Osmotic Computing: A New Paradigm for Edge/Cloud Integration

Massimo Villari*, Maria Fazio*, Schahram Dustdar†, Omer Rana‡, Rajiv Ranjan§

*Università degli Studi di Messina, Italy
†Technical University of Vienna, Austria
‡Cardiff University, United Kingdom
§Newcastle University, United Kingdom

I. Introduction

With the promise of potentially unlimited power and scalability, Cloud computing (especially IaaS) supports the deployment of reliable services across a number of different application domains. In the domain of Internet of Things (IoT), Cloud solutions can improve the quality of services moving towards new business opportunities, such as location-based content delivery, data provisioning in distributed multi-tenant environment, multi-tier enterprise applications. The success of Cloud-centric IoT programming models and resource orchestration techniques is proven by available mature solutions, such as Amazon IoT and Google Cloud Dataflow. However, recent technological advances have disrupted the current centralized Cloud computing model, moving Cloud resources close to end users. This evolution is mainly required for the adaptation of the Cloud paradigm to the IoT phenomenon. The increasing need for supporting interaction between IoT and Cloud computing systems has also led to the creation of the Edge Computing model, which aims to provide processing and storage capacity as an extension of available IoT devices, without the need to move data/processing to a central datacenter. This reduces communication delays and the overall size of data that needs to be migrated across the Internet and public/private datacenters.

We discuss benefits and challenges in orchestrating Cloud and Edge resources according to a new paradigm which we refer to as "Osmotic Computing". This paradigm is driven by the significant increase in resource capacity/capability at the network edge, along with support for data transfer protocols that enable such resources to interact more seamlessly with data center-based services. The approach is aimed at highly distributed and federated environments, and enables the automatic deployment of microservices that are composed and interconnected over both Edge and Cloud infrastructures. Borrowed from chemistry, "osmosis" represents the seamless diffusion of molecules from a higher to a lower concentration solution, which we believe should represent how services can be migrated across data centers to the network edge. Hence, Osmotic Computing implies the dynamic management of services and microservices across Cloud and Edge datacenters, addressing issues related to deployment, networking and security, thus to provide reliable IoT support with specified levels of Quality of Service (QoS). Osmotic Computing inherits challenges and issues related to elasticity in Cloud datacenters, but adds several proper features due to the heterogeneous nature of Edge microdatacenters and Cloud infrastructures. Moreover, the reference scenario includes different stakeholders (Cloud providers, Edge providers, IoT providers, Application providers,etc) that contribute to the provisioning of IoT service and applications in a federated environment. In the following sections, we provide a clear description of the basic principles of Osmotic Computing, highlighting research and business opportunities arising from such a new computing paradigm.

II. Motivations

Current Cloud computing programming models and resource orchestration techniques are immensely challenged by the recent evolution of the IoT phenomenon. IoT comprises many billions of Internet Connected Devices or Things where devices (sensors, actuators, mobile phones, routers, gateways, switches) monitor cyber and physical world. While the IoT is seen as the means for connecting disparate sensing and actuation devices (via the Internet) with applications and services, Cloud computing offers computation and storage capabilities required by IoT applications and services. This complex fusion of IoT devices and Cloud resources has led to creation of new Edge applications in the domains of supply chain management, transport, monitoring of built environment and energy grids. These diverse applications are expected to exponentially increase the number of IoT devices connected to Clouds. For example, CISCO estimates that currently 15 billion devices are connected to the Internet, and this number is set to explode to 50 billion by 2020. In another estimate, researchers at HP Labs estimate that by 2030, there will be 1 trillion sensors. Such an explosion in IoT device ecosystem will lead to production of data at the Edge in increasingly significant volumes and also at very high velocity.

Despite an increase in availability of IoT devices, Cloud-centric IoT programming models (e.g., Amazon IoT and Google Cloud Dataflow) and resource orchestration techniques are inappropriate in the context of emerging Edge applications for the principal reason that they assume that the intelligence and resource capacity necessary for data processing reside

predominantly in the Cloud datacenter. Thus, to implement complex IoT-oriented computing systems, both Cloud and Edge resources should be exploited setting up a hybrid virtual infrastructure, as shown in Figure 1. Cloud and Edge datacenters will me managed in a federated environment, where different Providers share their resources for IoT services and application support.

