Importance of Application-level Resource Management in Multi-cloud Deployments

Zoran Dimitrijevic, Cetin Sahin, Christian Tinnefeld, Jozsef Patvarczki
SAP Labs
{zoran.dimitrijevic, cetin.sahin, christian.tinnefeld, jozsef.patvarczki}@sap.com

***Abstract*—Cloud service providers started with Infrastructure as a Service (IaaS) offerings and over time expanded into Platform as a Service (PaaS) and Software as a Service (SaaS). Even though each provider has a rich product offering, there are many scenarios where a multi-cloud strategy is desirable: utilizing economic dynamics, preventing data lock-in with one vendor, circumventing geographic restrictions, complying with local regulations, or combining on-premise and public-cloud resources. The challenge from a consumer perspective with multi-cloud deployments is the lack of a common abstraction for the offered products and a standardized way to express all of the application requirements for the resulting deployments. In this paper, we contribute by making yet another case for multi-cloud deployments and by predicting the emergence of a new generation of application-level resource managers which will natively support multi-cloud for enterprise applications. We identify three main components of the feedback loop controlled application-level resource managers: the software life-cycle manager, the data storage and access manager, and the service execution manager.**

I. Introduction

The success of the World Wide Web enabled the rise of large Internet companies during the dot-com era which led the expansion of parallel and distributed computing from scientific and national supercomputing centers into corporate data centers and commercial applications. Google led the technological change with proprietary technologies like MapReduce [13], GFS [16] and Borg [34] which inspired the next generation of open-source projects in big data infrastructure including Apache Hadoop [14], HDFS [29], Mesos [19], Hive [32], and Spark [35]. These solutions enabled companies to build their on-premise data centers running on Linux, utilizing a mixture of commercial and open-source big data analytics software.

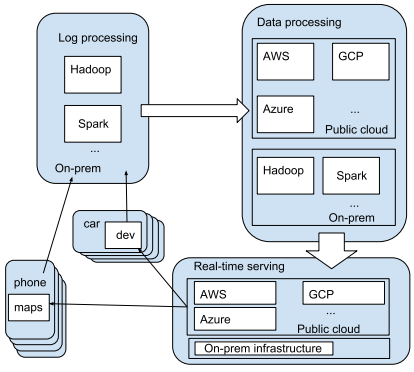

Fig. 1. Example: Data lifecycle for globally distributed maps application.

The emergence of public cloud enabled next generation of startups to run big data and globally distributed applications without building their own data centers. They are often built so that they can be easily containerized and can be effectively packaged using Docker images [1]. Kubernetes [8] is quickly becoming the preferred common infrastructure to enable running containerized distributed applications on either cloud or on-premise data centers. The popularity and success of these new applications running everywhere (custom apps usually run on smartphones while services run and store data in public cloud) changed the boundary conditions for capturing, processing, and storing data so much that governments adapted regulations regarding security and privacy of personal data. The European Union General Data Protection Regulation (GDPR) [5] is an example of this trend.

As an example of a globally distributed application, Figure 1 depicts a familiar mapping application that runs on GPS-enabled devices including smart phones and car navigational systems. These devices are connected to Internet and use real-time services from a globally distributed infrastructure that runs on a hybrid of on-premise hardware and public cloud. Highly private data on each device is logged on provider's data-centers and is then processed and stored using big data infrastructure.

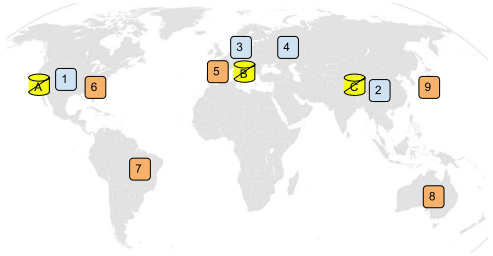

Fig. 2. Example: Data storage, processing and serving locations.

Figure 2 depicts typical locations where globally distributed applications access their backend services. Yellow disks depict locations of data centers storing private data. Often, these locations are constrained by local regulations. Blue squares numbered 1 to 4 depict where mapping service process private data and host personalized services. Orange squares numbered 4 to 8 depict frontends that only contain public data which can be served anywhere. It is crucial to always keep private data secure and make sure that processing and storing is done according to complex set of still evolving legal regulations

978-1-7281-0218-4/19/$31.00 ©2019 IEEE
DOI 10.1109/IC2E.2019.00006

(like GDPR [5] or China's Cybersecurity Law [4]). Once this data is aggregated and anonymized, it is used to improve the quality of service (accuracy of maps, real-time update of traffic conditions, personalized search and navigation, etc.). We expect that maintaining privacy while providing high quality-of-service and adhering to local regulations will be commonly solved by leveraging multi-cloud deployments.

