TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 01/11 pp 1–9
Volume 22, Number 1, February 2017

Towards a Full-Stack DevOps Environment (Platform-as-a-Service) for Cloud-Hosted Applications

Zhenhua Li, Yun Zhang, and Yunhao Liu*

Abstract: If every programmer of cloud-hosted apps possessed exceptional technical capability and endless patience, the DevOps environment (also known as Platform-as-a-Service, or PaaS) would perhaps become irrelevant. However, the reality is almost always the opposite case. Hence, IT engineers dream of a reliable and usable DevOps environment that can substantially facilitate their developments and simplify their operations. Current DevOps environments include Google App Engine, Docker, Kubernetes, Mesos, and so forth. In other words, PaaS bridges the gap between vivid IT engineers and stiff cloud systems. In this paper, we comprehensively examine state-of-the-art PaaS solutions across various tiers of the cloud-computing DevOps stack. On this basis, we identify areas of consensus and diversity in their philosophies and methodologies. In addition, we explore cutting-edge solutions towards realizing a more fine-grained, full-stack DevOps environment. From this paper, readers are expected to quickly grasp the essence, current status, and future prospects of PaaS.

Key words: cloud computing; Platform-as-a-Service (PaaS); DevOps; development; operation; environment

1 Introduction

Recent years have witnessed fundamental changes to the IT industry via the emergence of cloud computing. Cloud computing is widely recognized as having three layers: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Among these three layers, IaaS systems (e.g., Amazon Web Services and Microsoft Azure) offer basic infrastructure utilities like Virtual Machines (VMs) and object storage, and have received the most attention from the IT market. As such, IaaS standardization has matured. In contrast, there are a diversity of SaaS systems available on line (e.g., Salesforce, Gmail, and Office 365). As SaaS systems are closest to the end users, most Internet users are familiar with them. In between IaaS and SaaS are PaaS systems (e.g., Google App Engine, Google Borg, and Docker), which connect the IaaS and SaaS systems. Although many cloud users are unaware of the existence of PaaS, it has recently been perceived as increasingly important by cloud developers and operators[1]. In addition, an increasing number of startup companies are currently focusing on PaaS projects and markets.

From a technical perspective, PaaS systems are understood to be DevOps (an abbreviation of development and operations) environments for cloud-hosted applications. In other words, PaaS is expected to provide efficient tools that help IT engineers quickly build cloud apps. Cloud computing was originally introduced to free programmers from the burden of having to attend to trivial details with respect to servers, switches, and routers. In practice, however, the performance of IT engineers continues to suffer due to the technical complexities of cloud computing.

Cloud computing spans almost every area of computer science and technology, and thus involves

• Zhenhua Li, Yun Zhang, and Yunhao Liu are with the School of Software, Tsinghua University, Beijing 100084, China. E-mail: {lizhenhua1983, hitzhangyun, yunhaoliu}@gmail.com.
∗ To whom correspondence should be addressed.
Manuscript received: 2016-11-13; revised: 2016-12-07; accepted: 2016-12-08

full-stack techniques. Yet proficient full-stack engineers are rare in the job market; most engineers have expertise in only one tier in the stack, e.g., web services. As a result, cloud-computing engineers must deal with complicated configurations, deployments, and performances, with regard to hardware, Operating System (OS), container (e.g., cgroups[2], Docker, and Rocket[3]), database (DB), the Transmission Control Protocol/Internet Protocol (TCP/IP), the Domain Name System (DNS), and firewall, to name a few. Hence, IT engineers dream of one day having a reliable and usable DevOps environment to substantially facilitate their developments and simplify their operations, i.e., to bridge the gap between vivid IT engineers and stiff cloud systems, as demonstrated in Fig. 1.

Specifically, as demonstrated in Fig. 2, a full-stack DevOps environment (or PaaS) for cloud-hosted apps must provide at least the following functions: (1) app coding, building, and testing, (2) VM/container monitoring, (3) resource management, (4) task scheduling, (5) concurrency coordination, (6) system-log analysis, and (7) information visualization. Furthermore, there are a few advanced and extended functions such as RESTful (where REST stands for REpresentational State Transfer) Application Programming Interfaces (APIs), team collaboration, service integration, security protection, and privacy preservation. In a nutshell, PaaS summarizes and refines the valuable experiences and concerted efforts of pioneer cloud-computing engineers with respect to application developments and operations. The key goal is to provide IT engineers with a powerful and perceptive assistant, such that they can rapidly launch cloud-hosted apps that possess multiple desirable properties, including extensibility, scalability, reliability, security, concurrency, and cost efficiency.

