, web, databases, query systems, search engines, Internet services, etc. Intervention provides the types of SLO features sought. We used phrases like service level, response time, performance, percentile, percentage, latency, processing speed, etc. Outcome describes the target metrics from intervention, e.g., milliseconds, seconds, hours, 99^{th}, averaged, peak, monthly and yearly. Context is the types of documents used in our search such as primary studies and company handbooks. The final strings were used to search Google, Bing and digital libraries. An example search targeting VPN Services is shown in Equation 2. To obtain more SLO data, we do not put a limit on the content of each phrase as long as it is relevant in the range of this study and meets the phrase category requirement.

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

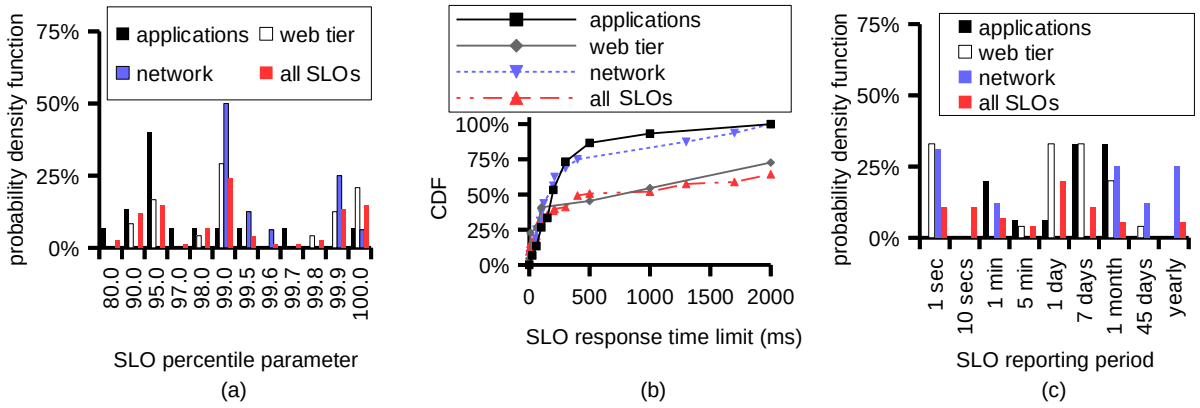

Fig. 4. Distributions of (a) SLO percentiles, (b) delay goals and (c) reporting periods. CDF stands for cummulative distribution function.

B. Literature Selection & Quality Assessment

The literature selection process gathered SLOs from primary studies and assessed the quality based on the SLO features provided. Recall, we allowed a wide range of valid search terms using the PICOC process. As a result, we manually examined 8,530 academic papers and 1,104 other documents. We did this in two phases. First, we examined headings, URLs and search results to remove documents that did not contain SLO details or supplied only vague descriptions. After the first phase, roughly 300 documents remained. In the second phase, we conducted a deep reading into the candidate documents and filtered them on authenticity and based on the SLO features they described. Namely, we excluded academic documents that did not reflect real-world SLOs, and we excluded documents that did not provide the minimum required SLO features. Table I shows a list of SLO features and indicates which features must be present in our search. Any SLO found without the required features were not considered. Only SLOs with a reporting period time, a latency time and a percentile requirement are included. For practical reasons, documents in languages other than English, and documents that reported SLO features from other studies were also excluded. Filtering resulted in 81 SLOs that met the minimum requirements from less than 300 candidate documents. The quality of these SLOs were assessed based on their workloads. All SLOs for workloads in the computer science and general software were included. SLOs for anomalous workloads like help desk calling service failed the quality assessment check. We finally extracted 75 out of the 81 SLO candidates.

C. Data Extraction and Synthesis

6 SLO features were recorded into a spreadsheet during the SLO extraction process. All the features in Table I except for *uptime* and *type of cloud service process* make up the 6 features. These features were found manually by reading through the document. In many cases, SLOs were not published with a similar naming scheme. Therefore, common regular expression tools offered little aid in this process. Figure 4, Figure 5 and Figure 6 reports the synthesized data for academic and industry sources. Further details on these figures are discussed in Section IV.

When SLOs report response time and percentile goals, their settings are normally clear. However, reporting period was often omitted or implicit. We looked for clues that explained how often data was compiled and/or what penalties were incurred on violations to infer SLO reporting period, when necessary. For this study, we did not consider objectives targeting average latency.