II. Survey of Existing Technologies

Resource allocation problems are typically NP-hard, but due to their practical relevance, a large body of relevant academic work, algorithms, heuristics, commercial and open-source solutions are available.

A. Resource management within a datacenter

Resource management at cluster level is covered by academia for several decades, partially motivated by operating clusters for high performance computing (HPC): Buyya [9] surveys representative cluster systems, describes resource management as well as scheduling techniques. Workload managers such as Slurm [21] or TORQUE [31] enable resource scheduling in systems based on the widely adopted Message Passing Interface (MPI) [25]. The advent of large-scale Internet applications over the last 15 years increased the size and distribution of the utilized clusters as well as underlying availability and scalability properties and requirements in comparison to high performance computing. This resulted in the statement by Zaharia et al. that *the datacenter needs an operating system* [36]. Subsequent state-of-the-art work on datacenter level resource management emerged out of industry and open-source projects.

Apache Yarn [33] decouples the programming model from the resource management infrastructure in Hadoop [14]. Yarn enables scheduling of jobs executed by programming frameworks such as Hadoop MapReduce, Spark, Tez, and others. Such applications can express their resource requests through an extensible resource vector (e.g. required processing and memory), locality preferences, and priority of requests within the application. The Resource Manager matches a global model of the cluster state against a digest of the different resource requests. Apache Yarn supports plugable schedulers and elastic scaling of cluster nodes.

Facebook Bistro [17] addresses the scheduling of two different workloads on the same cluster. The two different workloads are either data-intensive batch jobs or latency sensitive requests from live customers. Bistro presents a tree-based resource model which incorporates information such as resource type, node count and change rate. In addition, it captures information about each job such as the degree of concurrency, the change rate, the involved data, and average duration.

Mesosphere with Apache Mesos [23] as its kernel is a cluster manager centered around deploying containers. The incorporated job scheduler (primarily intended for running Spark on Mesos) takes the required memory and CPU resources as input. It can operate in a coarse-grained mode providing lower latency or in a fine-grained mode providing higher utilization.

Uber Peloton [10] is a cluster management system built on top of Apache Mesos. Mesos takes care of resource allocation and task execution, while Peloton covers task placement and preemption as well as job and task lifecycle. Peloton's resource model defines how all resources in a cluster are divided and applies hierarchical max-min fairness. All resources available in the cluster are grouped into resource pools and every resource pool has different dimensions such as CPU, memory, disk size, or GPU.

Kubernetes [8] is emerging as a preferred cluster manager for running distributed containerized applications in public and private clouds. Kubernetes is inspired by Google Borg [34], cluster manager for Google private data centers running most Google services and data-processing pipelines. In Google Borg, each job consists of multiple tasks and each task runs inside a Linux container and can express its resource requirements such as CPU cores, memory, disk space, or TCP ports. Kubernetes runs applications in Linux containers that are grouped in pods. A pod is a group of one or more containers with shared network and storage. Docker [1] is the most commonly used container runtime. Kubernetes supports networking and storage models that enable plugable solutions for running Kubernetes clusters in public cloud and in private data centers. It is currently one of the most active open-source projects.

B. Resource management across multi-clouds

Over the last decade, the increase of scale and product variety offered by cloud computing providers such as Amazon Web Services, Microsoft Azure or Google Cloud Platform introduces new resource management challenges: no longer are resources typically consumed from just a single provider, but it is possible to mix and match the utilized resources from different vendors (often referred to as *multi-cloud*). The resulting challenge of measuring and comparing the service offerings from a business perspective and in a standardized way is addressed by the Cloud Services Measurement Initiative Consortium [30] who publishes a Service Measurement Index Framework [11]. This framework currently defines over 30 attributes of cloud service characteristics grouped into seven categories such as accountability, agility, assurance, financials, performance, security and privacy, and usability. While this might seem only relevant from a business perspective at a first glance, academic work in the context of cloud computing suggests that such attributes will also impact the requirements towards resource management [26], [28].

