INDICES: Exploiting Edge Resources for Performance-aware Cloud-hosted Services

Shashank Shekhar*, Ajay Dev Chhokra*, Anirban Bhattacharjee*, Guillaume Aupy†, Aniruddha Gokhale*
*Vanderbilt University, Nashville, TN 37235, USA and †INRIA & Université de Bordeaux, France
* Email: {shashank.shekhar,ajay.d.chhokra,anirban.bhattacharjee,a.gokhale}@vanderbilt.edu and †guillaume.aupy@inria.fr

***Abstract*—Despite the known benefits of hosting cloud-based services, the longer and often unpredictable end-to-end network latencies between the end user and the cloud can be detrimental to the response time requirements of the interactive cloud-hosted applications. Existing efforts that exploit edge/fog technology to migrate services closer to clients in order to improve response times do not fully resolve this problem as they do not focus on performance and interference issues at the migrated locations. This paper proposes INDICES framework that addresses these limitations by providing a novel solution that determines when and to which MDC a service should be migrated to and thus provides the desired performance. Empirical results validating our claims are presented using a setup comprising a centralized cloud and MDCs composed of heterogeneous hardware.**

***Index Terms*—Cloud Computing, Cloud latency, Micro Data Center, Cloudlet, Edge Computing, Fog Computing, Performance Interference, Resource Management.**

I. Introduction

Although hosting interactive applications in the cloud may be economically attractive and sometimes even necessary, particularly for applications such as cognitive assistance and online gaming, real-world experiments have shown that the latencies experienced by geographically distributed users of interactive services may be on the order of hundreds of milliseconds [1] which may not be acceptable to users [2]. For any cloud-hosted interactive application, the dominant factors that affect the round trip latencies are the network delay between the client and the cloud, particularly the round-trip delay between the nearest access point of the client and the cloud, and the time it takes to serve the client request. Thus, any improvement in round trip latencies must focus on reducing the network delays and the server processing time.

Fog computing in the form of cloudlets [3] or Micro Data Centers (MDCs) [4] have emerged as one of the key mechanisms to manage and bound the transit latency by supporting cloud-based services closer to the clients. MDCs[1] can be viewed as "a data center in a box," which act as the middle tier in the emerging "Edge–Fog–Cloud" hierarchy.

Most efforts to date in exploiting fog/edge resources have focused on cyber foraging, where tasks are offloaded from mobile devices to the cloud/MDCs for faster execution and to conserve resources on the mobile endpoints [5]. Whatever small number of prior efforts on moving tasks from the central clouds to the MDCs exists have either (a) not considered the resulting application performance because these efforts tend to overlook server overloading within MDCs. This in fact may worsen the user experience as compared to that of a traditional cloud-hosted interactive service, or (b) make very simplistic assumptions regarding their performance models.

In this paper, we address the performance-aware cloud to MDC service migration problem. Our approach accounts for a fundamental system property called performance interference, which otherwise is often overlooked. Performance interference is an inherent property of any virtualized system that is caused by co-located applications contending for resources thereby leading to performance degradation [6], [7]. Specifically, we focus on a "just-in-time and performance-aware" service migration approach for moving cloud-based interactive services hosted in the centralized cloud data center to a MDC.

A number of challenges including the heterogeneity in the hardware, and difficulty in measuring performance interferences and other system and network performance metrics must be overcome. We address these challenges in the context of providing a ubiquitous deployment approach that spans the cloud-edge spectrum. Our solution is called INDICES (**IN**telligent **D**eployment for ub**I**quitous **C**loud and **E**dge **S**ervices) framework, which codifies our algorithms for online performance monitoring, performance prediction, network performance measurements, server selection and application migration from the cloud to the fog. This paper focuses only on the key ideas and experimental validation; a more detailed explanation appears in [8].

The rest of the paper is organized as follows: Section II compares related work with our work; Section III provides details about INDICES including its underlying system model and assumptions, the problem formulation, and details of its implementation; Section IV presents empirical proof that validates our claims; and finally Section V presents concluding remarks alluding to lessons learned and future work.

II. Related Work

In this section we compare and contrast INDICES with related work along three dimensions: network latency-based server selection, performance interference-based server selection and performance-aware edge computing. Unlike our work, our survey has found that existing works seldom consider all dimensions holistically.