IV. Characterizing Realities and Myths

Several assumptions are commonly made about the performance goals defined by SLOs across academia and industry. Our SLR results give insight on which assumptions are realities or myths. Each SLO collected is accompanied by 6 SLO features. There are 3 key features we focus on to either refute or confirm these assumptions. Figure 4 depicts the distributions of these features as SLO percentile, delay time, and reporting period. The data is partitioned by workload type because it is difficult to compare SLOs across competing types of services where latencies and expectations can differ by orders of magnitude. The workloads are software applications, web service, and networking system. Figure 5 (b-c) illustrates the distributions of SLO delay and percentile grouped by industry and academic sources. Figure 5 (a) is a distribution of the sources by type. Figure 6 is a heat map of delays and percentiles from industry sources.

Reality 1: SLO delays and reporting periods differ greatly across workloads. Figure 4(b) shows that approximately 40% of SLOs across the 3 workload types define their response time limits in the range 0-100ms.

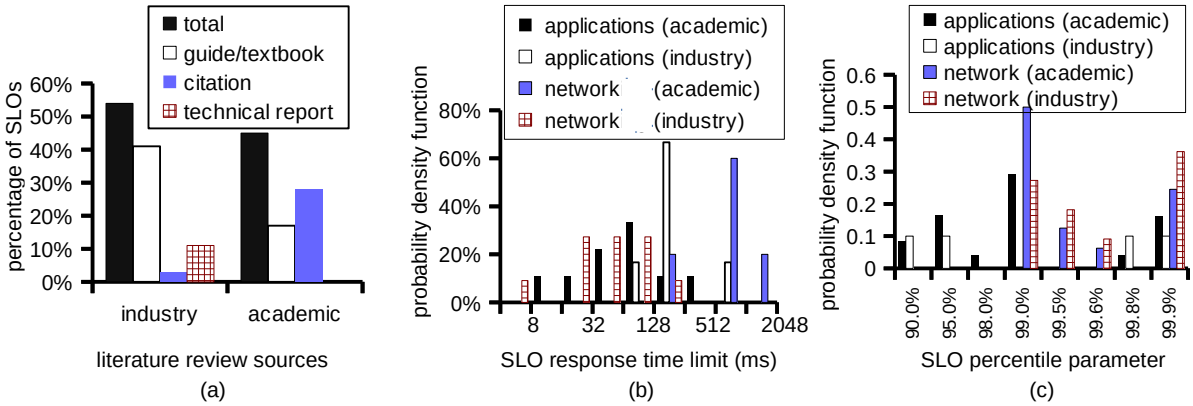

Fig. 5. (a) distribution of sources in our study, (b) distribution of SLO response time goals grouped by industry and academic sources, (c) distribution of SLO percentile goals grouped by source.

The other 60% of response time limits vary greatly across workloads. We can see network and software applications rarely set their SLOs greater than 500ms. Figure 4(c) also illustrates a wide range of reporting periods across workloads. 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: 90^{th}, 95^{th} and 99^{th} percentiles are widely used SLO percentage goals. Figure 4(a) illustrates the percent of time workloads should meet their performance goals. Software applications and web services have a wide range of percentiles varying from 80% to 99.9%. However, there are a concentrated number of SLOs at specific percentiles. Over 35% of SLOs for software applications use the 95^{th} percentile while the majority of web services use 99^{th}. A large fraction of SLOs for network services use the 99^{th} percentile as well, but its distribution differs from the other workloads'. Web services and software applications tend to have significant amount of SLOs with 90^{th} percentile.

Reality 3: Strict SLOs targeting single-digit delays are uncommon in practice. The SLOs tend to focus on the top right corner in Figure 6, which stands for a smaller SLO percentile and a larger SLO delay time. The majority of delays are between 100-1000ms 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. With increasingly more SLOs adopted in industry, single digit delays are still rare in practice.

Myth 1: 5–10 minute reporting periods. Consider microservices responding to requests in hundreds of microseconds. It is reasonable to expect SLO reporting periods of 5–10 minutes. This would capture enough samples for strict percentiles. Surprisingly, we found few SLOs with minute-granularity reporting periods. They consist of only less than 15% of our samples. Reporting periods in days and yearly length are most common, especially for complex services. Reporting periods of seconds are common for microservices and networking hardware.

Myth 2: Response time goals used in academic papers mirror goals set in practice. Figure 5(b) shows that SLO delays in academic sources differ sharply from industrial sources. On network serivces, academic sources are much more lenient. For whole applications, they are too strict. To be sure, we can not draw conclusions from our small sample— we simply can not corroborate a common assumption. We can report that both distributions are significantly different (p=0.9). While delays differ, we observe that the SLO percentiles in industry are mirrored by academia. We see no significant difference in the distributions.