The STRATOS project [27] introduces a cloud broker service which can deploy and run cloud applications on different cloud providers. The objectives of the broker are a) optimizing costs and b) avoiding lock-in. The former is expressed by taking the prices charged for different service into consideration. The latter is incorporated by aiming at an equal distribution of the different workloads across cloud providers. Coutinho et al. [12] address the cost and execution time optimization across

multiple cloud providers. They provide a heuristic based on a a greedy randomized adaptive search procedure which takes a) the different cloud provider packages and their respective cost and provided resources as input and b) the consumer requirements in terms of maximum cost, maximum time, disk storage, memory capacity, and processing demands. Han et al. [18] propose a recommendation system which helps to select a cloud service for a particular task based on the quality-of-service of the network offered by the service provider and their virtual machine platform.

Gardener [15] is an open-source project for managing multiple Kubernetes clusters that can run in public cloud or in private data centers. This is one of the approaches for managing multi-cloud applications running on Kubernetes clusters.

III. Future Multi-cloud Directions

Over the last decade, industry and academia have shown great interest in managing cloud resources and provisioning them efficiently in a multi-tenant environment considering the relevant cost and service metrics as discussed in Section II. However, most of these developments mainly focus on a single cloud provider. With the advances in cloud technologies, the notion of solely consuming all services from a single cloud is disappearing. The 2018 State of the Cloud report [6] reveals that 81% of enterprises adopt a multi-cloud approach. What are the *main dynamics* of this trend?

1) Economic Dynamics: For small-scale enterprises and businesses, the cloud provides a rich stack of solutions that make it cheaper to develop and operate distributed applications. However, as the product matures and reaches a vast consumer space, the costs of operating it in the cloud increase. By adopting multi-cloud solutions, companies are able to select which services to use from the competing public cloud providers. Also, by adopting solutions that can operate on different cloud providers, companies can migrate their applications to a different cloud provider or to custom on-premise data centers.

From a price-performance aspect, the crucial advantage of moving to multi-cloud is the ability to negotiate with cloud providers. Being able to operate and run workloads on multiple cloud providers gives great flexibility and comfort for enterprises while they are negotiating with vendors. The competition between cloud vendors will lead to better deals and reduced cost for service and application deployments.

Another major economical concern for moving away from a single cloud provider business model is completely relying on the pricing model of the cloud vendor. A drastic price increase could leave businesses in a very difficult situation if they cannot migrate to a different cloud vendor. To prevent having their businesses affected from such price fluctuations, companies started diversifying their workloads and services across multiple cloud providers, so that they will not be directly impacted in case of a drastic price increase by one particular provider.

2) Service Reliability: Other than the economic aspect, relying on a single cloud provider comes with a risk of service disruptions in case of failures in cloud providers. For example, during the early 2017 Amazon Web Services (AWS) S3 outage in the Northern Virginia Region [3], many businesses relying on only AWS became completely unavailable during the outage including Amazon's own dashboard website. Although multi-cloud does not solve the problem of having service disruptions entirely (i.e., multiple cloud providers might have interruptions in their services in the specific region due to the unforeseen natural disasters), it improves the reliability and availability of services. A good use-case scenario is Waze in this context [2]. In one of his talks, Tarcic explains how Waze utilizes both AWS and Google Cloud Platform (GCP) for better reliability. Apparently, Waze was serving only on AWS in 2015 and Tarcic highlighted that Waze barely survived during the outage. However, the later and more disruptive AWS S3 outage [3] did not impact Waze, since it was serving on multiple cloud providers.

3) Regulations: Globally serving companies need to deal with region or country specific regulations such as the General Data Protection Regulation (GDPR [5]) in the European Union, the California Consumer Privacy Act (CCPA [20]) in the United States, or the China's cybersecurity law [4] that was first adopted in 2017. Since China has strict regulations for cloud providers (data centers in China are isolated from the global network and some services outside China are directly blocked), it is hard for cloud providers to serve inside such regulation-intensive regions that give the power to agencies of conducting a remote network inspection without informing the third party. This results in not having enough geographic coverage for some regions. The performance of services highly depends on bringing services closer to the users. Therefore, businesses operating in such regions need to work with local cloud providers. On the other hand, European GDPR [5] strictly requires storing and processing customer data in compliant data centers (i.e. physically located in the EU). It is very likely that we will have more regulations such as GDPR or CCPA and it will be increasingly harder for a single cloud provider to comply with all of these regulations. Also, increasingly code containing trade secrets and high-value software runs only as a service in secure locations in order to prevent piracy or to comply with local regulations (for example, encryption). Therefore, it is becoming necessary for enterprises and businesses serving in multiple regulated areas to use a multi-cloud approach.