[1]In the rest of the paper, we will use the term MDC to represent all emerging mechanisms, such as Cloudlets, Micro Datacenters (MDCs), Locavore infrastructures, etc.

A. Network Latency-based Server Selection

DONAR [9] addresses the global replica selection problem using a decentralized, selection algorithm where the underlying protocol solves an optimization problem that takes into account client performance and server load. Dealer [10] targets geo-distributed, multi-tier and interactive applications to meet their stringent deadline constraints by monitoring individual component replicas and their communication latencies, and selects the combination that provides the best performance. Kwon et al. [11] applied network latency profiling and redundancy for cloud server selection while suggesting using cloudlets. We contend that these efforts consider simplistic models of server workload and their impact on performance, and do not cater to edge resource management.

B. Performance Interference-aware Server Selection

Paragon [6] identified the sources of interference that impact application performance and developed micro benchmarks for heterogeneous hardware. The system benchmarks applications and classifies them to find collocation patterns for scheduling. SMiTe [12] designed rulers for estimating sensitivity and degree of contention between applications when they are collocated. Bubble-Flux [13] assures QoS for latency-sensitive applications by dynamic interference profiling of shared hardware resources and collocating latency-sensitive applications with batch applications. These works, however, do not apply to virtualized data centers where the hypervisor places its own overhead on the resources and impacts performance.

Our prior work [7] designed a performance interference-aware resource management framework that benchmarks applications residing in virtual machines and applies a neural network-based regression mechanism that estimates a server's performance interference level. However, hardware heterogeneity and per application performance were not considered.

Heracles [14] mitigates performance interference issues for latency-sensitive applications by partitioning different shared resources. However, partitioning for resources, such as memory bandwidth is still not available, and moreover, cache partitioning is only available on newer hardware which cannot be applied to existing hardware.

C. Performance-aware Edge Computing

Fesehaye et al. [15] described a design to select between cloudlets and central cloud server for interactive mobile cloud applications based on the number of hops, mobility and latency. SEGUE [16] is an edge cloud migration decision system that applies state-based Markov Decision Process (MDP) model incorporating network and server states. Both the approaches have not been evaluated on real systems and the results are only simulation-based.

III. Problem Formulation and Design of INDICES

The INDICES framework is geared towards platform-as-a-service (PaaS) cloud providers, who seek to meet service level objectives (SLOs) of soft real-time applications such as online gaming, augmented reality, virtual desktop etc, by improving application response times. To that end they exploit micro data centers. INDICES must fulfill two primary responsibilities: determining which users are experiencing SLO violations, and making decisions to migrate the impacted application from CDC to apt MDC host.

A. INDICES Architecture and Application Model

Figure 1 depicts our architecture for INDICES that consists of a centralized data center CDC, owned by a PaaS cloud provider. The CDC is connected to a group of micro data centers (MDCs), $M = \{m_1, m_2, .., m_n\}$. These MDCs are deployed at the edge, and are either owned by the CDC provider or leased from an edge-based third party MDC provider. A leased MDC is assumed to be exclusively under the control of the that CDC provider.[2]

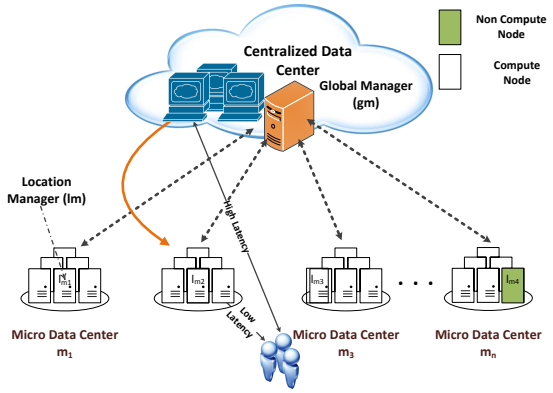

Fig. 1. Architectural Model of INDICES

The CDC contains a global manager gm, which is responsible for detecting and mitigating global SLO violations. We assume that for all $m \in M$, there exist links to the CDC with a backhaul bandwidth of b_m. Each MDC m comprises a set of compute servers, H_m, that can be allocated to the CDC for its operations at a specified cost. One of the hosts from H_m or a specially designated MDC host acts as the local manager (lm_m) for that MDC and is responsible for data collection, performance estimation, latency measurements and MDC-level decision making. This decision-making logic is deployed at the MDC by the CDC provider.