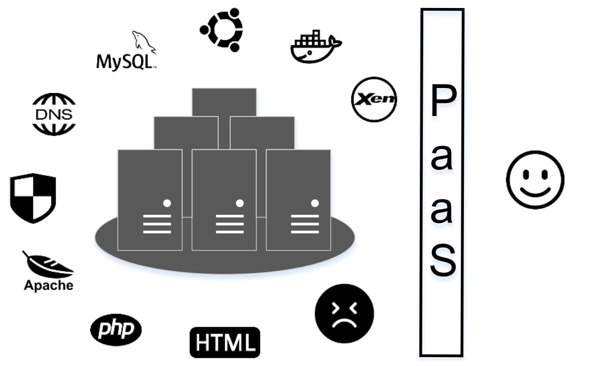


Fig. 1 PaaS bridges the gap between vivid IT engineers and stiff cloud systems.

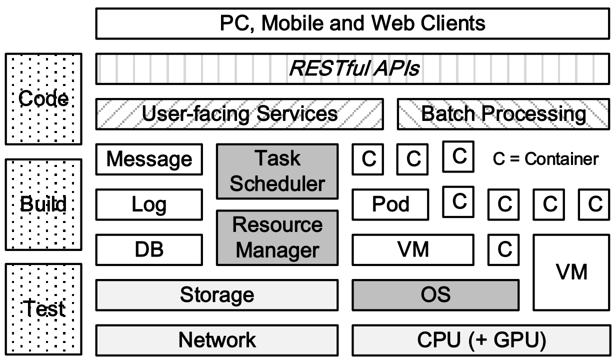


Fig. 2 A simplified architectural overview of today's cloud-computing DevOpS stack.

This paper is intended to serve as an easy-to-follow introduction to PaaS, as well as a concise survey of existing and newly emerging PaaS techniques. Readers need NOT be cloud computing professionals. First, we examine a comprehensive list of state-of-the-art PaaS solutions (e.g., Google App Engine, Google Borg, AWS Elastic Beanstalk, Microsoft Azure Stack, OpenStack Horizon and Magnum, VMware Cloud Foundry, Docker, Puppet, ZooKeeper, and Hadoop YARN) across various tiers of the cloud-computing DevOps stack in Section 2. On this basis, we identify areas of consensus and diversity in their philosophies and methodologies in Section 3. In addition, we explore a number of cutting-edge solutions (e.g., Google Kubernetes, Apache Mesos, Heroku, and our own recently developed Cloud Studio) towards a more fine-grained, full-stack DevOps environment in Section 4. Finally, we make our concluding remarks in Section 5.

2 State-of-the-Art PaaS Solutions

By "state-of-the-art" we mean that a PaaS solution is either popular in its user base or representative of the latest technical methodology. We first examine the PaaS solutions of the three international market giants—Google, Amazon Web Services (AWS), and Microsoft Azure. Then, we present the open-source solutions from OpenStack, VMware, and dotCloud. Finally, we introduce three classical tools for large-scale system configuration, coordination, and resource management, i.e., Puppet, ZooKeeper, and Hadoop YARN.

Google App Engine (GAE). The key advantage of GAE lies in its significant simplification of the development and deployment of web services. With its specially designed and implemented Software Development Kit (SDK) libraries for cloud applications, GAE offers a series of useful functions that can enable cloud apps to rapidly get online. These functions include automatic resource scaling, distributed caching, task and message queuing, reliable data storage, and so forth. Moreover, GAE provides support for clients to simultaneously access multiple versions of an app. Client requests (or workloads) can be automatically routed (or forwarded) to different app versions according to specific settings of the app developers, thus facilitating the A/B or split testing commonly used in the development of market strategies.

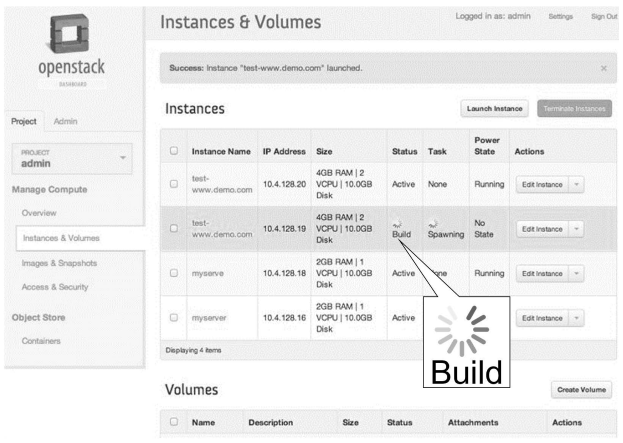
Google Borg. While this powerful, fundamental "efficiency weapon" has been used in most of Google services, Borg was kept secret until 2015 when the paper entitled "Large scale cluster management at Google with Borg" was published in the ACM EuroSys'15 Conference Proceedings[4]. This publication clearly indicates the vital importance of Borg to Google, since the company's best ideas and systems are seldom released to the public. Borg targets efficient management and high utilization of large-scale distributed server clusters. Specifically, Borg is leveraged to run hundreds of thousands of jobs for thousands of Google services/applications. It spans a number of clusters, and each cluster consists of up to tens of thousands of physical servers.

