What's Inside the Cloud?
An Architectural Map of the Cloud Landscape

Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai
FZI Karlsruhe
Haid-und-Neustr. 10-14,
76131 Karlsruhe, Germany
{alenk, klems, nimis, tai}@fzi.de

Thomas Sandholm
Hewlett-Packard Laboratories
1501 Page Mill Road,
Palo Alto, CA 94304
thomas.e.sandholm@hp.com

Abstract

We propose an integrated Cloud computing stack architecture to serve as a reference point for future mash-ups and comparative studies. We also show how the existing Cloud landscape maps into this architecture and identify an infrastructure gap that we plan to address in future work.

1. Introduction

Cloud computing is emerging as a model in support of "everything-as-a-service" (XaaS) [8]. Virtualized physical resources, virtualized infrastructure, as well as virtualized middleware platforms and business applications are being provided and consumed as services in the Cloud. Engineering software systems that use "everything-as-a-service" and which in turn may be provided as Cloud services themselves, however, requires a good understanding of the numerous emerging Cloud computing technologies as well as of already available services solutions offered in the open Cloud market.

1.1. Objective

How do the various Cloud technologies and offerings ranging from open source frameworks like Hadoop MapReduce to commercial services from Amazon and Google relate to each other? How do we compare and understand these Cloud technologies and services from a technology, software architectural, and from a business perspective?

1.2. Approach

As a first step towards answering these questions, we propose a generic Cloud computing stack that classifies Cloud technologies and services into different layers. We explain each layer through examples and demonstrate how this model helps in explaining the overall Cloud computing landscape. We further illustrate the use of the stack in modeling a Cloud computing ecosystem of various providers. The Cloud computing stack aims at facilitating communication about different Cloud technologies and services, including placing more complex offerings such as Google App Engine on the Cloud computing landscape, and at supporting the design of software systems that wish to use and compose existing Cloud technologies and services.

The different technologies and tools were categorized in our stack based on the highest-level interface offered to its users. We also recognize that some tools allow a wide range of lower-level plug points, but we focus on representing the primary use of the technology, which is most likely to happen on the highest level. Our architectural stack provides guidance about how to combine, and interchange technologies. For example, a developer at The New York Times [27] needed to transform a large number of documents and was able to obtain machines to run on from a virtual machine provisioning system (Amazon EC2 [4]) and parallelize the processing using a parallel programming framework (Hadoop MapReduce [30]). Others could learn from his example but may wish to replace either the machine provisioning system or the programming framework to meet their needs. Our categorization of existing technologies does not aim at being complete, which is impossible in the currently rapid evolution of the Cloud landscape, but rather serves as a reference point or snapshot to guide future developments. The tools we mention are for the same reason limited to the most popular or well known.

2. Reference Stack

Distributed computing technologies targeted at enterprise systems integration in the 90's such as the Open Group's DCE [29], and the Object Management Groups

CLOUD'09, May 23, 2009, Vancouver, Canada
978-1-4244-3713-9/09/$25.00 © 2009 IEEE

CORBA [28] offered programmable interfaces to overcome the complexities of remote procedure calls. Specifications such as OMA, in the CORBA case, served as reference architectures for services offered at different levels in the frameworks, e.g. vertical (industry domain specific) versus horizontal (infrastructure support) services. These categorizations simplified interoperability and promoted vendor independence. A decade later the Grid community proposed the Open Grid Services Architecture (OGSA) [20] as an effort to standardize de-facto Grid service interfaces and help adoption of Grid technologies across organizations at a global scale. The Service Oriented Architecture (SOA) inherent in the WS-I specifications [18] serve a similar purpose for the Web services B2B community.

What differs the Cloud from these earlier technologies is the "grass-root" evolution of a wide range of technologies from successful Web 2.0 enterprises such as Google, Facebook, and Amazon, and the enhanced programmability (access to APIs, RPCs, SDKs etc) offered to developers. This plethora of interfaces brings an unprecedented opportunity to systems integrators or mash-up technologists to realize their business ideas with minimal investment in infrastructure development. It can be a daunting task to navigate around the technologies in the Cloud landscape, and although various mind maps have been proposed (e.g. [57]), they provide little architectural guidance as to how the offerings may be integrated.