For this work, we consider a set $Apps$ of latency-sensitive applications that can be collaborative or single user and interactive or streaming in nature. Each application $a \in Apps$ is initially deployed in a CDC, with U_a number of users and is assumed to be containerized inside a virtual machine (VM). We assume that for a collaborative application a, its users are located in proximity of each other where they incur similar round trip latencies, e.g., in a collaborative educational application [17].

[2]Sharing of MDCs across different CDC providers is a dimension of our future work.

Each application a can be hosted on any active host in CDC, $\eta \in H$, where H is the set of all active hosts that provide virtualization using a hypervisor or virtual machine monitor (VMM). We let eed_a represent the expected execution duration for which the application will be used by the end-user clients. An interactive or streaming application comprises multiple individual interactions between the user and the application. Each interactive or streaming step of a is assumed to take an estimated execution time $eet_{a,\eta}$ on host η; for collaborative applications, it indicates the time needed for all users to have completed that step. Finally, for all users $u \in U_a$, let $el_{a,\eta,u}$ represent the estimated round-trip network latency and ϕ_a be the application-defined bounds on acceptable response time for each interactive step of the application.

Formally, the SLO for each application a hosted on host η can be characterized by:

$$eet_{a,\eta} + \max_{u \in U_a}(el_{a,\eta,u}) \leq \phi_a \tag{1}$$

Over time, a subset PA from the set of applications $Apps$ are identified by the system as suffering from performance degradation such that each application $p \in PA$ has a subset of one or more users, $U'_p \subseteq U_p$ experiencing SLO violations. SLO violation can be noticed either by the client-side instrumentation capability included with the "app" that the user installs or via a predictive capability used by the CDC based on user profile and location.

Our objective is thus to minimize the SLO violations, which is achieved by identifying and migrating application p to a MDC host $h \in H_m$ that will provide significantly improved performance. Since any application migration will involve state transfer, we assume that application p has the snapshot of current state which has to be transferred as part of the migration over the backhaul network from CDC to MDC m. Moreover, $ci_{p,h}$ is the initialization cost of the migrated application p on host h_m before the application can start processing requests on the MDC host. However, once the user-specific state has been transferred, there is minimal interaction between the CDC-based server and the MDC-based server for the remainder of the functioning of application p. For this paper we do not consider further consolidation of resources where applications migrate back to the CDC. The transfer cost $transfer_{p,h}$ incurred while transferring application p from CDC to host h of a MDC, and associated constraint are defined in the following equations:

$$transfer_{p,h} = \frac{s_p}{b_m} + ci_{p,h} \tag{2}$$

$$transfer_{p,h} \ll eed_p \tag{3}$$

where, b_m is the backhaul bandwidth, s_p is the size of the snapshot of the application's state, and eed_p is the remaining expected execution duration of application p's usage by the client. Equation 3 is a necessary condition for the motivation to use the edge and our solution to be relevant. To ensure that Equation 3 holds, we do not require transferring entire images of the VM and its containers. Instead, we use a layered file system architecture at the MDC that is pre-populated with base images used at the CDC. This assumption is realistic because we surmise that a MDC is either owned entirely or leased exclusively by a CDC provider. We also ensure Equation 3 holds by considering δ_p as a tolerance percentage value for the application user before (s)he starts to observe the improved response time:

$$transfer_{p,h}/eed_p \leq \delta_p \tag{4}$$

Finally, another critical issue we must account for is that any migration of a new application from CDC to a MDC should not violate the SLOs of existing applications in that MDC. To capture this aspect, let J_h represent the set of all applications currently running on a MDC host h, $eet_{j,h}$ be the estimated execution time for each application $j \in J_h$, which must be updated when we make a decision to migrate p to the same host, and $el_{j,h,u}$ be their corresponding measured round-trip network latency. These quantities must satisfy:

$$\forall j \in J_h, eet_{j,h} + \max_{u \in U_j}(el_{j,h,u}) \leq \phi_j \tag{5}$$

B. Performance Estimation: Problem and Challenges

The performance of an application depends on several factors including the workload, the hardware hosting platform, and co-located applications that cause performance interference [6], [7]. Below we describe their role in the performance estimation problem:

1) Workload Estimation: For the cloud-hosted interactive applications of interest to us, we assume that the workload variation is not significant within a single user session with the service. However, different sessions may have different workloads, for example, in an image processing application, the quality and hence the size of the captured and relayed image may vary for different client mobile devices. Thus, we consider each workload as a different application setting, which is reflected in the application response time.

