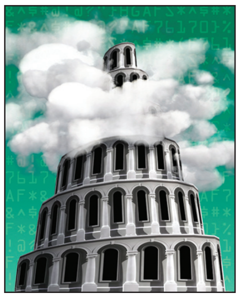

The AppScale Cloud Platform

Enabling Portable, Scalable Web Application Deployment

Chandra Krintz • *University of California, Santa Barbara*

AppScale is an open source distributed software system that implements a cloud platform as a service (PaaS). AppScale makes cloud applications easy to deploy and scale over disparate cloud fabrics, implementing a set of APIs and architecture that also makes apps portable across the services they employ. AppScale is API-compatible with Google App Engine (GAE) and thus executes GAE applications on-premise or over other cloud infrastructures, without modification.

As compute power, disk storage, and high-end network communication costs plummet, cloud computing has emerged to provide intuitive, utility-style access to vast pools of resources (compute, storage, networking, and software services). Although such processing power is cheap and readily available, accessing it from cloud infrastructure providers via infrastructure as a service (IaaS) currently requires significant expertise, experience, and time to customize, configure, deploy, and manage virtual machines (VMs).

Recent advances in platform-level cloud computing (platform as a service, or PaaS) have significantly simplified cloud use by giving developers complete software/runtime stacks (versus the self-service VMs of IaaS) on which to execute their Web-accessible applications (apps) and services. PaaS systems offer programmatic access to scalable, distributed, and fault-tolerant cloud services, which eliminates the need for developers to write or deploy their own, and lets them focus on innovation. Cloud platform application services typically include key-value, relational, object, or blob data storage, data caching, email and messaging, authentication, monitoring, resource/service acquisition, background tasking, and data analytics technologies, among others. Extant PaaS systems automatically fully or partially configure, deploy, and scale the apps and services they execute. Unfortunately, given the current state of the art in PaaS systems, a key barrier to their widespread use remains: lock-in to a particular cloud system or app service implementation. We can address this portability problem with a new cloud platform called AppScale.

The Portability Problem

Public PaaS systems such as Google App Engine (GAE), Microsoft Azure, Amazon Elastic Beanstalk, and VMWare CloudFoundry all offer similar cloud service ecosystems for app use. Unfortunately, they do so via different APIs and language bindings, scale and service-level guarantees, performance levels, pricing models, and standards, rules, and restrictions to which apps must comply. The sheer multitude of offerings and options makes it challenging for new and expert software developers to determine which set of services is best for their apps, for some definition of "best" (price, performance, scale, configurability, familiarity, ease of use, and so on). Moreover, once users choose a platform, code their app to its service interfaces, and configure it for that system, they become locked in to both the public cloud fabric and to the service implementation the platform chooses to export. This lock-in occurs because changing, even to a similar service or platform from a different provider, requires the developer to exert significant porting effort (that is, code changes in the app).

Surprisingly, this lack of portability across popular app services also occurs even for open source technologies. With cloud services' popularity, a vast diversity of open source alternatives have emerged that implement service functionality. For example, the immensely popular key-value (informally dubbed NoSQL) datastores, initially described and used by Google and Amazon for their internal "big data" concerns, are now widely available as open source. Currently, more than 200 NoSQL options offer a similar overall service for apps, but they differ in one or more characteristics, such as their programming interfaces (APIs), query language and query support, deployment topology (master/slave versus peer-to-peer), performance and scaling behavior, data consistency policies, replication policies, and programming language bindings, among others. Moreover, each service alternative has a unique methodology for its configuration and deployment in a distributed setting, which imposes overheads and learning curves on developers and system administrators because each is a complex system, performance is sensitive to configuration, and software updates are frequent.

A similar diversity in offerings is available for other app services, including authentication, MapReduce data analytics, full text search, multitasking and distributed coordination, caching, SQL databases, and messaging. Although open source app services offer developers more implementation choices than public cloud systems (which choose, limit, or restrict implementations for scaling and management purposes), the choice of any single implementation also leads to code lock-in. As for cloud fabrics and proprietary systems, it's difficult to know in advance which option is best for a particular app or workload, and moving between service implementations, even if the app is designed to do so, consumes time, lines of code, and programmer focus that could instead be used for the core innovation.