To substantially enhance the resource utilization of each physical server, Borg makes three critical choices. First, Borg does not employ VMs, which can severely degrade the working efficiency of physical servers. Instead, it utilizes lightweight Linux containers (LXC, or cgroups). Second, all jobs running on Borg are classified as one of two kinds of heterogeneous workloads, namely end-user-facing services (or long-running services) and batch jobs, for the application of different resource-management and task-scheduling strategies. Third, apps are deployed in physical servers in a mixed manner, i.e., multiple logically isolated apps can run on the same server, whereas traditionally one app runs on a dedicated and exclusive cluster of servers.

AWS Elastic Beanstalk and others. Being both the inventor and dominant force in the cloud computing market, AWS provides the Elastic Beanstalk system to help developers rapidly and conveniently deploy and manage applications on top of AWS. In a sense, the AWS Elastic Beanstalk scheme is similar in appearance to that of GAE. In fact, there are many more PaaS tools offered by AWS than Elastic Beanstalk. For example, AWS has released several tools for app developers, such as CodeCommit, CodeDeploy, and CodePipeline, as well as tools for system operators, such as CloudWatch, CloudFormation, CloudTrail, Config, Console, OpsWorks, and Service Catalog.

Microsoft Azure Stack. Despite its somewhat late entry into the cloud computing market, Microsoft Azure has firmly established itself in second place behind AWS. It explicitly positions itself as a full-stack DevOps environment for cloud-hosted apps. From a pure PaaS perspective, there are at least a few tens of services/systems worthy of attention in the Microsoft Azure Stack. For instance, Azure supplies multiple tools for the developers: Visual Studio Team Services, Visual Studio Application Insights, DevTest Labs, Xamarin (for faster creation of cloud-powered mobile apps), and Storage Explorer, to name a few. In addition, Azure provides a few useful monitoring and management tools: Azure Resource Manager, Scheduler, Log Analytics, Automation, Site Recovery, etc.

OpenStack Horizon and Magnum. As the de facto standard of open-source cloud computing solutions, OpenStack has also made inroads into the PaaS layer. Since its inception, OpenStack has provided Horizon, an information or resource visualization tool (also known as DashBoard) for cloud computing systems. As shown in Fig. 3, with OpenStack Horizon a user can clearly see how many servers s/he owns, the level of resources in every server, and how well every server is operating. Meanwhile, OpenStack Horizon behaves as a coarse-grained console for system resource management, thus enabling convenient one-click start up, shut off, stand by, and hibernate operations for users. Recently, the OpenStack community released



Success: Instance "test-www.demo.com" launched.

Instance Name	IP Address	Size	Status	Task	Power State	Actions
test-www.demo.com	10.4.128.20	4GB RAM | 2 VCPU | 10.0GB Disk	Active	None	Running	Edit Instance
test-www.demo.com	10.4.128.19	4GB RAM | 2 VCPU | 10.0GB Disk	Build	Spawning	No State	Edit Instance
myserve	10.4.128.18	2GB RAM | 1 VCPU | 10.0GB Disk	Active	None	Running	Edit Instance
myserver	10.4.128.16	2GB RAM | 1 VCPU | 10.0GB Disk	Active			Edit Instance

Name	Description	Size	Status	Attachments	Actions

Fig. 3 Snapshot of OpenStack Horizon. Server operation progress is indicated by a coarse-grained rotating circle.

a novel tool called Magnum to effectively support the running of containers.

VMware Cloud Foundry. The well-recognized virtualization company VMware has until recently seemed irrelevant in relation to "open source" computing (otherwise, the open-source VirtualBox project would likely not have emerged). Nonetheless, in April 2011 to many people's surprise, VMware released the first open-source full-fledged PaaS solution known as Cloud Foundry. Based on Ruby on Rails, Cloud Foundry acts as a distributed system composed of multiple independent sub-systems that communicate with each other through message passing. Using an identical set of code bases, this system can be deployed on a mega data center, a couple of personal computers, or a bundle of public-cloud VMs. Those familiar with the design philosophy of OpenStack may have noticed considerable similarity between Cloud Foundry and OpenStack.