2) Heterogeneity: Our CDC and MDCs consist of heterogeneous hardware and hence each application's performance can vary significantly from one hardware platform to another [6]. Therefore, we need an accurate benchmark of performance for each hardware platform.

3) Performance Interference: Although hypervisors/VMMs provide a high degree of security, fault, and environment isolation, the level of isolation is inadequate when it comes to performance isolation for the following reasons:

- *Presence of non-partitionable shared resources:* On-chip resources including cache spaces, cache and DRAM bandwidths, and interconnect networks are difficult to partition [18]. Although, Intel has introduced Cache Allocation Technology [19] to partition the last level cache (LLC), it is still not widely used and cannot be applied to older generation servers. The load imposed on these shared resources by one application is detrimental to all the cache- and memory-sensitive applications [20].

- *Resource overbooking:* Resource overbooking is common in cloud data centers, which precludes strict CPU reservations and can lead to lower level caches (L1 and L2) getting shared. Overbooking beyond the server capacity can lead to significant performance issues for the applications.

C. Cost Estimation and Objective Formulation

The objective of the framework is to assure the SLOs for all the identified applications $p \in PA$ while minimizing the overall deployment cost. Each MDC host h involves a monetary allocation cost as it is either leased or could be leased to other providers if owned by the centralized cloud. In addition, the running servers have operational costs, such as need for power and cooling. Thus, the provider wants to use as few MDC servers as possible and hence the deployment cost depends on the duration for which the MDC server is on. This cost $\widetilde{T}_h$ for deploying $p \in PA_h$ applications on host h is the extra duration for which the server has to be turned on and can be represented as:

$$\widetilde{T}_h = \begin{cases} 0, & \text{if } \max\limits_{p \in PA_h}(eed_p) < \max\limits_{j \in J_h}(eed_j), \\ \max\limits_{p \in PA_h}(eed_p) - \max\limits_{j \in J_h}(eed_j), & \text{otherwise} \end{cases} \tag{6}$$

We define a constant α_h denoting the cost of powering on the MDC server, and constant β_h denoting the cost for transferring the state to host h. Their values depend on the host h and its corresponding MDC. The cost for deployment on host h is thus defined as:

$$C(h) = \alpha_h * \widetilde{T}_h + \beta_h * \sum_{p \in PA_h} transfer_{p,h} \tag{7}$$

The optimization problem we solve for this research can then be formulated as:

$$\begin{aligned} \underset{h \in H}{\text{minimize}} & \sum C(h) \\ \text{subject to} \quad & eet_{p,h} + \max_{u \in U_p}(el_{p,h,u}) \leq \phi_p, \\ & \forall j \in J_h, eet_{j,h} + \max_{u \in U_j}(el_{j,h,u}), \\ & transfer_{p,h}/eed_p \leq \delta_p \end{aligned} \tag{8}$$

D. INDICES Design

Given the scale of the system, a centralized approach to performance prediction and cost estimation for every application hosted in the CDC/MDC and its clients is infeasible. Thus, we take a hierarchical approach where individual MDCs with their local managers and the global manager of the CDC participate in a two-level decision making as shown in Figure 1. The details of the architecture and the solution to the problem described above are available in [8].

IV. Experimental Validation

We now present results of evaluating INDICES in the context of a latency sensitive application.

A. Experimental Setup

Table I illustrates the hardware platforms and their counts used in our experiments. The CDC uses Openstack cloud OS version 12.0.2 where the guests receive their own public IP addresses. The MDC servers are managed directly by libvirt virtualization APIs and the guests communicate via port forwarding on the host. Each machine has Ubuntu 14.04.03 64-bit OS, QEMU-KVM hypervisor version 2.3.0 and libvirt version 1.2.16. Guests are configured with 2 GB memory, 10 GB disk, Ubuntu 14.04.03 64-bit OS and either 1 or 2 VCPUs. Since we are not concerned with VM migration within a CDC, we do not depict the CDC heterogeneity.