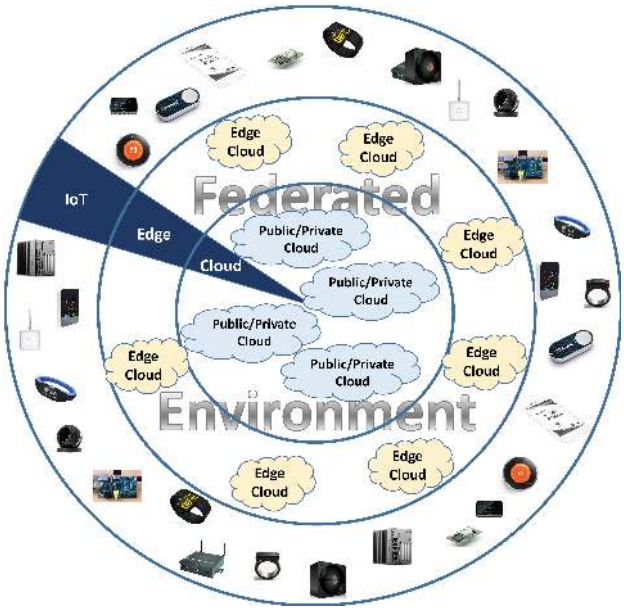

Fig. 1: Edge and Cloud Computing for IoT.

The burden of data upload towards datacenters leads to inefficient use of communication bandwidth and energy consumption, and a recent study by CISCO (http://goo.gl/M09Ucj) shows that total datacenter traffic will triple by 2019, worsening the situation further. To save network bandwidth store-and process later approaches can not be adopted, because this undermines real-time decision making, which is often a necessary requirement behind IoT. On the contrary, Edge computing aims to lay computing needs on the resource constrained Edge devices, as shown in Figure 1. Edge applications are highly time-sensitive (e.g., hazard warning application to help communities with prediction and preparation for environmental conditions such as storms, landslides, and flooding) because they perform immediate analysis of, or response to, collected sensing data. However, even if Cloud-based programming models cannot support the desired degree of sensitivity for IoT applications, they can strongly increase computation and storage availability whenever necessary. As a result, the prevailing Cloud-centric IoT programming model needs to be revised into something that is more adaptable and decentralized to meet the needs of emerging IoT applications.

III. Osmotic Computing

Osmotic Computing aims to decompose applications into microservices and perform a dynamic tailoring of microservices in smart environments exploiting resources in Edge and Cloud infrastructures. Application delivery follows an osmotic

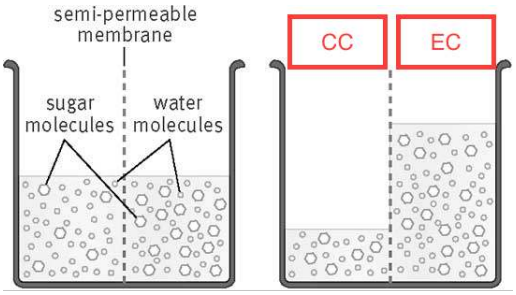

Fig. 2: Basic Concepts of Osmotic Computing

behavior where microservices in containers are deployed opportunistically in Cloud and Edge systems. Like the movement of solvent molecules through a semi-permeable membrane into a region of higher solute concentration to equalize the solute concentrations on the two sides of the membrane i.e., *osmosis* (in the context of chemistry), in Osmotic Computing the dynamic management of resources into Cloud and Edge datacenters evolves towards the balanced deployment of microservices satisfying well defined low-level constrains and high-level needs, as shown in Figure 2. However, differently from the chemical osmotic process, Osmotic Computing allows a tunable configuration of the resource involvement, following resource availability and application requirements (see Figure 3). This is an important distinction, i.e. how the difference in configuration (very much infrastructure and application dependent) is set up to determine whether microservices should migrate from Cloud to Edge or vice versa.

The Osmotic Computing goes beyond the simple elastic management of deployed resources, because deployment strategies are related to requirements of both infrastructure (e.g., load balancing, reliability, availability) and applications (e.g., sensing/actuation capabilities, context awareness, proximity, QoS) requirements, and they can also change during the time. Due to the high heterogeneity of physical resources, the microservice deployment task needs to adapt the virtual environment to the involved hardware equipment. Thus, a bidirectional flow of adapted microservices from Cloud to Edge (and vice versa) has to be managed. Moreover, the migration of microservices in the Edge/Cloud system implies the need of a dynamic and efficient management of virtual network issues, in order to avoid application breakdown or degradation of QoS.

