Toward Software-defined SLAs

Enterprise computing in the public cloud

Jason Lango, Bracket Computing

The public cloud has introduced new technology and architectures that could reshape enterprise computing. In particular, the public cloud is a new design center for enterprise applications, platform software, and services. API-driven orchestration of large-scale, on-demand resources is an important new design attribute, which differentiates public-cloud from conventional enterprise data-center infrastructure. Enterprise applications must adapt to the new public-cloud design center, but at the same time new software and system design patterns can add enterprise attributes and service levels to public-cloud services.

This article contrasts modern enterprise computing against the new public-cloud design center and introduces the concept of software-defined SLAs (service-level agreements) for the public cloud. How does the public cloud stack up against enterprise data centers and purpose-built systems? What are the unique challenges and opportunities for enterprise computing in the public cloud? How might the on-demand resources of large-scale public clouds be used to implement software-defined SLAs? Some of these opportunities might also be beneficial for other public-cloud users such as consumer Web applications.

Today the dominant architectural model for enterprise computing is the purpose-built system in private data centers, engineered to deliver guaranteed service levels to enterprise applications. The architectural model presented by large-scale multitenant public clouds is quite different: applications and services are built as distributed systems on top of virtualized commodity resources. Many large-scale consumer Web companies have successfully delivered resilient and efficient applications using this model.

Getting enterprise applications into the public cloud is no easy task, but many companies are nonetheless interested in using cloud infrastructure broadly across their businesses, whether via public-cloud or private-cloud deployments. New levels of flexibility and automation promise to streamline IT operations. To become the primary computing platform for most applications, the public cloud needs to be a high-performance enterprise-class platform that can support business applications such as financial analysis, ERP (enterprise resource planning) systems, and supply chain management. The next section looks at practical systems considerations necessary for implementing enterprise cloud services.

ENTERPRISE SLAS VS. PUBLIC-CLOUD DESIGN CENTER

Enterprise can be interpreted broadly as a business context requiring premium attributes such as high availability, security, reliability, and/or performance. This definition holds regardless of whether an application is legacy or new. For example, an enterprise analytical database might be implemented using a new scale-out architecture, and yet have enterprise requirements. Data security may be at a premium for either regulatory or business reasons. Data integrity is at a premium because a mistaken

business decision or financial result can cost the company real revenue or possibly even a loss in market value. Enterprise service levels are simultaneously of high business value and technically challenging to implement.

SLAs specify enterprise service-level requirements, often in the form of a legal contract between provider and consumer, with penalties for noncompliance. Concrete and measurable SLOs (service-level objectives) are individual metrics used to test that an SLA is being met. This distinction is important in the context of this article, which later identifies programmatically enforceable SLOs governed by a software-defined SLA.

In this article, *public cloud* refers to a platform that deploys applications and services, with on-demand resources in a pool large enough to satisfy any foreseeable demand, run by a third-party CSP (cloud service provider). Many popular IaaS (Infrastructure-as-a-Service) and PaaS (Platform-as-a-Service) providers meet this definition, including Amazon Web Services, Microsoft Azure, and Google Compute Engine. Cloud computing conventionally includes on-demand self-service, broad network access, resource pooling (a.k.a. multitenancy), rapid elasticity, and measured service.¹²

Unfortunately, there is a recognized gap between service levels the enterprise expects and what today's public cloud delivers. Current public-cloud SLAs are weak—generally providing 99.95 percent data-center availability and no guarantee on performance—and penalties are small.⁴ Which service levels matter? Why are they challenging? How can they be implemented? Let's take a long view on the advancement of both the public-cloud and enterprise infrastructure. While the public cloud is still undergoing rapid development and growth, it's possible to observe some trends.

RELIABILITY AND AVAILABILITY

The availability component of an enterprise SLA can be technically challenging. For example, a business-critical application might not tolerate more than five minutes of downtime per year, conforming to an availability SLO of 99.999 percent ("5 nines") uptime. In contrast, resources in the public cloud have unit economics falling somewhere between enterprise and commodity hardware components, including relatively high expected failure rates. Amazon's virtual block devices, for example, have an advertised annual failure rate of 0.1 – 0.5 percent, meaning up to 1 in 200 will fail annually.