Docker. Since it has gained tremendous popularity in a short period of time, most people who have heard of Docker are not aware of the company—dotCloud—that created it. As the hottest technique in the cloud computing market in the last few years, Docker allows applications to run within separate (i.e., logically isolated) containers by leveraging a series of Linux kernel features like cgroups and namespace. Since many containers share the same Linux kernel, the resource utilization and working efficiency of Docker are much higher than those of VMs. The design philosophy of Docker clearly resembles that of Google Borg. In the IT industry, efficiency and compatibility are often at odds, which is also the case in Docker. Despite its efficiency, Docker has a few shortcomings with respect to compatibility, such as not supporting 32-bit architectures or Windows servers. Moreover, increased sharing of Docker resources results in decreased isolation of its containers, such that security concerns have been raised at times.

Puppet and ZooKeeper. For the user who has purchased and maintained 100 VMs from AWS EC2, it is often necessary to perform the same or a similar set of operations on these VMs. Instead of inputting and running hundreds of repeated Shell commands, users can consider writing a short Puppet script (e.g., 100-VM.pp). Puppet is a classical centralized configuration management system that is applicable to Linux, Unix, and Windows platforms. Via Puppet, system administrators or operation staff can easily manage massive repeated, trivial configuration details. Puppet uses a simple client/server (C/S) architecture to organize system nodes, so that average IT engineers can use it smoothly and without much difficulty. However, the centralized C/S architecture can become a system bottleneck.

ZooKeeper has been a popular tool in recent years for distributed coordination and configuration management. Although it is more complicated than Puppet, it is simpler than Paxos[5]. It provides a number of desired functions in distributed systems, such as distributed lock, message queue, master node election, and dynamic configuration. In the scientific theory of distributed coordination, the ideal scheme is the Paxos protocol designed by Leslie Lamport, and Paxos has been practically implemented in the Google Chubby system. However, Paxos is very complicated due to its comprehensive considerations of various possible situations. As an everyday real-world distributed system, Paxos is often regarded an "over-provision". To this end, although ZooKeeper follows the basic principles of Paxos, it simplified its implementation in the real world by establishing a usable and effective ZooKeeper Atomic Broadcast (ZAB) protocol. Currently, many PaaS systems use ZooKeeper, such as the resource manager of Apache Mesos and the message queue of Apache Kafka.

Hadoop YARN. YARN is the abbreviation for "Yet Another Resource Negotiator", so it was clearly developed for Hadoop's resource management. Why is it "Yet Another"? Because prior to the advent of YARN, Hadoop's original resource manager had limited scalability (e.g., a Hadoop server cluster could barely accommodate more than 4000 nodes) and unsatisfactory resource utilization. Motivated by these difficulties, the YARN developers designed a two-layer resource scheduler, i.e., ResourceManager + ApplicationMaster, whereby scheduling strategies for specific tasks are determined by corresponding applications (i.e., the so-called ApplicationMaster) rather than a single entity. These changes remarkably enhanced Hadoop's scale limitations and resource utilization.

3 Consensus and Diversity

Based on the above state-of-the-art PaaS solutions, we identified a number of common principles that are consistent in most of them. Below we list a few key points of consensus in the development and operation of cloud-hosted apps:

- **No one PaaS solution fits all scenarios or meets all requirements,** even if it is full-fledged or spans the full stack. As such, cloud app developers and operators must develop specific solutions for specific scenarios, rather than strive to produce a single solution panacea.
- **Usability and reliability dwarf other performance metrics in practice.** Although academic solutions (e.g., Spark) can seem to greatly outperform industrial solutions (e.g., Hadoop), the former is not likely to replace the latter any time soon. In real-world scenarios, user interfaces, security protections, code reviews, debugging difficulties, and the maturity of the community are often the primary factors.
- **Neither VMs or containers function as actual physical machines.** Almost all VM providers (e.g., AWS EC2 and Aliyun ECS) caution that VMs be used only in a stateless manner, i.e., make every attempt not to store persistent data in a VM because it can easily crash or even disappear at any time[6, 7]. There is no doubt that containers are even more fragile.
- **Container is unlikely to grab the entire market of VM,** so a dynamic balance will likely remain between the market shares of each. Generally speaking, containers are associated with efficiency and utilization, while VMs have better compatibility and isolation. In fact, running containers in a VM may well become a popular paradigm for developing and operating cloud-hosted applications.
- **Service Level Agreements (SLAs) essentially matter more to customers than vendors.** SLAs are not laws, and even many laws are not followed by people in certain situations. Moreover, few cloud customers could precisely define and measure the performance metrics listed in SLAs. Most customers never measure their relevant performance metrics. Thus, for a cloud customer, SLAs need not be taken too seriously in discussions with a cloud vendor, but can rather be used as a reference and guide.
- **Public clouds facilitate startups, while private clouds benefit industry veterans.** For a startup company, using AWS EC2 + S3 + RDS can save significant time and money on infrastructure maintenance. However, a considerable number of opportunities would be lost in optimizing system efficiency, since in this case the infrastructure is a "black box". This is why in its first few years Dropbox fully relied on Amazon S3 for file content storage[8, 9], but has since migrated the vast majority of its file-content data to its private object storage cloud Magic Pocket[10].