In the spirit of earlier distributed computing architectures we therefore propose a first architectural categorization of Cloud technologies as a stack of service types. Our stack was inspired by the "everyting as a service" (XaaS) taxonomy; Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS); recent developments of a Cloud stack for the HP, Intel, and Yahoo! Open Cirrus Cloud testbed [45]; and a survey of state-of-the-art Cloud services tools. The stack is depicted in Figure 1.

2.1. Infrastructure as a Service

On the lowest level of the infrastructure closest to the hardware we distinguish two types of services, *Physical Resource Set* (PRS) and *Virtual Resource Set* (VRS) services. Both of these service types provide a management front-end API for a set or pool of resources in order to allow higher level services to automate setup and tear-down, demand-based scalability, fail-over and operating system hosting. Primary functionality includes starting and stopping individual resources, OS imaging, network topology setup and capacity configuration. The PRS layer implementation is hardware dependent and therefore tied to a hardware vendor, whereas the VRS layer can be built on vendor independent hypervisor technology such as Xen [14] or on top of a PRS service to run in multi-vendor Clouds such as the Open Cirrus testbed. Examples of PRS services include, Emulab [33] and iLO [32]. VRS services include Amazon EC2 [4], Eucalyptus [43], Tycoon [37], Nimbus [36], and OpenNebula [53]. The reason to split these Resource Set (RS) services into two types allows automated management of physical as well as virtual resources. Furthermote some Cloud applications may incur too much overhead from running on a hypervisor but might still want to use similar automated management capabilities offered by virtual machine monitors (VMMs) and VMM management toolkits such as the VRS examples given. Another reason for demarcation is that different types of resources such as storage, network and compute node resources might need to be virtualized in different ways. However, they might still be able to have a common PRS interface.

One level higher up in the stack but still in the IaaS category we distinguish three types of *Basic Infrastructure Services* (BIS), computational, storage, and network. Some examples are MapReduce [11] (computational), GoogleFS [52] (storage), and OpenFlow [39] (network). As the highest level in the IaaS stack we consider *Higher Infrastructure Services* (HIS). Amazon's Dynamo [22], and Google's Bigtable [17] all fall into this category as they are typically built on top of BIS tools.

2.2. Platform as a Service

Moving up to the PaaS level of our integrated stack we categorize the services into *Programming Environments* and *Execution Environments*. Example of the former is Sun's project Caroline [42] and the Django framework [13], and examples of the latter are Google's App Engine [24], Joyent's Reasonably Smart [34] and Microsoft's Azure [40]. As seen by these examples an Execution Environment PaaS typically also encompasses a Programming Environment PaaS. You could potentially replace the Django framework in Google App Engine with your own Programming Environment and Microsoft Azure offers a wide range of alternative programming tools under the Azure runtime umbrella. This decoupling between execution and development environments is thus represented by having two categories in our stack model.

2.3. Software as a Service

All the applications that run on the Cloud and provide a direct service to the customer are located in the SaaS layer. The application developers can either use the PaaS layer to develop and run their applications or directly use the IaaS infrastructure. Here we distinguish between *Basic Application Services* and *Composite Application Services*. Examples of *Basic Application Services* are the OpenId [46]