Business-critical applications often have a low tolerance for application-level data inconsistency and zero tolerance for data corruption. Many enterprise applications may be reimplemented using an "eventual consistency" architecture to optimize both performance and availability at the cost of compensating for temporary inconsistency.³ When the business risk or penalty is high enough, however, some enterprise applications prefer taking some downtime and/or data loss rather than delivering an incorrect result. If the availability SLO is stringent enough, it places pressure on software to implement rapid recovery to maintain the requisite amount of uptime.

Leading CSPs have pushed for developers to adopt new fault-tolerant software and system-design patterns, which make few assumptions about the reliability and availability of the underlying infrastructure. The public-cloud design center encourages "designing for failure"¹⁰ as part of normal operation to achieve high availability. This creates a need for fault-tolerant software to compensate for known unreliable infrastructure, metaphorically similar to how a RAID (redundant array of independent disks) compensates for unreliable physical media. Reliability and availability have become software problems. On the plus side, it's an opportunity to build more robust software.

PERFORMANCE

Enterprise-application performance needs vary. End-user-facing applications might be managed to a specific response-time SLO, similar to a consumer Web application measured in fractions of a second. Important business applications such as ERP and financial analysis might be managed to both response time and throughput-oriented SLOs, supportive of specific business objectives such as overnight trading policy optimization.

In the public cloud, many performance challenges are byproducts of multitenancy. Physical resources behave as queuing systems: oversubscription of multitenant cloud infrastructure can cause large variability in available performance.¹⁶ “Noisy neighbors” may be present regardless of whether storage is rotational or solid state, or whether networking is 1 gigabit or 100 gigabits. Compute oversubscription can also negatively impact I/O latency.¹⁸ An operational tradeoff exists between performance and cost. Multitenant public clouds allow for high utilization rates of physical infrastructure to optimize costs to the CSP, which may be passed on as lower prices. Unfortunately, performance of shared physical resources cannot be guaranteed at the lowest possible fixed cost. Performance of oversubscribed physical resources can fluctuate randomly but is “cheap,” whereas performance of statically partitioned physical resources can be guaranteed but at a higher cost. Amazon Provisioned IOPS (I/O operations per second) is an example of this tradeoff, where guaranteed performance comes at roughly double the cost.²

Flexible use of virtual resources is a requirement in the public cloud, especially if performance is to be guaranteed. Distributed systems must be actively managed to achieve performance objectives. The advantage of on-demand resources is that they can be reconfigured on the fly, but this is also a major software challenge.

SECURITY

Security requirements vary by application category but generalize as risk management: the higher the business or regulatory value of an application or dataset, the more stringent the security requirements. In addition to avoiding denial of service, which aligns with availability, and avoiding “data leakage,” there is also a desire to increase “mean time to compromise” by putting multiple layered security controls in place, in recognition that no individual system can be perfectly secure.

The public cloud is an interesting environment from an enterprise security perspective. On the one hand, a multitenant public cloud is considered a new and worrisome environment. On the other hand, the ability to impose logical security controls and automate policy management across running workloads presents an opportunity. Logical controls are more flexible, auditable, and enforceable than physical controls. Network access-control rules are a classic example of logical controls, which may now be applied directly to virtual machines rather than indirectly via physical switch ports. Logical segmentation can be provisioned dynamically and can shrink to fit the exact resources in a running workload and move when the workload moves.

The public cloud demands new security tools and techniques, requiring a rethinking of classic security techniques. There is a need for programmatically expressing security SLOs. User, application, and dataset-centric policy enforcement are worthy areas of further exploration toward the challenge of implementing higher-level security SLAs (e.g., “users outside the finance group may not access financial data” and “data at rest must be reencrypted every two hours”).