On the other hand, we found considerable diversity in PaaS solutions in terms of their functionality, popularity, maturity, and openness. As the services provided by a given PaaS solution become more fine-grained, its restrictions on upper-layer apps increase accordingly. As such, there is currently no widely-recognized PaaS solution that can satisfy all parties' requirements. When a PaaS solution behaves well in some tier(s) or metric(s), it typically performs poorly in others. At the two extremes emerge constrained and open PaaS solutions.

A constrained PaaS solution would make full or primary use of underlying resources, i.e., computation, storage, and network resources. App developers, however, usually must follow a private/specific set of constraints in terms of data formats and APIs. At times, app developers may be confined to using certain programming languages. GAE is representative of a constrained PaaS. When programmers build apps based on GAE, they can take advantage of the techniques and resources within the Google cloud platform, but are subject to a variety of constraints such as limited libraries and language supports, HTTP-style APIs, and a lack of persistent session states.

At the other extreme, open PaaS solutions impose no invasive constraints on developers' program codes, but give app developers the freedom to preserve as much as possible their original programming languages, system frameworks, components, and containers. Heroku (detailed below in Section 4) is representative of an open solution that supports all popular programming languages as well as relative "minority" languages like Ruby.

Between these two extremes, there are a few semi-open PaaS solutions that make their source codes publicly available, e.g., OpenStack, Docker, and Cloud Foundry. For instance, to improve the openness of Docker, dotCloud makes it open-source and maintains a centralized repository[11] for all Docker users to freely upload their Docker images and easily seek other Docker images they desire.

4 Frontier Explorations

Having identified the philosophical and methodological degrees of consensus and diversity in state-of-the-art PaaS solutions, we now explore a number of frontier solutions towards a more fine-grained, full-stack DevOps environment.

Google Kubernetes. “Kubernetes” is not a common English word, so many people often use its abbreviation “K8s”. The term “Kubernetes” stems from ancient Greek and refers to a pilot or steersman. Google is said to have used this name for a subtle purpose (as depicted in Fig. 4): Given that Docker projects itself as a whale who travels the sea while carrying with containers, Kubernetes is steering the direction of this “container era”. Although introduced around just one year before this paper is being written, Kubernetes now occupies a slightly larger market share than Docker, with mighty support from several cloud computing market giants.

Currently, most people tend to regard Kubernetes as an upper-layer framework of Docker. That is to say, the Kubernetes team has forged a service-centric distributed system based on Docker, in which a service can automatically scale and diagnose itself when necessary. At the same time, in order to shed its dependence on Docker, Kubernetes now supports another competitive container technique called Rocket, which was developed by CoreOS[3], as illustrated in Fig. 5.

Kubernetes is often viewed as an open-source version of Google Borg. In addition to its open-source nature, Kubernetes has made great inroads in improving its openness. For instance, no matter which programming language is used to write a given app, the app can be directly mapped to a Kubernetes Service, which then communicates with the Internet or other Services via standard TCP-based protocols. Most notably, Kubernetes adds a novel Pod layer between the server node and the containers running on this node, as shown in Fig. 2. Multiple containers can simultaneously run in the same Pod, thereby effectively enhancing data-communication efficiency between these containers. Recently, a novel, hot “micro-service” idea has arisen in the IT industry, in which an integrated service is decomposed into multiple independent micro-services that are connected via network communications. In this sense, Kubernetes is tailored to support micro-services.

Fig. 4 The logos of Docker and Kubernetes.

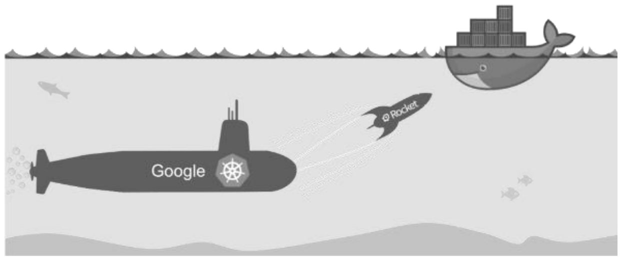


Fig. 5 Google Kubernetes is launching a “Rocket” towards Docker (illustration from Ref. [12]).