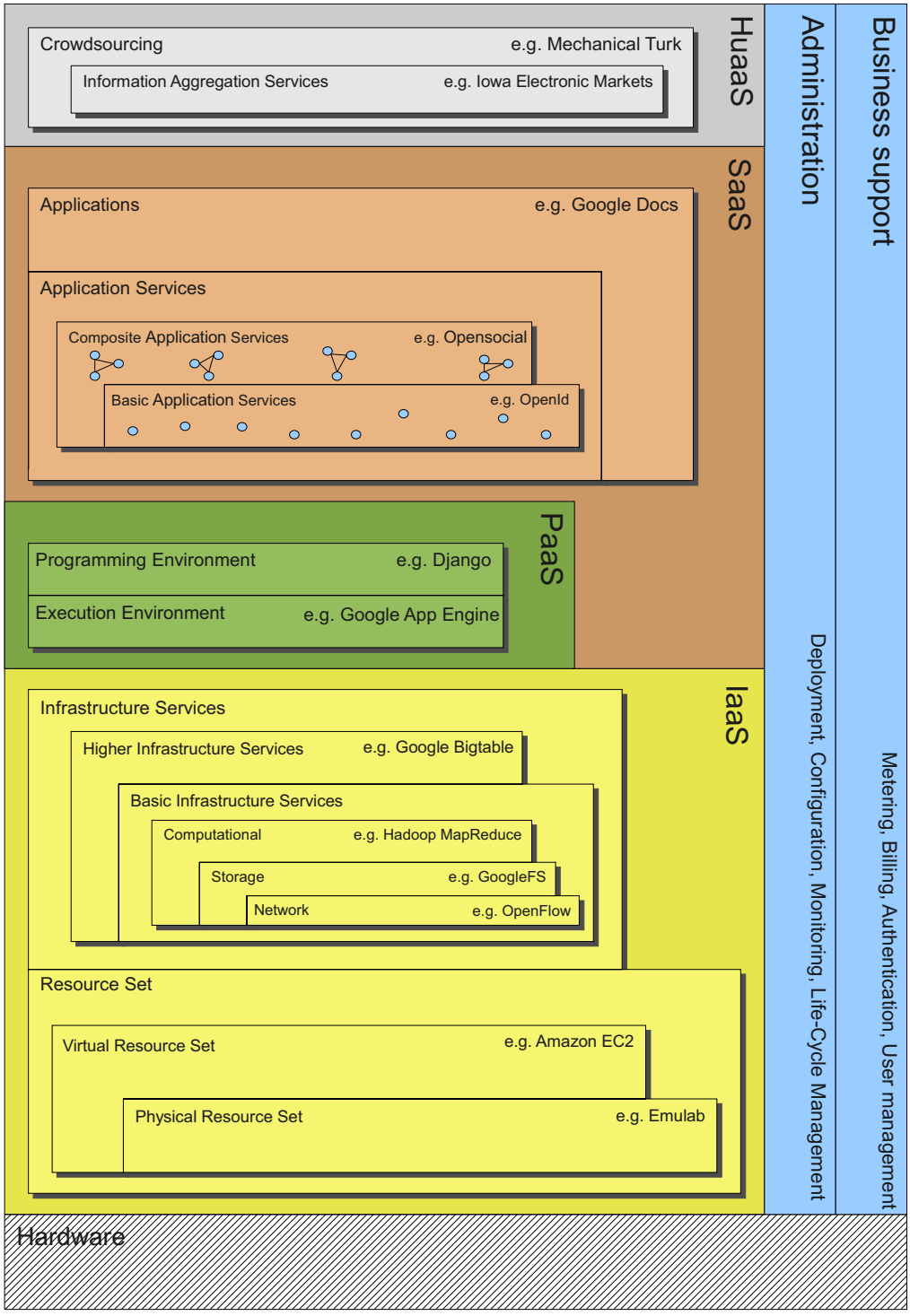

Figure 1: Cloud stack

and Google Maps [26] services. In the *Composite Application Service* category we have the mash-up support systems with *Opensocial* as the prominent example allowing entire social networks like MySpace to be used as *Basic Services*. We categorize *Basic* and *Composite* services into *Application Services*, which comprise the highest level building blocks for end-user applications running in the Cloud, such as Google Docs [25], Microsoft's Office Live [41] and OpenSocial mash-ups such as Auciti's Hangout [54].

2.4. Human as a Service

Some services rely on massive-scale aggregation and extraction of information from crowds of people. Each individual in the crowd may use whatever technology or tools he or she see fit to solve the task. We call this top-most layer in our stack *Human as a Service* (HuaaS). In some cases human intelligence is used to contribute arbitrary services, such as "newsworthy" video streams (YouTube [58]), or on-demand subtask solutions (Amazon Mechanical Turk [3]). In our stack, these tools belong to the *Crowdsourcing* (CS) category. Some human intelligence aggregation services are more controlled and more targeted at predicting events or promoting popular ideas. Examples include the Iowa Electronic Markets [9], which are mainly used to predict outcomes of political races, and Digg [12], which is used to promote popular news. We call this latter category of services *Information Aggregation Services* (IAS), since they all aim at producing a single aggregate number representing the popular opinion of the crowd in various ways, for example using market mechanisms.