TABLE I
Server Architectures

Conf	Hardware Model	sockets/cores/ threads/GHz	L1/L2/L3 Cache (KB)	Mem Type/ MHz/GB	Count
A	i7 870	1/4/2/2.93	32/256/8192	DDR3/1333/16	2
B	Xeon W3530	1/4/2/2.8	32/256/8192	DDR3/1333/6	1
C	Core2Duo Q9550	1/4/1/2.83	32/6144/-	DDR2/800/8	1
D	Opteron 4170HE	2/6/1/2.1	64/512/5118	DDR3/1333/32	9

We use PARSEC and Splash-2 benchmarks [21] to generate the training data. To preclude profiling every new application on all the hardware, we need some training data. PARSEC targets Chip-Multiprocessors composing virtualized data centers, and provides a rich set of applications with different instruction mix, cache and memory utilization, needed for stressing different system subcomponents. We selected 20 tests from the benchmarks for data generation and validation. Due to lack of access to servers in different geographical regions, we used the network emulation tool, *netem*, and hierarchy token bucket based traffic control, *tc-htb*, for emulating the desired network latencies and bandwidth among the client, CDC and different MDCs.

B. Application Use Case

Our use case is an image processing application that performs feature detection, e.g., facial recognition in computer vision. We use the well-known Scale Invariant Feature Transform (SIFT) [22] to find the scale and rotation independent features. The client-side interface of the application continuously streams frames from a video or a web camera at a fixed rate of a frame per 200 milliseconds. The video resolution is 640x360 pixels and average frame size is 56 KB. The server comprises a Python-based application that receives frames over a TCP socket, processes it, and responds with the identified features along with the processing time. The client expects to receive a response within this duration, implying that 200 ms is the deadline for the application. Although our use case considers the performance for a single client connected to the cloud-hosted application, it can easily be extended to multiple clients residing in a similar latency region.

When the image processing application is submitted for hosting in our cloud, we execute it on different hardware

platforms in isolation to find its base execution times. For hardware platforms A, B, C, D defined in Table I, the base execution times, eet_a, were measured to be 86, 91, 146, 157 ms, respectively. Table II displays the emulated ping latency el_a from this client to CDC or different MDCs in the same region as the client. The table also lists their server composition, and the measured 95^{th} percentile network latency while sending TCP/IP and HTTP post requests of 56 KB size and receiving a response of size less than 1 KB. The expected duration for which the client needs to perform the image processing, eed_a, was set as 1 hour and the SLO was set to 95%.

TABLE II
CDC AND MDC SET UP FOR USE CASE (SECTION IV-B)

Conf	Distance	Ping Latency (±20%) ms	TCP latency(ms)	HTTP latency(ms)	Servers
1	1 hop	<1	2	6	1C + 1D
2	2 hops	5	14	28	1A + 2D
3	Multi hops	20	54	96	1B + 2D
4	Multi hops	30	76	142	1A + 3D
5	Central	50	127	220	1D

C. Evaluating the Performance Estimation Model

We first benchmarked our use case application on hardware platform D in order to develop its performance estimators. The threshold to discern applications with similar interference performance profile was set to 10% error. However, as illustrated in Figure 2, none of the existing applications met the criteria. Thus, we decided not to use any of the existing estimators for the use case application and benchmarked the application on all hardware configurations to develop its estimators. Figure 2 confirms that the estimation errors were high for all the hardware types requiring us to develop its estimators. We also found that the mean absolute percentage estimation error for our use case application to be less than 4% on all the platforms with low standard deviations as depicted in Figure 3. We can also account for this estimation error in our response time constraint (Equation 1) for stricter SLO adherence.