Apache Mesos. Compared with Docker and Kubernetes, Mesos is more open and has finer granularity, and as such is called “the kernel of distributed systems”. Mesos was initially developed by the famous AMPLab at UC Berkeley[13] and was then widely used in Twitter. One year after the Mesos startup, its initiator Ben Hindman, together with his UC Berkeley team members, made a visit to Twitter. At that time, only eight engineers in Twitter were present, which disappointed Ben Hindman. Nevertheless, three out of these eight engineers later joined the Mesos project, and all were former Google employees. They told Ben Hindman that they had (unfortunately) missed their chance with Google Borg, and did not want to miss Mesos and the opportunity to utilize a better method for “reconstructing” Google Borg.

To achieve more openness, Mesos features two major changes as compared with Docker and Kubernetes. First, Mesos explicitly separates resource management from task scheduling, so that applications can better acquire their desired resources. Second, Mesos provides resilient Framework interfaces to accommodate frameworks from other systems, such as Marathon[14] and Spark[15].

Furthermore, to improve resource management, the Mesos team put forward a novel resource allocation strategy they named DRF, i.e., Dominant Resource Fairness[16]. The idea for DRF originated from the observations of many IT engineers that, in the presence of multiple types of resources, an appropriate resource allocation strategy should focus on the dominant share of resources desired by users. For example, let us suppose that Mesos is simultaneously allocating resources from a physical server to two users A and B, where user A is running CPU-intensive tasks and user

B is running memory-intensive tasks. In this case, DRF would strive to allocate more CPU resources to A's tasks and more memory to B's tasks.

Heroku. Founded in 2007 and purchased by Salesforce in 2010, Heroku struggled for a few years in the cloud computing market. However, Heroku recently captured attention for its neutral, open PaaS platform. To overcome or alleviate the invasion of too many PaaS solutions to users' app codes and the resulting "cloud lock-in" problem, Heroku designed and implemented a highly portable PaaS platform that attempts to "be compatible with all mainstream PaaS solutions," as depicted in Fig. 6. In addition, through their comprehensive observations and techniques, the Heroku team has refined 12 factors necessary for wisely developing and operating cloud-hosted apps[17]. Notably in 2011, the inventor of Ruby joined Heroku as its chief architect, which reflects the positive attitude of the programming community towards Heroku.

Cloud Studio (currently in alpha phase). Almost all the mainstream PaaS systems have been developed by big companies like Google and Amazon. However, this does not mean that small companies and organizations cannot contribute to PaaS. In fact, small teams can build specific PaaS solutions to satisfy special requirements. For example, the authors of this paper (together with a few other team members at Tsinghua University) have designed and deployed an open-source PaaS system called Cloud Studio, which is publicly available at `http://thucloud.com` and soon will also be available at `http://cloudcomputing.studio`. Cloud Studio has recently released a series of useful DevOps tools for cloud-hosted apps: VirtualPool (for accommodating VMs and containers), iDashBoard (for displaying system information and facilitating user operations), Dual-Cloud Web Acceleration (especially for Google, Gmail, GitHub, and Dropbox), Cloud Disk (for hosting user files), iRecommend (for recommending professionals based on big data analysis), etc.

With respect to the iDashBoard tool, we are the first to implement a fine-grained progress bar for VM operations (see Fig. 7), which replaces the simple, coarse-grained rotating circle used in existing PaaS solutions (see Fig. 3). This implementation is non-trivial, since its accuracy heavily relies on tight coordination between the physical server, the VM, and the VM Manager (VMM). As demonstrated in Fig. 8, iDashBoard deploys a Proxy in the physical server and a Client in the VM, whereby the Proxy interacts with