A. Multi-cloud Resource Management

In the past, most applications run on a single on-premise machine or a mainframe, with on-premise storage solutions. Next generation of applications run on private data centers (e.g., Yahoo, Google, or Facebook). Most new applications natively support public cloud. Existing enterprise software is often refactored so that it can also efficiently run in public cloud. Applications in the cloud usually operate in software-as-a-service model (as opposed to packaged model). In addition

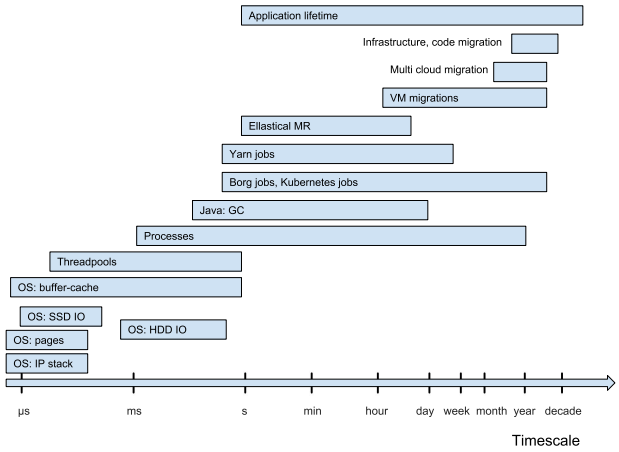

Fig. 3. Outline for various system components used in large scale applications.

to running production workloads and services, application developers also run their development, testing, staging and other pipelines in the cloud. Many large companies operate globally, and have developers all over the world. These various environments usually have different reliability and security requirements. For example, some portions of source code should not be visible to all developers in all regions (for example, encryption or code with high-value trade secrets). Testing and staging environments can run on cheaper, less reliable infrastructure (spot instances or on-premise mini data centers).

Figure 3 illustrates typical order-of-magnitude timescales for various components of systems running as globally distributed applications. Our intention here is not to be too strict but to depict, on a log timescale, how important is to start handling the month, year and even decade-level resource optimization problems using automated application-level schedulers. Schedulers and resource managers in operating systems operate at the level of microseconds up to few seconds. Some state in operating systems is longer lived, for example frequently accessed blocks can stay in buffer cache for days or months, and some processes live until reboots. Large deployments try to reboot all of the machines in their fleet at least once every 6 to 12 months (mostly to be up-to-date with security fixes [24]). As we move up the timescale, application code, virtual machines or frameworks deal with scheduling thread-pools, processes, garbage collection and so on.

Data-center resource managers schedule new jobs at the order of seconds or sometimes minutes, depending on the available resources and user quotas. These jobs then often run on hundreds or thousands of machines for minutes, hours or even weeks. Portion of these jobs are long-lived services that can run multiple years (even though individual replicas will be restarted or migrated in a rolling-upgrade fashion).

When applications run on multi-cloud, new resources and cloud services become available during their life span. Local regulations as well as prices change. Popular applications must scale to sustain exponential increases in user traffic. They face new challenges such as botnets attacking their services or slashdot effects when unexpected event happens (popular videos can expose scalability issues in video streaming services, news event can bring down web servers, etc.). Some applications can challenge the ability of cloud providers to support their scale. With the popularity of machine learning, hardware accelerators are becoming necessary for efficient large-scale computing workloads (for example, Google's tensor processing unit (TPU) [22] can efficiently run TensorFlow [7]). Political decisions, censorship attempts, or pricing changes can all trigger application owners to quickly migrate from one cloud provider to another. In order to handle these events, we predict the emergence of application-level resource managers designed to optimize and control seasonal, annual and even multi-year infrastructure. Their goal will be to efficiently run current workloads and to provide infrastructure for migration after application changes (including the changes in the applications software, user traffic and the underlying multi-cloud offerings).

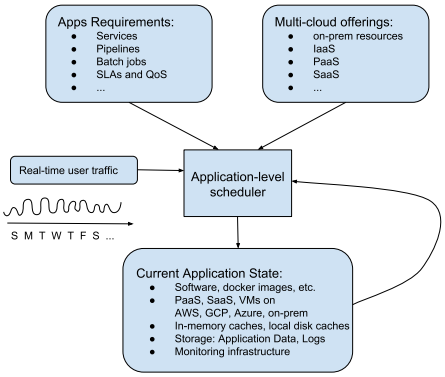

Fig. 4. Application-level resource scheduler.