A breakthrough approach to address the above issues is to decouple the management of user data and applications from the management of networking and security services. Osmotic Computing aims to move in this direction, providing a flexible infrastructure, by providing an automatic and secure microservice deployment solution. Specifically, Osmotic Computing is based on an innovative application-agnostic approach, exploiting lightweight container-based (e.g., Docker, Kubernetes) virtualization technologies, for the deployment of microservices in heterogeneous Edge and Cloud datacenters.

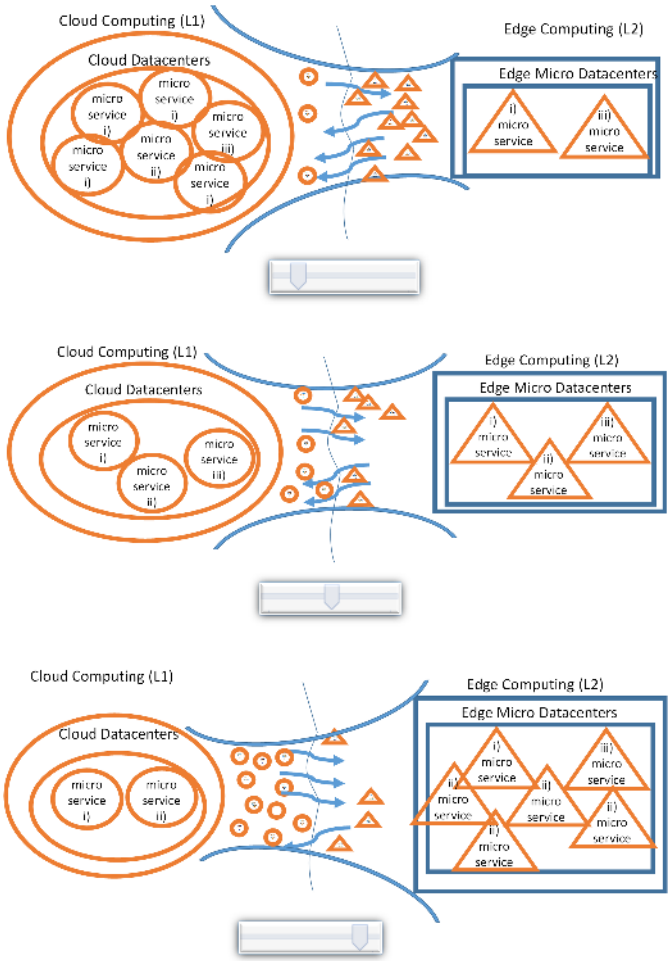

Fig. 3: Osmotic Computing Osmotic Computing in Cloud and Edge Datacenters

IV. Osmotic ecosystem

Figure 4 illustrates some of the key concepts in Osmotic Computing with reference to service deployment. Osmotic Computing spans two main infrastructure layers. The L1 layer consists of Cloud datacenters, where several types of service and microservice are provided. For the Osmotic Computing purposes, at this layer, microservices are composed according to end users high-level requirements. The L2 layer identifies the Edge computing environment, that includes data capture points and gateway nodes, able to undertake operations (average, min, max, filtering, aggregation etc.) on local data. These devices capture data with a pre-defined frequency (often dictated by the rate of change of the phenomenon being observed), depending on the capacity of the device to record/collect data and also based on specific system requirements that need to be satisfied. Devices at L2 can carry out various more advanced operations on the raw data collected in the environment, such as encryption of an incoming data stream or encoding/transcoding operations before forwarding this data for subsequent analysis to L1. Due to different properties of systems at L1 and L2, we envision a distributed heterogeneous Cloud composed of different types of resources located at each of the two layers. Understanding how a microservice hosted on a Cloud at L1 can interact and coordinate with a microservice in L2 is a key research challenge in such systems. Each level has its own objective functionalities which influences the types of operations carried out. For instance, L2 generally consists of resource constrained devices (i.e., limited battery power, network range, etc.) and network elements, which must carry out tasks without overloading available resources.