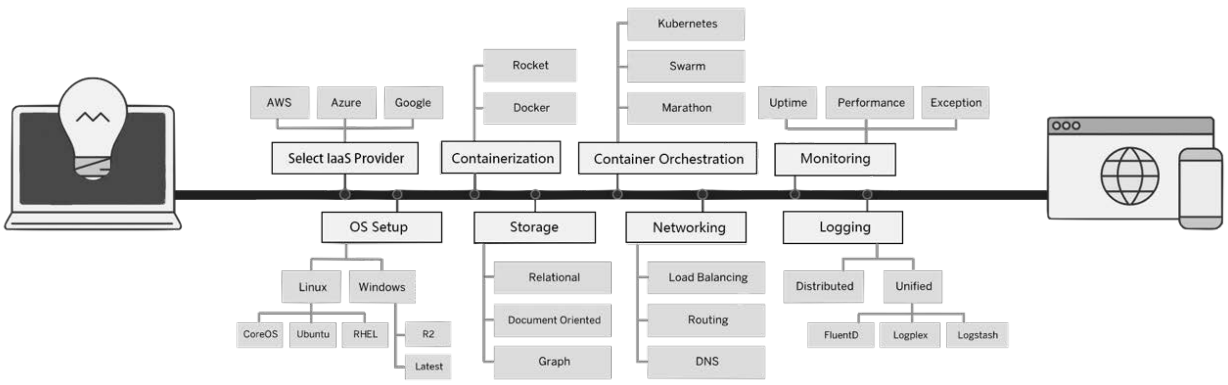
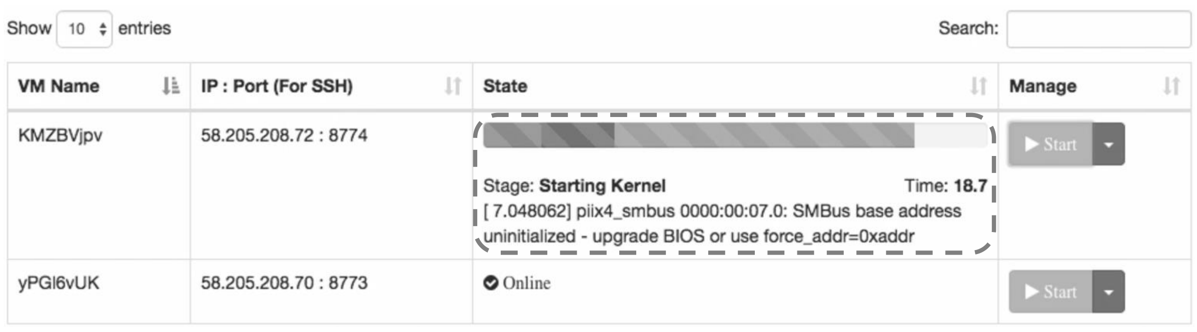


Fig. 6 Heroku aims to build a PaaS platform for all mainstream PaaS systems.


Show 10 entries Search:

VM Name	IP : Port (For SSH)	State	Manage
KMZBVjpv	58.205.208.72 : 8774	Stage: **Starting Kernel** Time: **18.7** [7.048062] piix4_smbus 0000:00:07.0: SMBus base address uninitialized - upgrade BIOS or use force_addr=0xaddr	▶ Start
yPGl6vUK	58.205.208.70 : 8773	✓ Online	▶ Start

Fig. 7 Fine-grained progress bar for VM operation implemented in iDashBoard.

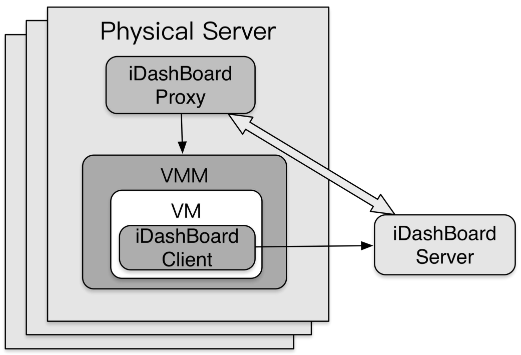


Fig. 8 Implementation of the fine-grained progress bar for VM operation, which requires tight coordination between the physical server, the VM, and the VMM.

the VMM and the Client reads the VM information. Meanwhile, the Proxy and the Client report to an external iDashBoard Server. In the near future, we plan to implement a real-time, visualized topology graph for a cloud computing system that will replace the tedious linear list used in current PaaS solutions.

5 Concluding Remarks

Today, cloud computing is a rapidly evolving industry characterized by fierce competition, which also holds true for its DevOps environment or PaaS. As yet, there is no unified set of PaaS APIs or data formats, and the three market giants (i.e., AWS, Google Cloud, and Microsoft Azure) are holding fast to their respective PaaS standards. Thus, Heroku's vision of a unified PaaS has yet to be realized. In order to develop an appropriate "sweet-spot" solution, many DevOps teams are using current state-of-the-art PaaS solutions as references and leveraging open-source PaaS-related tools to construct their own optimal PaaS solutions. Is this too complicated a process for today's IT engineers who are in need of solutions?

In our opinion, the answer is NO. As we have mentioned repeatedly, cloud computing technologies are evolving rapidly and substantially, and are thus constantly generating novel concepts and tools. For example, when IT engineers had only just mastered Hadoop (MapReduce), Spark was introduced, which reputedly outperforms Hadoop by a factor of 100. Also, when IT engineers had mastered the operations of Docker, Kubernetes and Mesos arrived to provide finer granularity and higher generality. If engineers try to master all new techniques, they will soon exhaust themselves.