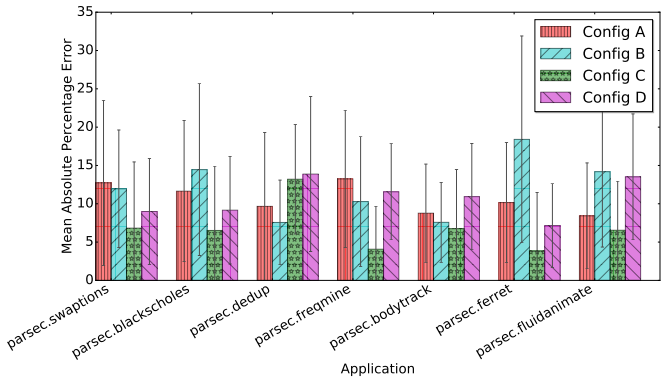

Fig. 2. Estimation of SIFT Profile Similarity with Parsec Benchmark

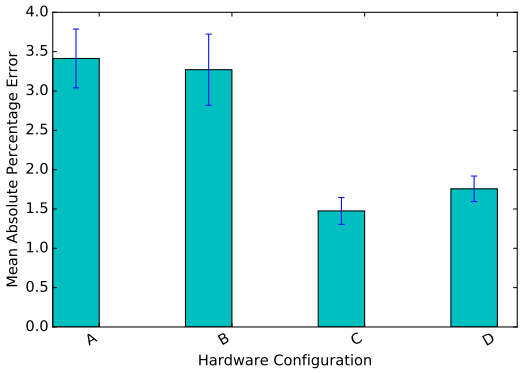

Fig. 3. SIFT Application Performance Estimation Error

D. Evaluating the Server Selection Algorithm

We compare our server selection algorithm results against two approaches: server selection algorithms based on minimum number of hops and least loaded server (among reachable MDCs). From Table II we observe that the minimum hop is 1. There are 2 servers in the minimum hop MDC 1 with hardware configuration types C and D. We create interference load on both the servers but ensured that the total load on the server does not exceed its capacity in terms of memory and vCPUs to eliminate unrealistic performance deteriorations. For the least-loaded server algorithm, we considered the server with least existing allocated resources, i.e. containing only a single VM. We did not consider a server with no existing load as it results in acquiring a new server and thus causes additional cost to the service provider. We found the server of hardware type D with MDC configuration 4 to be least loaded.

Applying SLO from Equation 1, INDICES found 2 servers of type A and D from MDC 2 and one server of type B from MDC 3 to be suitable for which we plot their response times for eed_a of one hour. Figure 4 displays the comparison of each of the suitable servers found by INDICES against the least loaded server. We observe that in this scenario, the least loaded server had 100% SLO violation because of network latency. However, the servers found by INDICES met their deadline 100%, 99.38% and 98.94%, respectively, which was well over the target SLO of 95%. Also, the minimum hop servers met the deadline only 66.64% and 60.64% of times due to performance interference shown in Figure 5.

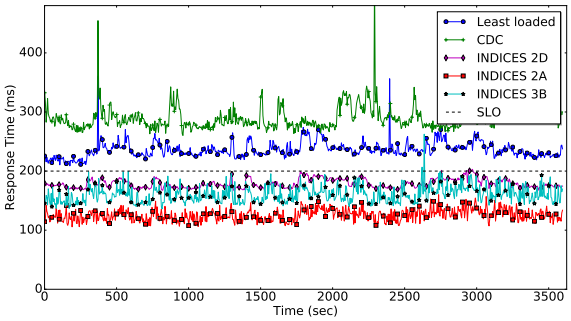

Fig. 4. INDICES vs Least Loaded

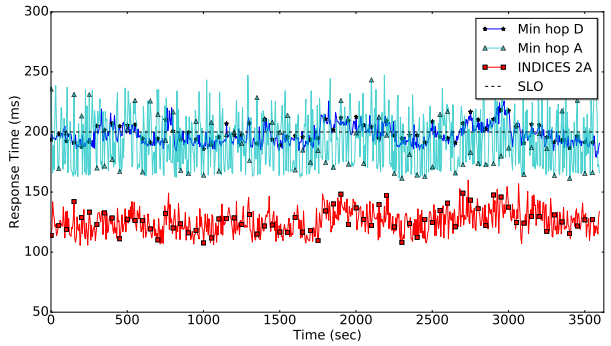

Fig. 5. INDICES vs Minimum Hop

V. Conclusions

This paper presents a deployment approach that exploits the available edge/fog resources in the form of micro data centers, which are used to migrate cloud-hosted applications closer to the clients so that their response times are improved. In doing so, our algorithm ensures that existing edge-deployed services are not unduly impacted in terms of their performance nor are the operational and management costs for the cloud provider overly affected. These objectives are met using an optimization problem, which is solved using a two-level cooperative and online process between system-level artifacts we have developed and deployed at both the micro data centers and centralized cloud data center. Our experimental results evaluating our framework called INDICES support our claims.

This work has provided deep insights that require further research. Some of these include the need for readily available benchmarks, better approaches to collecting measurements, workload consolidation across MDCs and CDCs, revenue generation and energy saving issues, and MDCs that are shared across different CDC providers. Going forward, we will also expand on our assumptions and limitations such as trust, security, workload variations and user mobility.