Datacenters at L1 and micro-datacenters at L2 can belong to different providers. However, according to a federated scenario, providers can establish relationships and cooperate to share resources and services, thus increasing their business opportunities [1], [2]. In this scenario an Osmotic Computing framework is application agnostic, offering user applications with run-time environment working in a distributed and secure way. Thus, the main types of microservices that the Osmotic Computing framework has to orchestrate and deploy into Cloud and Edge infrastructue are: 1) general purpose microservices (MS), that are strictly related to the specific applicative goal; 2) microservices for network management (MS Net) for setting up virtual networks among microservices deployed in the distributed and federated Cloud/Edge system; 3) microservices for security management (MS Sec), to support cross-platform development of security-enabled microservices.

The microservice provisioning solution can benefit from aggregating different types of resources in the L1 and L2 deployment environments. Understanding how these systems could be aggregated to support particular application requirements (particularly non-functional requirements, such as latency, throughput, security, budget, etc.) remains an important challenge. In particular, the proposed solution follows an advanced approach where microservices are opportunistically deployed in virtual components, called ”containers”. Container-based virtualization technologies (e.g., LXC, Docker, PXE, Google Container and Amazon Compute Cloud Container) have emerged as a lightweight alternative to hypervisor-based approaches (e.g., Xen, Microsoft Hyper-V) used in the Cloud. A container permits only well-defined software components (e.g., database server) to be encapsulated, which leads to significant reduction of deployment overhead and much higher instance density on a single device as compared to a hypervisor. Hence, the new container-based approaches permit deployment of lightweight microservices on resource constrained and programmable smart devices on the network Edge such as gateways (Raspberry Pi, Arduino), network switches (HP OpenFlow) and routers (e.g., CISCO IOx), but also to increase performance in the dynamic management of microservices in Cloud datacenters.

The Osmotic Computing attempts to characterize how:

- composed microservices must be automatically adapted to the deployment sites, considering location and context of deployment, since containers are strictly related to the capabilities of the physical host.
- a decision maker has to map microservices to the relevant location. Such a decision would be influenced by

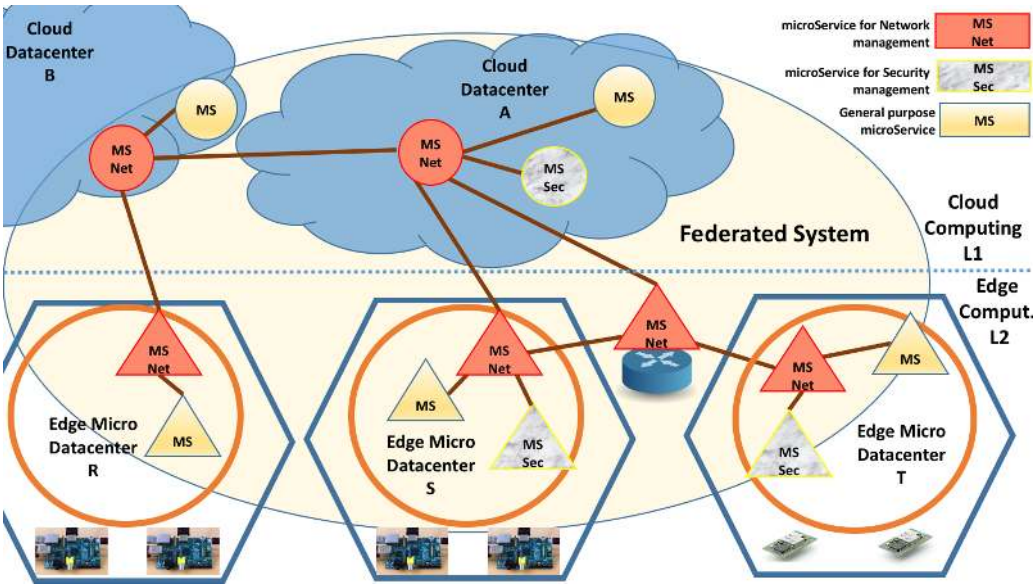

Fig. 4: A two Layer (L1/L2) Federated Cloud Environment in Osmotic Computing .

constraints identified by the specific application and the infrastructure provider, such as utilization of specialist resources (e.g., a GPU cluster), improving revenue or reducing management overheads (e.g., system administration and/or energy costs).

- adaptation of microservices to fluctuations in the computing environment must be performed over time – during the execution of microservices. It implies that a feedback driven orchestration is necessary to detect changes in infrastructure performance and QoS metrics.

V. Research directions

To make most effective use of the Osmotic Computing paradigm, the following research directions are proposed.