Figure 4 depicts a scheduler for the application-level resource management. Using the current state of the infrastructure on which an application runs (including the utilization metrics from monitoring and logs, data storage and database locations, etc.), and up-to-date application requirements and multi-cloud offerings (current pricing and the expected performance for each available resource), an application-level scheduler makes decisions how to change the multi-cloud infrastructure (migrate or replicate data, move processing pipelines to cheaper or faster regions, etc.). We expect that the first generation of these application-level resource managers to operate in a semi-automatic fashion helping people monitor and migrate their workloads through configuration changes. Second generation of these managers will likely be mostly automatic where application developers only configure the managers themselves (scheduler algorithms, training models, and feedback loops).

We identify three main components of application-level resource manager:

- *Software life-cycle manager.* Large-scale applications are often developed and managed by large teams contributing

all over the world. As a norm, they depend on large code base consisting of millions of lines of open source and proprietary code. This code is written in different languages under different software licenses. For security, compliance and software-quality reasons it is critical to professionally manage code and deployment processes (code repositories, building, testing, performance evaluation, packaging, deployments, rolling upgrades, monitoring, profiling, or debugging). In multi-cloud deployments, it is critical that software is tested on all supported platforms and that fleet management is able to perform efficient and timely upgrades of all services (software upgrades, VM and container restarts, etc.). Software life-cycle manager would also manage the supported versions of client-code running on user personal devices and schedule their updates.

- *Data storage and access manager.* Large-scale applications generate and process large amount of data including private user data and service logs containing user-identifying data. In addition to protecting user data according to service agreements, applications must also satisfy the dynamic sets of local regulations such as GDPR [5]. Data storage and access manager makes decisions where to store the data (in the available databases, object stores, and file-systems located in different geo regions), how to store the data (for example, ensuring that data is properly replicated or encrypted-at-rest), who can access each data item (access control), and how to delete the data (according to data-retention policies). Since the cost of data access and storage changes over time, Data storage and access manager is also responsible for data migration, backup, and integrity inspection (for reliability, disaster recovery, and compliance).
- *Service execution manager.* Large-scale applications run on a fleet of nodes (dedicated physical machines and VM instances in multiple public clouds) and clusters (Kubernetes, Spark, Hadoop, etc.). In order to achieve good price-performance, a service execution manager must monitor the performance and utilization of all services and pipelines and then make decisions how to optimize its fleet. Considering the lack of standard abstractions in multi-cloud deployments, we expect that most large-scale applications running on multi-cloud will evolve to rely only on a subset of well supported infrastructure (for example, VMs, Docker images and Kubernetes).

The application-level scheduler should act as a feedback-loop controller that takes the real-time user traffic and the current state of the system to model, tune and converge to the desired state of multi-cloud infrastructure adequately. The fine grained analysis of this traffic together with the desired requirements sets the base point for the required changes. Figure 4 shows the feedback loop of the scheduler that derived from the relationship between the base point and the Current Application State (observed values and parameters). To minimize the differences with respect to this relationship, every component of the application-level scheduler (Software life-cycle manager, Data storage and access manager, and Service execution manager) should have the responsibility of brokerage, credentials management, monitoring and logging.

IV. Conclusion

Because of the various reasons enumerated in this paper, globally distributed applications are commonly deployed in multi-cloud. The challenge from a developer perspective with multi-cloud deployments is that we still do not have common abstractions for the offered products and a standardized way to express all of the application requirements. In this paper, we contribute by making yet another case for multi-cloud deployments and by predicting the emergence of a new generation of application-level resource managers which will natively support multi-cloud for enterprise applications. We identify three main components with the brokerage capability of the application-level resource managers: the software life-cycle manager, the data storage and access manager, and the service execution manager. We also outlined a feedback-loop controller for predicting, changing and tuning the multi-cloud infrastructure.

Acknowledgements

We would like to thank SAP Labs, our managers and coworkers for supporting this research. We would also like to thank the conference chairs for the invitation and for helping us with their valuable comments and suggestions.

References

[1] Docker. *https://www.docker.com/*, 2013.

[2] Spinnaker enables multi-cloud deployments for waze, and saved 1,000 people. *https://blog.armory.io/spinnaker-enables-multi-cloud-deployments-for-waze-and-saved-1-000-people*, 2017.

[3] Summary of the amazon s3 service disruption in the northern virginia (us-east-1) region, 2017. https://aws.amazon.com/message/41926/?tag=bisafetynet2-20.

[4] CH china cybersecurity law. *http://www.mps.gov.cn/n2254314/n2254409/n4904353/c6263180/content.html*, 2018.