All scripts, source code, and experimental results for INDICES are available for download from https://github.com/shekharshank/indices.

Acknowledgments

This work is supported in part by the AFOSR DDDAS FA9550-13-1-0227 and NSF US Ignite CNS 1531079. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFOSR and NSF.

References

[1] Y. A. Wang, C. Huang, J. Li, and K. W. Ross, "Estimating the performance of hypothetical cloud service deployments: A measurement-based approach," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 2372–2380.

[2] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld, "Gaming in the clouds: Qoe and the users perspective," *Mathematical and Computer Modelling*, vol. 57, no. 11, pp. 2883–2894, 2013.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, 2009.

[4] V. Bahl, "Cloud 2020: Emergence of micro data centers (cloudlets) for latency sensitive computing (keynote)," *Middleware 2015*, 2015.

[5] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A context sensitive offloading scheme for mobile cloud computing service," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*. IEEE, 2015, pp. 869–876.

[6] C. Delimitrou and C. Kozyrakis, "Paragon: Qos-aware scheduling for heterogeneous datacenters," in *ACM SIGPLAN Notices*, vol. 48, no. 4. ACM, 2013, pp. 77–88.

[7] F. Caglar, S. Shekhar, A. Gokhale, and X. Koutsoukos, "An Intelligent, Performance Interference-aware Resource Management Scheme for IoT Cloud Backends," in *1st IEEE International Conference on Internet-of-Things: Design and Implementation*. Berlin, Germany: IEEE, Apr. 2016, pp. 95–105.

[8] S. Shekhar, A. Chhokra, A. Bhattacharjee, G. Aupy, and A. Gokhale, "INDICES: Exploiting Edge Resources for Performance-aware Cloud-hosted Services," Vanderbilt University, Institute for Software Integrated Systems, Nashville, TN, USA, Tech. Rep. ISIS-17-102, Feb. 2017. [Online]. Available: http://www.isis.vanderbilt.edu/sites/default/files/indices.pdf

[9] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "Donar: decentralized server selection for cloud services," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 231–242, 2010.

[10] M. Hajjat, D. Maltz, S. Rao, K. Sripanidkulchai *et al.*, "Dealer: application-aware request splitting for interactive cloud applications," in *8th international conference on Emerging networking experiments and technologies*. ACM, 2012, pp. 157–168.

[11] M. Kwon, Z. Dou, W. Heinzelman, T. Soyata, H. Ba, and J. Shi, "Use of network latency profiling and redundancy for cloud server selection," in *2014 IEEE 7th International Conference on Cloud Computing*. IEEE, 2014, pp. 826–832.

[12] Y. Zhang, M. A. Laurenzano, J. Mars, and L. Tang, "Smite: Precise qos prediction on real-system smt processors to improve utilization in warehouse scale computers," in *47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2014, pp. 406–418.

[13] H. Yang, A. Breslow, J. Mars, and L. Tang, "Bubble-flux: Precise online qos management for increased utilization in warehouse scale computers," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 607–618.

[14] D. Lo, L. Cheng, R. Govindaraju, P. Ranganathan, and C. Kozyrakis, "Improving resource efficiency at scale with heracles," *ACM Transactions on Computer Systems (TOCS)*, vol. 34, no. 2, p. 6, 2016.

[15] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang, "Impact of cloudlets on interactive mobile cloud applications," in *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International*. IEEE, 2012, pp. 123–132.

[16] W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "Segue: Quality of service aware edge cloud service migration," in *8th IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2016.

[17] F. Caglar, S. Shekhar, A. Gokhale, S. Basu, T. Rafi, J. Kinnebrew, and G. Biswas, "Cloud-hosted Simulation-as-a-Service for High School STEM Education," *Elsevier Simulation Modelling Practice and Theory: Special Issue on Cloud Simulation*, vol. 58, no. 2, pp. 255–273, Nov. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.simpat.2015.06.006

[18] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 22.

[19] "Cache allocation technology improves real-time performance," http://www.intel.com/content/www/us/en/communications/cache-allocation-technology-white-paper.html.

[20] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa, "Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations," in *44th annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2011, pp. 248–259.

[21] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: characterization and architectural implications," in *17th international conference on Parallel architectures and compilation techniques*. ACM, 2008, pp. 72–81.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.