A. Runtime Microservice Deployment

Osmotic Computing can benefit from the extension of virtualization capability to IoT devices. Recently, research activities in Cloud based solutions for IoT and Edge devices presented container-based virtualization as an alternative to virtual machines in the Cloud [3]. Such virtualization approach can be usefully adopted in IoT scenarios. For example, Docker [4] provides container-based virtualization, initially based on Linux Containers (LXC), that can completely encapsulate an application and its dependencies within a virtual container, Docker Swarm [5] provides a native orchestration framework for multi Docker deployments, and Kubernetes [6] is an open-source system for automating deployment, operations, and management of containerized applications.

An Osmotic Computing framework for the deployment of services in a PaaS is DRACO [7]. It extends the Von Neumann stored program control (SPC) computing model to create self-configuring, self-monitoring, self-healing, self-protecting and self-optimizing (self-managing or self-*) distributed software systems. As opposed to self-organizing systems that evolve based on probabilistic considerations, this approach focuses on the encapsulation, replication, and execution of distributed and managed tasks that are precisely specified.

An Osmotic Computing framework should provide a microservice Engine, allowing users and developers to deploy containers running microservices on IoT and Edge devices, enabling microservice execution and deployment. The innovation deliveded by Osmotic Computing will facilitate the creation of a market of virtual IoT based applications. Software adaptation and versioning mechanisms will allow Edge Cloud providers to deploy microservices consisting of a heterogeneous pool of physical devices. Benefits of Osmotic Computing include deployment of distributed IoT oriented microservices, software consolidation, and service optimization.

B. Microservice configuration

Branded price calculators are available from public Cloud providers (Amazon [8], Azure [9]) and academic projects (Cloudrado [10], CloudRecommender [11]), which allow comparison of Cloud datacenter resource leasing costs. Existing work in the Cloud datacenter context supports provider evaluation methods but lacks microservice and Edge datacenter configuration support. Multiple approaches have applied optimization [12] and performance measurement techniques [13] for selecting Cloud datacenter resources for deploying VM images according to QoS criteria (throughput, availability, cost, reputation, etc.). While doing so, they have largely ignored the need for VM images, a migration process with transparent decision support and adaptability to custom criteria, and, hence, lack flexibility. However, the configurations and QoS criteria for selecting and ranking microservices and datacenter resources on the network Edge differ from VM deployment on Cloud data centers.

In Osmotic Computing developing holistic decision-making frameworks that automate configuration selection across mi-

croservices and resources in Cloud and Edge datacenters to meet QoS constraints is necessary. To this end, novel decision-making techniques based on multi-criteria optimization (e.g., Genetic Algorithms) and multi-criteria decision making (e.g., Analytic Network Process) techniques should be investigated.

C. Microservice Networking

Both Software Defined Networks (SDN) and Network Function Virtualization (NFV)[14], offer useful solutions for supporting in-network/in-transit processing of data (between Edge/Data Centre) and providing network management abstraction independent of the underlying technology. Osmotic Computing would benefit from both of these, to enable more seamless data exchange between the data centre and network edge.

OpenStack through the Neutron component allows to setup intra-domain VLANs using Open Virtual Network (OVN [15]). OVS is a technology that provides a logical network abstraction on top of a physical network. OVN in particular provides Layer 2/Layer 3 virtual networking, to define logical switches and routers giving the possibility to setup security groups, Access Control Lists, multiple tunnel overlays (Geneve, STT, and VXLAN), TOR-based and software-based logical-physical gateways. OVN works with Linux (KVM and Xen), containers and the Data Plane Development Kit (DPDK [16]). DPDK, creating an Environment Abstraction Layer (EAL). OVN is becoming the de-facto standard as a set of data plane libraries and network interface controller drivers for fast packet processing. However, OVN fails in creating inter-domain and federated networks.

Osmotic Computing is based on an abstraction of networks that spawn from Cloud to Edge and vice versa for improving the performance of the communication among microservices. The network here represents an enabler that allows us to dynamically adjust the overall microservices behavior according to user requirements. The network management advances in Osmotic Computing should include the development of an interoperability layer for remote orchestration of heterogeneous Edge devices, for example, exploiting Software Defined Networking (SDN) and Network Function Virtualization (NFV) capabilities, accessible through an API. Moreover, the characterization of federated networks in the domain of Cloud and Edge is a concept totally missing in the scientific literature. In Osmotic Computing a specific metadata ontology for overcoming this issue should be assessed.