[5] EU General data protection regulation (GDPR). *https://eugdpr.org/*, 2018.

[6] Rightscale 2018 state of the cloud report, 2018. https://www.rightscale.com/lp/state-of-the-cloud.

[7] M. e. a. Abadi. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, Berkeley, CA, USA, 2016. USENIX Association.

[8] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes. Borg, omega, and kubernetes. *Queue*, 14(1):10, 2016.

[9] R. Buyya. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[10] M. Cai and M. Bansal. Peloton: Uber's unified resource scheduler for diverse cluster workloads. https://eng.uber.com/peloton/, October 2018.

[11] C. S. M. I. C. (CSMIC). Service measurement index framework version 2.1, 2014.

[12] R. de C. Coutinho, L. M. A. Drummond, and Y. Frota. Optimization of a cloud resource management problem from a consumer perspective. In D. an Mey, M. Alexander, P. Bientinesi, M. Cannataro, C. Clauss, A. Costan, G. Kecskemeti, C. Morin, L. Ricci, J. Sahuquillo, M. Schulz, V. Scarano, S. L. Scott, and J. Weidendorfer, editors, *Euro-Par 2013: Parallel Processing Workshops*, pages 218–227, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[13] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, pages 137–150, San Francisco, CA, 2004.

[14] M. C. Doug Cutting et al. Apache Hadoop. *https://hadoop.apache.org/*, 2006.

[15] R. Franzke and V. Chandrasekhara. Gardener - The Kubernetes botanist. *https://kubernetes.io/blog/2018/05/17/gardener/*, 2018.

[16] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In *ACM SIGOPS operating systems review*, volume 37, pages 29–43. ACM, 2003.

[17] A. Goder, A. Spiridonov, and Y. Wang. Bistro: Scheduling data-parallel jobs against live production systems. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, pages 459–471, Santa Clara, CA, 2015. USENIX Association.

[18] S.-M. Han, M. Mehedi Hassan, C.-W. Yoon, H.-W. Lee, and E.-N. Huh. Efficient service recommendation system for cloud computing market. In *Grid and Distributed Computing*, pages 117–124, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[19] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *NSDI*, volume 11, pages 22–22, 2011.

[20] C. L. Information. Sb-1121 california consumer privacy act of 2018. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB1121, September 2018.

[21] M. A. Jette, A. B. Yoo, and M. Grondona. Slurm: Simple linux utility for resource management. In *In Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP) 2003*, pages 44–60. Springer-Verlag, 2002.

[22] N. P. e. a. Jouppi. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pages 1–12, New York, NY, USA, 2017. ACM.

[23] D. Kakadia. *Apache Mesos Essentials*. Packt Publishing, 2015.

[24] M. Lipp et al. Meltdown: Reading kernel memory from user space. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 973–990, 2018.

[25] Message Passing Interface Forum. Mpi: A message-passing interface standard, version 2.2. Specification, High Performance Computing Center Stuttgart (HLRS), September 2009.

[26] F. Paraiso, N. Haderer, P. Merle, R. Rouvoy, and L. Seinturier. A federated multi-cloud paas infrastructure. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 392–399. IEEE, 2012.

[27] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski. Introducing stratos: A cloud broker service. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, CLOUD '12, pages 891–898, Washington, DC, USA, 2012. IEEE Computer Society.

[28] D. Petcu. Multi-cloud: expectations and current approaches. In *Proceedings of the 2013 international workshop on Multi-cloud applications and federated clouds*, pages 1–6. ACM, 2013.

[29] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The Hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, pages 1–10. IEEE, 2010.

[30] J. Siegel and J. Perdue. Cloud services measures for global use: The service measurement index (smi). *2012 Annual SRII Global Conference*, pages 411–415, 2012.

[31] G. Staples. Torque resource manager. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, SC '06, New York, NY, USA, 2006. ACM.

[32] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[33] V. K. Vavilapalli et al. Apache Hadoop YARN: Yet another resource negotiator. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 5:1–5:16, New York, NY, USA, 2013. ACM.

[34] A. Verma, L. Pedrosa, M. R. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes. Large-scale cluster management at Google with Borg. In *Proceedings of the European Conference on Computer Systems (EuroSys)*, Bordeaux, France, 2015.

[35] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

[36] M. Zaharia, B. Hindman, A. Konwinski, A. Ghodsi, A. D. Joesph, R. Katz, S. Shenker, and I. Stoica. The datacenter needs an operating system. In *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'11, pages 17–17, Berkeley, CA, USA, 2011. USENIX Association.