FROM PURPOSE-BUILT SYSTEMS TO DISTRIBUTED SYSTEMS

Enterprise data centers are typically optimized for a predetermined set of use cases. Purpose-built systems, such as the one described in figure 1, are engineered to achieve specific service levels with a fixed price/performance via preintegrated components. These come in various form factors: hardware appliances, preintegrated racked systems, and more recently, virtual appliances and cloud appliances (providing an out-of-the-box private cloud with preconfigured SLAs). Vendors vertically integrate hardware and software components to provide service-level attributes (e.g., guaranteed rate I/O, reconfiguration of physical resources, fault isolation, etc.). Higher-level SLAs are met by combining a vendor's deployment recommendations with best practices from performance and reliability engineering.

Purpose-built systems currently offer very high performance levels for workloads that require high-bandwidth inter-node communication. I/O-intensive data analytics are an example: achieving low response-time SLAs means that sustained inter-node traffic in tens to hundreds of gigabytes per second will exceed the conventional 10-gigabit Ethernet commonly found in large-scale public-cloud environments.

Enterprise buyers might justify additional expense for specific use cases—for example, technical computing users paid for early access to GPGPUs (general-purpose computing on graphics processing units) for parallel computation, while data warehousing users paid for InfiniBand or proprietary Banyan networking for higher-bandwidth data movement. In practice, specialized technology such as GPGPU has been delivered in limited quantities and geographies in the public cloud, with expanded availability over time. It takes a premium to continually stay on the bleeding edge.

STATIC AND INTEGRATED, MEET DYNAMIC AND DISTRIBUTED

CSPs are continually improving their offerings. The hardware gap between purpose-built systems and the public cloud is closing. Public-cloud providers have created hardware designs tailored for large-scale deployment and operational efficiency, with the Open Compute Project as a popular example.⁹ The economic incentive for CSPs is clear: more use cases means more revenue. Amazon Web Services has been increasing instance (VM) performance for some time, motivated lately by business analytics (Amazon Redshift). This trend will likely continue because of the practical benefits of right-sizing virtual machine resources—for example, simple scalability issues (Amdahl's law), reduced cost of data movement between nodes, or price/performance/power efficiency. Additionally, some CSPs might acquire purpose-built systems that provide guaranteed service levels, such as cloud appliances.

The public cloud is dynamic and distributed, in contrast to a purpose-built system, which is static and integrated. The CSP's virtualized resources are optimized for automation, cost, and scale, but the CSP also owns the platform and hardware-abstraction layers. Abstraction is a challenge in providing higher-level SLAs—it is difficult to guarantee service levels, not knowing which virtual resources are collocated within the same performance and failure domains.

Enterprise infrastructure has an opportunity to transform. New enterprise applications are being written against cloud-friendly software platforms such as Cloud Foundry and Hadoop. Implicitly, in the challenging effort of implementing highly available distributed systems, applications will be made robust against component failures, and the need for highly available infrastructure will diminish over time. Moreover, CSPs and platform services can help accelerate this transition. Microsoft Azure, for example, makes failure domains visible to distributed applications: a workload can allocate nodes from independent failure domains within the same data center. SLAs can be delivered in software on the public cloud, providing enterprise attributes and service levels to enterprise applications.

TOWARD SOFTWARE-DEFINED SLAS

On-demand resources in the public cloud are effectively infinite relative to the needs of the enterprise consumer. To appreciate this, it helps to get a sense of scale. Although rarely publicly disclosed, individual public CSPs have server counts conservatively estimated to be in the hundreds of thousands and growing rapidly.¹⁴ That's already at least one order of magnitude larger than a

reasonably large enterprise data center with tens of thousands of servers. At that scale, it's possible for an entire enterprise data center to fit within the CSP's idle on-demand capacity. In contrast to the millions of end users on a large-scale Web site, a population of 50,000 end users is a large number for an enterprise-business, custom in-house departmental, batch-processing, or analytical application.