D. Microservice Security

Significant literature exists on supporting security within Cloud and IoT systems, but a Cloud-Centric approach is often used [17],[18]. The challenges of integrating IoT devices with Cloud systems (coining the term “Cloud of Things”) are outlined in [19]. The authors discuss a business model for such an architecture as well as the limitations and issues related to the security of IoT devices. Bui investigates the security level of Docker in [20] by considering two main areas: (1) the internal security of Docker, and (2) how Docker interacts with the security features of the Linux kernel, such as SELinux and AppArmor, in order to harden the host system. The proposed analysis shows that Docker provides a high level of isolation and restricts access to physical resources used for containers using namespaces, cgroups, and its copy-on-write file system, even with the default configuration. An Osmotic Computing framework needs a coherent security policy that must be supported within both a Cloud data center and an edge computing environment, to enable microservice execution and migration. Ensuring that the same security considerations are observed, for a particular microservice, across both of these environments remains a challenge.

Such security features will enable self-identification processes that will make the deployment of microservices inside Cloud and Edge devices easier and more secure, also facilitating the wide adoption of the Osmotic Computing technology. In addition, another objective of Osmotic Computing is to add security capabilities to the container engine in order to enable the secure deployment of containers including microservices on IoT devices. More specifically, an Osmotic Computing framework should allow developers to build chains of trust involving both Edge devices and Cloud systems by means of a transversal security.

E. Edge Computing

Recent efforts in creating an open source “IoTCloud” (providing sensors-as-a-service) and middleware oriented efforts in the European Open IoT project indicates significant interest in this area from the academic community. In the same context, HTTP/REST-based APIs, such as Xively, Open Sen.se, Think Speak etc. indicate strong commercial interest, in applications ranging from smart cities to intelligent homes. This also aligns with the *Fog Computing* efforts involving Cloudlets (from Cisco), which involve ”small Clouds” which are geographically scattered across a network and act as ”small datacenters” at the Edge of the network [21]. Cloudlets aim to give support to IoT devices by providing increased processing and storage capacity as an extension of those devices, but without the need to move data/processing to a central datacenter [22], [23]. This leads to reduced communication delays and the overall size of data that needs to be migrated to a datacenter. Osmotic Computing is not an alternative to such efforts, instead it focuses on seamless transfer/migration and execution of microservices across cloudlets and data centers.

The related approach [24] of “mobile offloading” is centered on the need to off-load complex and long running tasks from mobile devices to Cloud-based datacenters. To reduce potential battery power consumption and potential application delay due to intermittent network connectivity, tasks from mobile devices (which are considered to have lower computation and storage capabilities compared to a datacenter) are executed at a datacenter, with periodic synchronization between the Edge device and the datacenter. An alternative approach (to achieve the same outcome) involves creating a mobile device ”clone” within a datacenter as a virtual machine, examples include CloneCloud and Moitree.

Our approach suggests the need to combine "mobile offloading" with "data centre offloading" i.e., we off-load computation initially carried out within a datacenter to a mobile device. This "reverse" off-loading enables computation to be undertaken closer to the phenomenon being measured (overcoming latency and data transfer costs). The Osmotic Computing approach is therefore focused on understanding the types of microservices which would be more relevant to execute at the Edge, rather than within a datacenter environment, and vice versa.

F. Microservice workload contention and interference evaluation

The co-deployed microservices on Cloud or Edge datacenters can lead to contention problems which will affect QoS. During deployment of microservices, orchestration techniques must consider which microservices should be combined on a datacenter resource, to minimize resource contention due to workload interference. Workload resource consumption and QoS are not additive, so understanding the nature of their composition is critical to deciding which microservices can be deployed together. Recent work has investigated several approaches to minimize the impact of workload interference on the QoS of hosted applications on Cloud datacenters.