Our work at the University of California, Santa Barbara, addresses these portability limitations that extant cloud systems and app services impose with new research and technology. In particular, we attempt to reduce lock-in and encourage and facilitate broader use (and thus investigation) of these systems for deploying current and future Web-based and data-intensive apps. Toward this end, we've designed, developed, and released as open source the AppScale cloud platform (http://appscale.cs.ucsb.edu). AppScale is a software infrastructure that implements a PaaS cloud to exploit the benefits that such systems offer (ease of use through a simplified deployment and execution model, and automatic deployment, configuration, and elastic scalability).

AppScale differs from other PaaS offerings in three primary ways. First, AppScale executes automatically over multiple IaaS clouds (on-premise and public), providing "write-once, deploy-anywhere" functionality for cloud applications that execute over it. Second, AppScale implements the programming model and APIs of the de facto PaaS public cloud standard: GAE (http://code.google.com/appengine/docs/whatisgoogleappengine.html). Applications that execute over App Engine also execute over AppScale without modification, extending app portability to multiple IaaS and PaaS cloud fabrics. Third, the AppScale software architecture integrates (and automatically configures, deploys, and scales) multiple alternatives for each service or API its apps use within a given cloud instance.

Such choice precludes application/code lock-in to any particular cloud system (IaaS or Paas) or service implementation (NoSQL, SQL, tasking, messaging, authentication, and so on). Moreover, AppScale lets users experiment with these different technologies with a low barrier to entry, and gives cloud researchers a platform that facilitates the investigation of cloud technologies using a diversity of real apps and integrated technologies.

Although open source app services offer developers more implementation choices than public cloud systems, the choice of any single implementation also leads to code lock-in.

AppScale

Figure 1 depicts the design of the AppScale system. Our AppScale approach is unique in that we fully emulate GAE by implementing the App Engine programming model and APIs both to bring successful public cloud technologies on-premise and under developer control, and to export these APIs via a plug-in software architecture that decouples the app from its service implementation so that its implementation (and underlying cloud fabric) can be swapped out without changing the app.

GAE is a public cloud platform (distributed Web service stack) that hosts more than 1 million active apps today within Google's data centers (http://googleappengine.blogspot.com/2011/05/year-ahead-for-google-app-engine.html). This widespread use and uptake has resulted from the programming model that App Engine implements, which reflects a

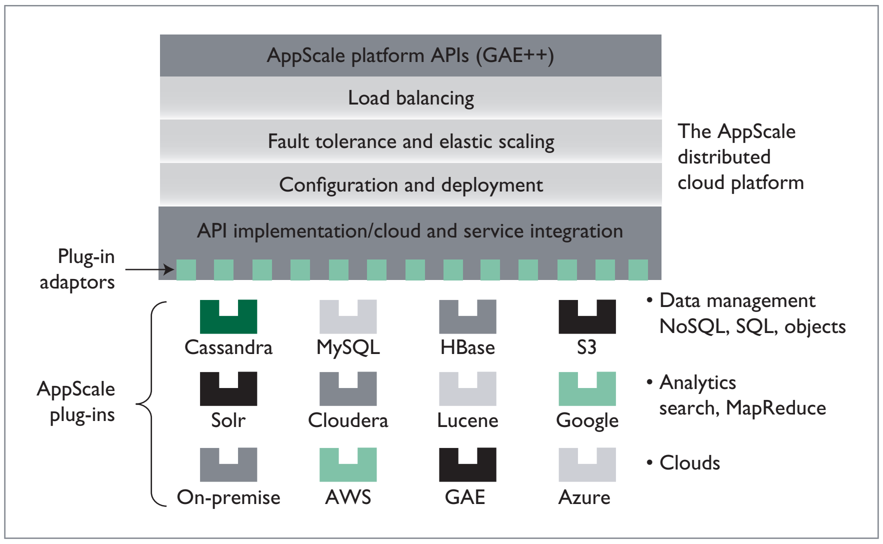
AppScale platform APIs (GAE++)
Load balancing
Fault tolerance and elastic scaling
Configuration and deployment
API implementation/cloud and service integration
The AppScale distributed cloud platform
Plug-in adaptors
AppScale plug-ins
Cassandra
MySQL
HBase
S3
Solr
Cloudera
Lucene
Google
On-premise
AWS
GAE
Azure
• Data management NoSQL, SQL, objects
• Analytics search, MapReduce
• Clouds