This new design center fundamentally alters an architectural assumption in today's enterprise applications and infrastructure: the resource envelope is no longer fixed as in purpose-built systems or capacity-managed by central IT. Even additional CPUs and RAM can now be logically provisioned by enterprise applications and platform services at runtime, either directly or indirectly, by launching new virtual machines. The resource envelope is limited only by budget, but software has to be designed for the public cloud in order to exploit this.

While limited SLAs are available from the CSP, application and platform software components are generally required to provide guarantees around application characteristics such as performance, resiliency, availability, and cost. Because of the challenges associated with multitenancy, public-cloud applications currently make few assumptions about the infrastructure underneath them. They are built to tolerate arbitrary failures by design and implement their own SLAs. There is an opportunity to create new architectural design patterns to help systemically solve some of these problems and allow for reusable components.

SD-SLAs (software-defined SLAs) are expected to increase in platform software components and cloud services optimized for the public-cloud design center. The next section provides examples, implementation considerations, and limitations and future opportunities.

DEFINING SD-SLAs

SD-SLAs offer a new design pattern that formalizes SLAs and SLOs as configurable parameters of public-cloud software components. Those components then manage underlying resources to meet specific measurable SLO requirements. With on-demand resources, a software systems layer can be implemented to meet some SLOs, which previously required planning, static partitioning, and overprovisioning of resources. Cloud service APIs may then begin to incorporate SD-SLAs as runtime configurations.

Programmatic SLOs within an SD-SLA might specify metrics for fundamental service levels such as response times, I/O-throughput, and availability. They might also specify abstract but measurable attributes such as geographic or workload placement constraints. Some examples: Amazon's service-oriented architecture featured a data service managed to a realtime SLA, which was dynamically sized and load-balanced to "provide a response within 300 ms for 99.9 percent of its requests."⁸ Amazon Provisioned IOPS allows for a given number of I/O operations per second to be configured per storage volume.

Many interesting targets for software-defined SLAs are presented in the recent *ACM Queue* article, "There's Just No Getting around It: You're Building a Distributed System,"⁶ which also describes the challenge of building real-world distributed systems.

SD-SLAs should be vendor- and technology-independent, specified in logical units, and objectively measurable—for example, configure a desired number of I/O operations per second, as opposed to the number of devices necessary to achieve it; or an amount of bandwidth between nodes, as opposed to a physical topology.

IMPLEMENTATION CONSIDERATIONS AND EXAMPLES

SD-SLAs must necessarily be implemented in a distributed system in the public-cloud design center: for runtime-configurable SLOs to scale out; for high availability and fault tolerance; and to use on-demand compute and I/O resources.

First, consider this simple example: a reconfigurable I/O-throughput SLO guaranteeing some number of IOPS in the context of a distributed key-value store (see figure 2). Assume the key-value store uses N-way replication with quorum-like consistency, as in Dynamo, and that underlying storage volumes support a configurable performance capacity, as in Amazon Provisioned IOPS. Given an initial configuration for I/O-throughput T , an SD-SLA-aware resource manager would allocate volumes sufficient to provide the desired aggregate I/O capacity. Conservatively and suboptimally, let's assume it allocates $T \times N$ IOPS to each volume, as each `get()` operation generates N concurrent

I/O requests. In this example, the SD-SLA-aware resource manager could treat both SLO reconfiguration and poor-performing volume scenarios as a standard replica failure/replacement, providing automatic reconfiguration without further complicating the system with additional data copy code paths. In the event that the I/O-throughput SLO is reconfigured to T' , new volumes would be allocated at $T' \times N$ IOPS and old volumes failed out, until the system converged to T' aggregate I/O-throughput capacity. In the interim, a weighted I/O distribution policy might be used to maximize I/O throughput. In the real world, further performance and cost optimization would be required, and more sophisticated algorithms could be considered, such as an erasure coding instead of simple replication.

Given the challenge of distributed system development, a one-size-fits-all SD-SLA implementation is unlikely. A variety of programmatic SLOs may be implemented in application services, platform software components, or the CSP itself. The specific application context determines which components are appropriate for a given use case. As both the public cloud and enterprise applications are moving targets, the industry is likely to continue iterating on which attributes are provided by the CSP, versus application, versus software components and services in between.