Hardware-based approaches add complexity to the processor architecture and are difficult to manage over time. Govindan et al. [25] developed a scheme to quantify the effects of cache contention between consolidated workloads. However, the aforementioned techniques focus on contention issues of only one hardware resource type (i.e. cache) while ignoring others. Nathuji et al. [26] present a control theory-based approach to consolidation that mitigates the effects of cache, memory, and hardware pre-fetching contention of co-existing workloads. However they focus on only CPU-bound or compute-intensive applications. To the best of our knowledge, none of the existing academic approaches nor the container orchestration frameworks such as OpenShift Origin, Amazon EC2 Container Service, and Kubernetes can automatically detect and handle resource contentions among co-deployed microservices across Cloud and Edge datacenter resources. Software-based technique that learns microservice contention via simulation and benchmarking, should be investigated, then apply that learned knowledge in real time to detect contention between microservices via active monitoring and prediction. Hence, research in Osmotic Computing should be focus on novel microservice consolidation techniques that can dynamically detect and resolve resource contention via microservice performance characterization, workload prioritization and coordinated deployment.

G. Monitoring

Monitoring tools that were popular in the grid and cluster computing era included R-GMA, Hawkeye, Network Weather Service (NWS), and Monitoring and Directory Service (MDS). These tools were concerned only with monitoring QoS metrics at the hardware resource-level (CPU percentage, TCP/IP performance, available non-paged memory) not application-level QoS metrics (e.g., end (network Edge)-to-end (Cloud datacenter) request processing latency). Cluster-wide monitoring frameworks (Nagios, Ganglia, Apache Hadoop, Apache Spark) provide information about hardware metrics (cluster utilization, CPU utilization, memory utilization and nature of application: disk-, network-, or CPU-bound) of cluster resources that may belong to public or private Cloud datacenter. Monitoring frameworks used by Amazon Container Service (Amazon CloudWatch) and Kubernetes (Heapster) typically monitor CPU, memory, filesystem, and network usage statistics, so they are not able to monitor microservice-level QoS metrics (query processing latency of database microserver, throughput of data compression microserver, etc.). To the best of our knowledge, none of the approaches in literature can (i) monitor and instrument data (workload input and QoS metrics, disruptive event) across microservices, Cloud datacenter, in-transit network, and Edge datacenter or (ii) detect root causes of QoS violations and failures across the infrastructure based on workload and QoS metrics logs. Researchers should investigate scalable methods (based on self-balanced trees) to monitor QoS and security metrics across multiple-levels of Osmotic Computing including microservices, Cloud datacenters and Edge micro-datacenters.

H. Microservice orchestration and elasticity control

The run-time orchestration of microservices in a scalable Edge/Cloud system is a complex research problem due to the difficult to estimate microservice workload behavior in terms of data volume to be analysed, data arrival rate, query types, data processing time distributions, query processing time distributions, I/O system behavior, and number of users connecting to different types and mix of microservices. Without knowing the workload behaviors of microservices, it is difficult to make decisions about the types and scale of Cloud and Edge datacenter resources to be provisioned to microservices at any given time. Kubernetes [6] and OpenShift Origin [27] offers a microservice container reconfiguration feature, which scales by observing CPU usage ("scaling is agnostic to the workload behavior and QoS targets of a microservice"). Amazon's autoscaling service [28] employs simple threshold based rules or scheduled actions based on a timetable to regulate infrastructural resources (e.g., if the average CPU usage is above 40%, add an additional microservice container).

In Osmotic Computing , the traditional notion of run-time control and reconfiguration which only considers resources hosted in Cloud datacentes, to resources that are deployed and available at the Edge, should be extended. Machine learning techniques for developing predictive models to forecast workload input and performance metrics across multiple, co-located microservices on Cloud and Edge datacenter resources should be investigated. Additionally, intelligent, QoS-aware, and contention-aware resource orchestration algorithms should be developed based on the above models, monitoring systems, and configuration selection techniques.

VI. Conclusions

The paper presented Osmotic Computing as a new paradigm for orchestrating resource in IoT, Edge and Cloud systems. Abstraction of microservices along with the use of a container-based approach allows deployment of new advanced services on heterogeneous infrastructures. Providing such services for IoT-based devices aims at increasing the capabilities and functionalities of existing Clouds and Edge systems. Therfore, Osmotic Computing enables microservices and resource orchestration mechanisms to be carried out over distributed Clouds. The seamless migration of services which can adapt their behaviour with resource properties, remains an important challenge. Whereas significant emphasis has been placed on (mobile) cloud off-loading (whereby software applications can be off-loaded from a mobile device to a data centre), we believe there is also the need for reverse off-loading – i.e. movement of functionality from the cloud to the edge devices, to counter for latency-sensitive applications and to mimise data sizes that must be transferred over a network. We believe Osmotic computing provides a useful basis for providing a unifying paradigm for this.