Figure 1. Design of the AppScale cloud platform. AppScale implements a multitier distributed Web service stack with automatic deployment, load balancing, and scaling, along with API adaptors for alternatives for each service API. Developers or systems operators deploy AppScale over virtualized cluster resources or cloud infrastructures and then upload their apps to the platform, using an AppScale Web service or command line toolset.

set of best practices for intuitive and expedited Web service development, as observed by Google engineers. The programming model extracts and provides "as-a-service" application support technologies that are common across a wide range of Web applications. Developers incorporate the services (data storage, communications, authentication, tasking, and so on) into their apps using a simple and intuitive API for each. This model lets developers focus on app innovation rather than on the ancillary support on which their apps rely. GAE automatically instantiates, scales, and manages faults for the app as well as its service ecosystem and isolates apps using sandbox support via high-level languages (Java, Python, and Go) and their runtimes.

Programming Model and APIs

AppScale implements the App Engine programming model for Web-based application development by implementing each API that GAE defines and supporting all the GAE programming languages. That is, AppScale is API-compatible and emulates GAE's fully distributed behavior — on-premise or over an IaaS cloud instead of on Google's resources — so that any app that executes over GAE also executes over AppScale, without code modification.

API-compatibility with GAE lets us engender a large and growing user community, provide developers with access to extant applications, and investigate the potential implementations of, and extensions to, public cloud systems using open source technologies. Developers code their applications to this set of APIs to access each of the ancillary services the app requires. AppScale, like App Engine, implements each of these APIs and then automatically deploys, manages, and scales the apps along with their service ecosystems.

To implement the services the GAE APIs export, AppScale provides a software framework into which we plug multiple, competitive implementations of each service (open source or proprietary). Each AppScale cloud instance plugs in a single service for each API, but users can choose from multiple alternatives upon deployment. This support lets users easily compare and contrast different service alternatives with their apps and workloads. For example, for the Datastore API, AppScale integrates more than a dozen different plug-in alternatives, including those for Cassandra, HBase, Hypertable, Redis, MySQL Cluster (which we employ as a key-value store), and SimpleDB (an Amazon public cloud datastore service). Apps that use the Datastore API can use any one of these plug-ins simply by executing over a different AppScale cloud instance (AppScale supports moving data if needed). As part of this plug-in support, the AppScale platform automatically configures, deploys, scales, and manages any faults of the service plug-ins to relieve both developers and cloud administrators of this significant burden.

Additional Features

The AppScale plug-in architecture also lets us export many other features and cloud technologies to app developers. In particular, AppScale efficiently profiles and extends its integrated services in a technology-independent way. For example, we provide a limited form of ACID transaction semantics across Datastore API plug-ins. We also integrate and export public cloud services as API plug-ins from popular cloud fabrics including Amazon Web Services (AWS), Google cloud technologies, and Microsoft Azure. We currently integrate these technologies via plug-in implementations for NoSQL, SQL, and unstructured storage. We also go beyond the GAE APIs and leverage the AppScale architecture's extensibility to provide access to VM control (start, stop, and monitoring), background tasking of arbitrary programs or scripts, and support for other services that are increasingly important for data-intensive apps, such as MapReduce (via Hadoop) and statistical analytics.

The AppScale platform also provides the scalability, ease of use, and high availability that users have come to expect from public cloud platforms and infrastructures. This includes elasticity and fault detection/recovery,[1] authentication and user control, monitoring and logging, cross-cloud data and application migration,[2] hybrid cloud multitasking,[3] and offline analytics and disaster recovery.[2,4] In particular, we couple elasticity and fault tolerance to start/stop platform components within and across VMs, and we ultimately rely on Apache Zookeeper — which we employ for distributed coordination and state management — for system survivability.