Runtime reconfiguration for SD-SLAs is challenging. QoS (quality of service) techniques such as I/O scheduling and admission control are necessary but not sufficient. Application- or service-specific implementation is necessary for dynamically provisioning RAM, CPU, and storage resources to meet changing SLOs or to meet SLOs in the presence of changing environmental conditions. The value of SD-SLAs, however, may justify significant engineering effort and cost. An example is the implementation of peer-to-peer object storage to allow for more fluid use of underlying resources, including the runtime replacement of compute nodes and flexible placement of data. Some SD-SLA implementations may use closed-loop adjustments from control theory.¹¹ Runtime reconfiguration may go hand in hand with resiliency to failure, as component replacement, initial configuration, and runtime adjustment may all be managed in a similar application-specific manner.

Placement of computation and data must be considered for performance and data-availability SD-SLAs. Collocation of computation and data can alleviate some performance issues associated with multitenant networking. Examples include flexible movement of computation and data implemented in Hadoop, Dryad, and CIEL; placement-related SLOs implemented in Microsoft Azure and Amazon Web Services (Affinity Groups and Placement Groups, respectively); and data availability SLOs, specifying geographic placement and minimum number of replicas, implemented in Google Spanner.⁷

Tagging may be used in general to identify resources subject to SD-SLAs and specifically to implement security SLOs. In addition to resource tagging supported natively by CSPs, host-based virtual networking and OpenFlow offer further opportunities to tag users and groups in active network flows, similar to Cisco TrustSec and IEEE 802.1AE (the MAC security standard, also known as MACsec). Security SLOs may be implemented by associating user and group tags with access controls. Similarly, dataset-level tagging in storage service metadata assists in the implementation of dataset-level SLOs (e.g., data availability, replication, access control, and encryption key management policy).

ON-DEMAND COST OPTIMIZATION

Even with the sophisticated tools and techniques around purpose-built systems, overprovisioning is the de facto standard method for guaranteeing service levels across the lifetime of a system.

The entire cost of a purpose-built system must be paid up front, including the overhead of overprovisioning to meet SLAs and accommodate increasing usage over time. In contrast, the on-demand resources in the public cloud can be allocated and freed as needed, and thus may be billed according to actual use. This is an opportunity for the public cloud to outperform purpose-built systems in terms of operational efficiency for variable workloads.

Costs and resource allocation required to meet an SD-SLA may be tuned to optimize operational efficiency. Given that variable resources may be required to achieve different SLOs, a given SD-SLA may come associated with a cost function. Here are two fundamental theorems for the economics of SD-SLAs: (1) a change in any SLO must always be traded against cost as a random variable; and (2) in the face of changing underlying conditions (e.g., unpredictable multitenant resources), cost is a random variable even when all other SLOs are fixed.

Programmatic cost modeling¹³ and optimization¹⁵ are new themes in public-cloud research, and work is ongoing.

LIMITATIONS AND FUTURE OPPORTUNITIES

Unsurprisingly, there are both theoretical and practical limitations to software-defined SLAs. Since cost is always a system-level parameter that needs to be managed, some combinations may not work. An invalid combination, for example, would be if an application demands 1 million IOPS with 1-ms worst-case response time for a cost that is lower than the cost of the physical systems necessary to deliver this realtime SLA. Even given unlimited cost, some SLOs may be physically impossible to achieve (e.g., a bandwidth greater than the physical capacity of the underlying CSP or resource allocation faster than the underlying CSP is capable of providing it). Moreover, a poorly designed cloud service may not be amenable to software-defined SLAs—for example, if fundamental operations are serialized, then they cannot be programmatically scaled out and up to satisfy an SD-SLA.