References

[1] M. Giacobbe, A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "Towards energy management in cloud federation: A survey in the perspective of future sustainable and cost-saving strategies," *Computer Networks*, vol. 91, pp. 438 – 452, 2015.

[2] A. Celesti, M. Fazio, M. Giacobbe, A. Puliafito, and M. Villari, "Characterizing cloud federation in iot," in *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 93–98, March 2016.

[3] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*,, pp. 171–172, March 2015.

[4] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, no. 3, pp. 81–84, 2014.

[5] Docker Swarm, https://docs.docker.com/swarm.

[6] Kubernetes - Horizontal Pod Autoscaler, http://kubernetes.io/v1.1/docs/user-guide/horizontal-pod-autoscaler.html.

[7] A. Celesti, N. Peditto, F. Verboso, M. Villari, and A. Puliafito, "Draco paas: A distributed resilient adaptable cloud oriented platform," in *2013 IEEE 27th International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*, pp. 1490–1497, May 2013.

[8] "Amazon web services simple monthly calculator." http://calculator.s3.amazonaws.com/. Accessed: 2016-04-01.

[9] "Azure pricing calculator." http://www.windowsazure.com/en-us/pricing/calculator/. Accessed: 2016-04-01.

[10] "Cloud computing price comparison — cloudorado - find best cloud server from top cloud computing companies." http://www.cloudorado.com/. Accessed: 2016-04-01.

[11] M. Zhang, R. Ranjan, S. Nepal, M. Menzel, and A. Haller, "A declarative recommender system for cloud infrastructure services selection," in *Proceedings of the 9th International Conference on Economics of Grids, Clouds, Systems, and Services*, GECON'12, (Berlin, Heidelberg), pp. 102–113, Springer-Verlag, 2012.

[12] M. K. Qureshi and Y. N. Patt, "Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches," in *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, (Washington, DC, USA), pp. 423–432, IEEE Computer Society, 2006.

[13] Q. Zhu and T. Tung, "A performance interference model for managing consolidated workloads in qos-aware clouds," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 170–179, June 2012.

[14] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, (New York, NY, USA), pp. 3–14, ACM, 2013.

[15] A. Levin, K. Barabash, Y. Ben-Itzhak, S. Guenender, and L. Schour, "Networking architecture for seamless cloud interoperability," in *2015 IEEE 8th International Conference on Cloud Computing*, pp. 1021–1024, June 2015.

[16] M. Paolino, N. Nikolaev, J. Fanguede, and D. Raho, "Snabbswitch user space virtual switch benchmark and performance optimization for nfv," in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*, pp. 86–92, Nov 2015.

[17] K. Lee, D. Kim, D. Ha, U. Rajput, and H. Oh, "On security and privacy issues of fog computing supported internet of things environment," in *Network of the Future (NOF), 2015 6th International Conference on the*, pp. 1–3, Sept 2015.

[18] I. Butun, B. Kantarci, and M. Erol-Kantarci, "Anomaly detection and privacy preservation in cloud-centric internet of things," in *Communication Workshop (ICCW), 2015 IEEE International Conference on*, pp. 2610–2615, June 2015.

[19] P. PARWEKAR, "From Internet of Things Towards Cloud of Things," in *Computer*, pp. 329–333, 2011.

[20] T. Bui, "Analysis of docker security," *arXiv preprint arXiv:1501.02967*, 2015.

[21] M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, W. Hu, and B. Amos, "Edge analytics in the internet of things," *IEEE Pervasive Computing*, vol. 14, pp. 24–31, Apr 2015.

[22] A. Celesti, M. Fazio, M. Villari, and A. Puliafito, "Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems," *Journal of Network and Computer Applications*, vol. 59, pp. 208 – 218, 2016.

[23] M. Fazio, A. Celesti, M. Villari, and A. Puliafito, "The need of a hybrid storage approach for iot in paas cloud federation," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 779–784, May 2014.

[24] S. Abolfazli, Z. Sanaei, E. Ahmed, A. Gani, and R. Buyya, "Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 337–368, First 2014.

[25] 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*, SOCC '11, (New York, NY, USA), pp. 22:1–22:14, ACM, 2011.

[26] M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 243–254, Aug. 2010.

[27] "Openshift origin." https://www.openshift.org/. Accessed: 2016-04-01.

[28] AWS - Autoscaling, https://aws.amazon.com/autoscaling/.