Fortunately, although cloud computing has created a novel market, it involves few really novel techniques. Once we examine the cloud-computing technical stack, we find that most cloud computing efforts involve system architecture and resource management, for which the fundamental resource concerns have always been computation, storage, and networking. In other words, the key techniques in the IT industry have never changed in their essence. These include the compilation, link, load, and execution of programs; the management of CPU, memory, and disk I/O by the OS; and TCP/IP-based protocols. Wise engineers can ascertain the simple basics behind the dizzying array of novelties, and thus adapt rather than be hijacked by them.

References

[1] H. Yu, *PaaS Implementation and Operation Management*, (in Chinese). Publishing House of Electronics Industry, China, 2016.

[2] Linux Control Groups (cgroups), https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt, 2016.

[3] CoreOS is building a container runtime rkt (Rocket), https://coreos.com/blog/rocket, 2016.

[4] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, Large-scale cluster management at Google with Borg, in *Proceedings of the 10th European Conference on Computer Systems (EuroSys)*, Bordeaux, France, 2015.

[5] L. Lamport, Paxos made simple, *ACM SIGACT News*, vol. 32, no. 4, pp. 18–25, 2001.

[6] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford, Heading off correlated failures through independence-as-a-service, in *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Broomfield, CO, USA, 2014.

[7] E. Zhai, R. Chen, D. I. Wolinsky, and B. Ford, An untold story of redundant clouds: Making your service deployment truly reliable, in *Proceedings of the 9th Workshop on Hot Topics in Dependable Systems (HotDep)*, Farmington, PA, USA, 2013.

[8] Z. Li, C. Wilson, Z. Jiang, Y. Liu, B.Y. Zhao, C. Jin, Z.-L. Zhang, and Y. Dai, Efficient batched synchronization in dropbox-like cloud storage services, in *Proceedings of the 14th ACM/IFIP/USENIX International Middleware Conference (Middleware)*, Beijing, China, 2013.

[9] Z. Li, C. Jin, T. Xu, C. Wilson, Y. Liu, L. Cheng, Y. Liu, Y. Dai, and Z.-L. Zhang, Towards network-level efficiency for cloud storage services, in *Proceedings of the 14th ACM Internet Measurement Conference (IMC)*, Vancouver, Canada, 2014.

[10] Inside the magic pocket, http://blogs.dropbox.com/tech/2016/05/inside-the-magic-pocket, 2016.

[11] Docker Image Library (or Hub), http://hub.docker.com, 2016.

[12] Docker in production: The bloody battle of container orchestrators, http://blog.octo.com/docker-en-production-la-bataille-sanglante-des-orchestrateurs-de-conteneurs, 2016.

[13] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. Katz, S. Shenker, and I. Stoica, Mesos: A platform for fine-grained resource sharing in the data center, in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, USA, 2012.

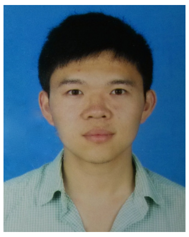
[14] Marathon: A container orchestration platform for Mesos and DCOS, https://mesosphere.github.io/marathon, 2016.

[15] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Frankli, S. Shenker, and I. Stoica, Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, USA, 2012.

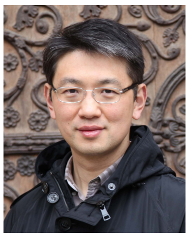
[16] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, Dominant resource fairness: Fair allocation of multiple resource types, in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, USA, 2012.

[17] The Twelve-Factor App, https://12factor.net, 2016.

Zhenhua Li is an assistant professor at the School of Software, Tsinghua University. He received the BSc and MSc degrees from Nanjing University in 2005 and 2008, respectively, and the PhD degree from Peking University in 2013, all in computer science and technology. His research areas mainly consist of cloud computing/storage, big data analysis, content distribution, and mobile Internet.



Yun Zhang is a master student at the School of Software, Tsinghua University. He received the BSc degree from the Harbin Institute of Technology in 2014. His research areas comprise cloud computing, big data analysis, and so forth.



Yunhao Liu is a professor at the School of Software, Tsinghua University. He received the PhD degree and MS degree in computer science and engineering from Michigan State University in 2003 and 2004, respectively, and the BEng degree from Tsinghua University in 1995. His research interests include distributed systems, wireless sensor networks/RFID, cyber physical systems, Internet of Things (IoT), privacy and security, and so forth. He is a fellow of ACM and IEEE.