V. Discussion

Our study examined SLOs available in public domain. Systematic literature review provided clear rules for inclusion and exclusion. These rules reduced our bias and simplified hard decisions on borderline cases. For example, web results from several large companies included SLOs with only availability and uptime goals. Even though we wanted to include industry sources, these results were excluded unless they also included response times goals. We were able to include SLOs that focused on availability only if they provided timeout thresholds for requests.

Our sample size is small and limits our ability to extrapolate. Instead of quantifying distributional parameters, we use our data to corroborate widely held perceptions. As business contracts, SLOs are simply uncommon

in the public domain. A larger sample would require proprietary data— perhaps from large surveys of system managers. Another weakness stems from the diverse sources used in our study. 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. Below, we share conclusions we have drawn from the study.

1. Translational evaluation for systems research. Systems research is translational, i.e., direct applications in practice are a salient feature. 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. This burdens industry practitioners and could slow down translations.

First, consider complex cloud services that combine data from multiple sources. Performance goals depend on microservices and suffer tail at scale effects where a few outliers inflate response time. These workloads are hard to configure for low response time. Instead, system managers tailor SLO percentiles based on their resource scheduling and provisioning policies [4]. Figure 5(c) plots SLO percentile parameters for whole applications. Multiple strict percentiles are represented from 90.0% to 99.9%. Research evaluation should follow suit, but often research papers evaluate only 1 set point [17], [33], [18], [12]. Janus et al. [16] are a positive example where research evaluation covers multiple percentiles.

On the other hand, microservices and infrastructure are designed for ultra-low latency. Competing setups offer different latency— for example, secure transport versus direct memory access— but deployed services *must* meet strict percentiles. Figures 5(b & c) show networking infrastructure SLOs in practice. Percentiles cluster but delay varies. Cake [30] is a positive example where research evaluation of infrastructure components considers multi-model latencies.

Open challenges for SLO performance goals. Response time limits below 10 milliseconds are challenging. Figure 6 reports one SLO with delay below 10 ms and it covers a networking infrastructure switch. Recent research discusses the killer micro-second, where many 10 us delays inflate response times by 10–40 ms [3]. Improvements in redesigning the system stack and modifying traditional low-level system optimization to be micro-second aware are needed.

Black swans prohibit extreme percentiles for complex software. Figure 6 are reports few SLOs with 99.8% and 99.5% percentile. 3 and 4 nine SLOs are challenging in practice because a few correlated problems can negate slack built into the SLO. Prior research has

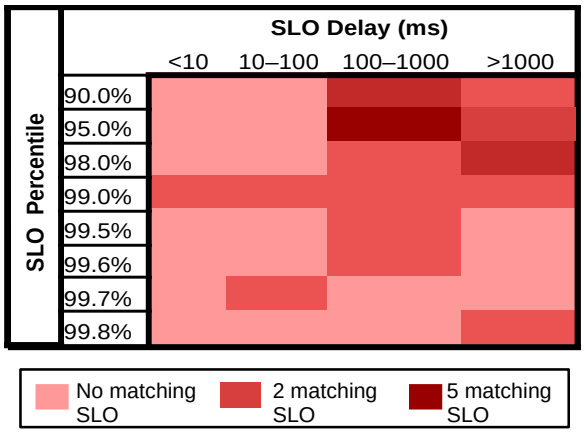

Fig. 6. Heat map of delay and percentile goals from industry sources.

shown it is possible to achieve extreme percentiles by inefficiently replicating resources [28]. However, cost-effective approaches remain elusive in practice. Computational sprinting has emerged as a possible solution [12], [22], [11]. Extreme percentiles cannot be evaluated over short reporting periods. Correlated events are rare and, in practice, system managers need slack in reporting periods to recover. Here, effective and representative time dilation is the challenge for researchers: How can a paper produced over 1 year study an SLO with a 1-year reporting period?

Our work studied SLOs as they are defined in practice, using absolute response time latency. However, response time comprises processing and waiting time. Processing time, in particular, affects SLO settings. Unfortunately, we did not have reliable access to processing time for the services studied. In future work, we would like to characterize the *slack* afforded between processing time and SLO latency settings.

Potpourri: This work was funded by NSF Grants 1749501 and 1350941. 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.

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

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

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

[26] 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.

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

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

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

[30] 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.

[31] 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.

[32] 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.

[33] N. Zacheilas and V. Kalogeraki. Chess: Cost-effective scheduling across multiple heterogeneous mapreduce clusters. In *ICAC*, 2016.

[34] W. Zhang, T. Wood, and J. Hwang. Netkv: Scalable, self-managing, load balancing as a network function. In *ICAC*, 2016.