With SD-SLAs, there are further opportunities to move to a continuous model for many important background processes, which previously needed to be scheduled because of the constraint of fixed resources. Consider that an enterprise-storage or database system, rather than trusting underlying physical storage controllers, might have a software process that scans physical media to ensure that latent bit errors are corrected promptly. Since this process is potentially disruptive to normal operation in a system with fixed compute and I/O resources, the typical approach is to run it outside of business hours, perhaps on weekends every two to four weeks. Future cloud services with SD-SLAs might be designed to allow important background processes to run continuously without impacting front-end service levels delivered to the application, since both the front-end service and continuous background processes may have independent programmatic SLOs that scale out using on-demand resources.

Dynamic resource management is an area where competition between CSPs may unlock new opportunities (e.g., “allocate a VM with a specific amount of nonvolatile RAM” or “add two more CPUs to this running VM”). Modern hypervisors already support this. Physical attributes can be disaggregated into individually consumable units. For example, compute resources can be allocated independently of I/O, I/O-throughput independent of capacity, and CPU and RAM independently of each other. This weakens the vertical-integration advantage of purpose-built systems. Amazon has approached this issue by offering a wide inventory of VM types,¹ although finding the right combination of CPU and RAM may still involve overprovisioning one or the other.

Enterprise macro-benchmarks must be tailored to the new public-cloud design center. Much effort has gone into rigorous infrastructure benchmarks such as SPC-1¹⁷ in the storage arena; however, the public cloud has introduced a fundamental economic shift—price/performance metrics need to factor in workload runtime. Thanks to the on-demand nature of the public cloud, price is a function of allocated resources over time, measured in hours or days since a workload started running, as opposed to a standard three-year life cycle of enterprise hardware. With SD-SLAs, allocated resources vary with time, front-end load, and whatever else is necessary to meet application SLAs. On the flip side, an I/O benchmark implemented in a massive RAM cache will yield stunning numbers, but price/performance must still be captured for this benchmark to be relevant. Further industry effort is necessary to evolve enterprise macro-benchmarks for the public cloud and SD-SLAs.

It is also natural to ask whether SD-SLAs are being met consistently. There are further opportunities to implement programmatic SD-SLA validation via automated test infrastructure and analytics.⁵ This offers the opportunity for third-party validation of SLAs and for assessing penalties appropriately.

Further industry and academic efforts can lead to fully flushing out the limits of software-defined SLAs. It would be worth seeing how far we can go along these lines, perhaps one day getting close enough to approximate: “What application response time are you looking for? Here’s what it will cost you.”

PUBLIC CLOUD TRANSCENDENT

The public cloud presents an opportunity to reimagine enterprise computing. It will be a rewarding journey for public-cloud services to take on the bulk of enterprise-computing use cases. As in past transitions, the transformation of enterprise applications from one model to the next can proceed incrementally, starting with noncritical applications and building upward as the ecosystem matures. The wheels are already in motion.

It is remarkable that a seven-year-old technology can be judged optimistically against the entire progress of enterprise infrastructure in the past 20-30 years. The pace of public-cloud innovation is relentless. A lot of energy and capital continues to pour into public-cloud infrastructure. Today, the public cloud is a multibillion-dollar market and growing rapidly. Any or all of today’s issues could be gone in the blink of an eye. Enterprise platforms have historically seen radical shifts in structure as a result of the changing economics of computing—from the mainframe to the client-server era. We are in the midst of another industry transformation.

Future enterprise applications and infrastructure may be built as distributed systems with reusable platform software components focused on the public cloud. This can assist information technology professionals and application developers in deploying fast and reliable applications without having to reinvent the wheel each time. Some enterprise features associated with reliability, availability, security, and serviceability could run continuously in this model. Runtime configuration of SD-SLAs provides an opportunity to manage based on the exact performance indicators that people want, as opposed to physical characteristics such as raw hardware or prepackaged SLAs. Enterprise applications can harness the scale, efficiency, and rapidly evolving hardware and operational advances of large-scale CSPs. These are all significant opportunities, not available in purpose-built systems but enabled by the large-scale, on-demand resources of the public cloud.

All engineers and IT professionals would be wise to learn about the public cloud and capitalize on

these trends and opportunities, whether at their current job or the next. The public cloud is defining the shape of new software—from applications to infrastructure. It is our future.