AppScale Deployment and Use

We release AppScale as a single VM and easy-to-use Web-based toolkit, which automatically deploys an AppScale cloud using one or more VM instances. Each instance implements one or more AppScale components and services. An AppScale cloud integrates and automatically deploys various open source technologies that facilitate its functionality, including Apache Zookeeper (for fault-tolerant distributed coordination), RabbitMQ (for distributed multitasking), ejabberd/Strophe.js (for messaging and channel communication), Lucene (for full text search), sendmail, memcached (for distributed caching), Hadoop MapReduce, R analytics, the network file system, and the Hadoop distributed file system, among others.

Cloud configuration and elasticity is automatic but can be informed by user preferences if desired. As mentioned previously, our tools deploy AppScale over extant IaaS fabrics; we currently support on-premise deployments over Eucalyptus[5] and public cloud deployments over Amazon Elastic Compute Cloud (EC2). Cloud administrators supply their credentials to the AppScale tools to initiate automatic deployment and then manage developers' cloud use. Developers log into an AppScale cloud to upload their apps. The AppScale cloud then executes and automatically scales the apps on the developers' behalf. Because AppScale can also run as a single VM instance, developers can run AppScale locally over virtualization to test and debug their applications prior to cloud deployment (AppScale or App Engine).

In summary, AppScale is an extensible and freely available distributed cloud platform that facilitates simplified development, automated deployment, and empirical investigation of cloud apps and their service ecosystems. AppScale enables applications written in high-level languages to execute via AppScale over different cloud fabrics and to employ a vast diversity of application service implementations without modification. Such portability enables novice and expert developers alike to quickly and easily develop GAE apps that implement interesting Web service and data analytic applications and use extant and emerging cloud systems without requiring them to become experts at the underlying technologies or locked in to any particular cloud or service implementation their apps use. Additional information on AppScale and directions for download and use are available at http://appscale.cs.ucsb.edu.

Acknowledgments

This work wouldn't be possible without an incredible group of students, led by Chris Bunch and Navraj Chohan, with whom I have had the privilege to work on this project over the past three years. This work was funded in part by Google, IBM, US National Science Foundation grants CNS-0546737, CNS-0905237, CNS-1218808, and the US National Institutes of Health grant 1R01EB014877-01.

References

1. C. Bunch et al., "A Pluggable Autoscaling Service for Open Cloud PaaS Systems," *Proc. IEEE/ACM Int'l Conf. Utility and Cloud Computing*, 2012.
2. N. Chohan et al., "North by Northwest: Infrastructure Agnostic and Datastore Agnostic Live Migration of Private Cloud Platforms," *Proc. 4th Usenix Workshop Hot Topics in Cloud Computing* (HotCloud 12), Usenix Assoc., 2012; www.cs.ucsb.edu/~ckrintz/papers/hotcloud12.pdf.
3. C. Bunch et al., "Language and Runtime Support for Automatic Configuration and Deployment of Scientific Computing Software over Cloud Fabrics," *J. Grid Computing*, vol. 10, no. 1, 2012, pp. 23–46.
4. N. Chohan et al., "Hybrid Cloud Support for Large-Scale Analytics and Web Processing," *Proc. 3rd Usenix Conf. Web Application Development* (WebApps 12), Usenix Assoc., June 2012.
5. D. Nurmi et al., "The Eucalyptus Open-Source Cloud-Computing System," *Proc. IEEE Int'l Symp. Cluster Computing and the Grid*, 2009; http://open.eucalyptus.com/documents/ccgrid2009.pdf.

Chandra Krintz is a professor of computer science at the University of California, Santa Barbara, and is CTO and cofounder of AppScale Systems. Her research interests include cloud computing, compilers and runtime systems, dynamic and adaptive optimization, high-performance computing, resource-aware Internet and embedded systems, and broadening participation in computing. Krintz has a PhD in computer science from the University of California, San Diego. Contact her at ckrintz@cs.ucsb.edu.