REFERENCES

1. Amazon Web Services. 2013. Amazon EC2 instances; <http://aws.amazon.com/ec2/instance-types/>.
2. Amazon Web Services. 2013. Amazon Elastic Block Store (EBS); <http://aws.amazon.com/ebs/>.
3. Bailis, P., Ghodsi, A. 2013. Eventual consistency today: limitations, extensions, and beyond. *ACM Queue* 11(3); <http://queue.acm.org/detail.cfm?id=2462076>.
4. Baset, S. A. 2012. Cloud SLAs: present and future. *ACM SIGOPS Operating Systems Review* 46(2): 57-66.
5. Bouchenak, S., Chockler, G., Chockler, H., Gheorghe, G., Santos, N., Shraer, A. 2013. Verifying cloud services: present and future. *ACM SIGOPS Operating Systems Review* 47(2): 6-19.
6. Cavage, M. 2013. There's just no getting around it: you're building a distributed system. *ACM Queue* 11(4); <http://queue.acm.org/detail.cfm?id=2482856>.
7. Corbett, J. C., et al. 2012. Spanner: Google's globally distributed database. *Proceedings of the 10th Usenix Conference on Operating Systems Design and Implementation*: 251-264.
8. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Voshall, P., Vogels, W. 2007. Dynamo: Amazon's highly available key-value store. *Proceedings of the 21st ACM SIGOPS Symposium on Operating Systems Principles*: 205-220.
9. Facebook. 2011. *Open Compute Project*; <http://www.opencompute.org/>.
10. Hamilton, J. 2007. On designing and deploying Internet-scale services. *Proceedings of the 21st Conference on Large Installation System Administration*.
11. Hellerstein, J. L. 2009. Engineering autonomic systems. *Proceedings of the 6th International Conference on Autonomic Computing*: 75-76.
12. Mell, P., Grance, T. 2011. The NIST definition of cloud computing. National Institute of Standards and Technology Special Publication 800-145; <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>.
13. Mian, R., Martin, P., Zulkernine, F., Vazquez-Poletti, J. L. 2012. Estimating resource costs of data-intensive workloads in public clouds. *Proceedings of the 10th International Workshop on Middleware for Grids, Clouds and e-Science*.
14. Netcraft. 2013. Amazon Web Services' growth unrelenting (May); <http://news.netcraft.com/archives/2013/05/20/amazon-web-services-growth-unrelenting.html>.
15. Ou, Z., Zhuang, H., Nurminen, J. K., Ylä-Jääski, A., Hui, P. 2012. Exploiting hardware heterogeneity within the same instance type of Amazon EC2. *Proceedings of the 4th Usenix Workshop on Hot Topics in Cloud Computing (HotCloud)*: 1-5.
16. Schad, J., Dittrich, J., Quiané-Ruiz, J.-A. 2010. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of the Very Large Data Base Endowment* 3(1-2): 460-471.
17. Storage Performance Council. 2013. *SPC Specifications*; <http://www.storageperformance.org/specs>.
18. Xu, Y., Musgrave, Z., Noble, B., Bailey, M. 2013. Bobtail: avoiding long tails in the cloud. *Proceedings of the 10th Usenix Conference on Networked Systems Design and Implementation*: 329-342.

LOVE IT, HATE IT? LET US KNOW

feedback@queue.acm.org

JASON LANGO is co-founder and CTO of Bracket Computing, an enterprise cloud computing company that he started while he was Entrepreneur in Residence at Sutter Hill Ventures. He has focused his career on enterprise computing, storage, security, and operating systems. Jason was Principal Engineer at Cisco (through IronPort Systems' acquisition) where he was lead architect for the Web security product line. Before that he spent over 7 years at NetApp as a senior engineer overseeing file system performance and proxy caching technology. He began his career at SGI working on large-scale process and thread scheduling in their original UNIX kernel (IRIX). Jason blogs at <http://lastbusinessmachine.com>.

© 2013 ACM 1542-7730/13/1100 \$10.00