2.5. Supporting services

Some services in the Cloud need to have access to all layers in the stack to present a coordinated view to its users, e.g., for debugging, accounting, monitoring and billing purposes. This is an, as of yet, largely unexplored area of research but global research testbeds such as GENI [21], the Open Cloud Consortium [44] and Open Cirrus [45] are expected to deliver solutions in this category for their users. Here we distinguish between the administrative support systems, well known from the Grid (e.g. EGEE GridICE [6]) and networking platforms (e.g. PlanetLab CoMon [47]), and business support systems for billing and federated authentication etc. In our survey we find the business support systems that span Cloud layers to be the most neglected part of the current Cloud infrastructure, and for our future work we thus plan on experimenting with mash-ups that address this shortcoming.

3 The Cloud Ecosystem

A rich ecosystem of Cloud computing services and providers has emerged, forming a complex environment in which Web-scale end-user applications and services are developed, tested, deployed and operated. While this complex service environment provides many choices, at the same time it poses a great challenge to engineers in charge of building resilient application architectures. The growing diversity of services, frameworks, platforms and tools within the Cloud computing community tends to obfuscate reasonable use and combination of advertised offerings. Our Cloud computing reference model serves as a means to categorize existing Cloud computing services on the basis of distinct service features. Thereby our reference model assists the design of application architectures that utilize and combine multiple Cloud computing services.

Research analysts of Merrill Lynch identified 10 companies with exposure to the Cloud, as well as a number of other promising Cloud computing service providers and enablers [49]. Amazon is perceived as one of the major players in the business, offering a wide range of prominent Cloud computing services such as Elastic Compute Cloud (EC2), Simple Storage Service (S3), SimpleDB and Simple Queueing Service (SQS) [4]. EC2 is a service that provides access to different types of Xen-virtualized machine images, a core infrastructure service on the Virtual Resource Set (VRS) layer of our reference model. Similar services are provided by a wide range of companies, including AppNexus Cloud [7], Bluelock Virtual Cloud Computing [10], ENKI Virtual Private Data Centers [15], FlexiScale Cloud Computing[19], GoGrid Cloud Hosting [23], Joyent Accelerators [34], Rackspace Mosso Cloud Servers [48], and Terremark Infinistructure [55]. With S3 and SimpleDB, Amazon provides two services that allow persistent data storage in the Cloud. We classify S3 and similar services, such as GoGrid Cloud Storage, Joyent BingoDisk, Nirvanix Storage Delivery Network, and Rackspace Mosso Cloud Storage as Basic Infrastructure Services (BIS) because they only provide basic storage functionality. Database-as-a-Service offerings like SimpleDB and technologies, such as Google Bigtable [17], 10gen MongoDB [1] and Hadoop HBase [31], on the other hand, are categorized as Higher Infrastructure Services (HIS) because they provide additional functionality, like a query language.

While it can be argued that Amazon's content delivery network service CloudFront [4] is a Higher Infrastructure Service (HIS), this category does not apply to more complex content and application delivery service offerings, such as Akamai's EdgePlatform. Java EE Web applications are deployed on EdgePlatform to run on Akamai's distributed computing infrastructure [2]. Therefore, Java design principles and existing program code can be used in order to

develop applications for EdgePlatform, since it is only the deployment model that changes. This is why EdgePlatform can best be categorized as a PaaS Execution Environment that draws on Java EE as Programming Environment. Similarly, Google App Engine is the PaaS Execution Environment for Python applications and Microsoft Azure a PaaS Execution Environment for applications built with Microsoft .NET technology.

3.1 Cloud Service Composition

Simple Cloud computing services can be combined into more complex and versatile service and application compositions. This is not only viable for services on the SaaS layer, as described in section 2.3, but also on lower layers of our stack model. Some engineers from Amazon illustrate how to design Web-scale application architectures by utilizing a combination of Amazon Web Services [56]. The example application GrepTheWeb [5] uses SQS queues in order to decouple controllers, the Hadoop MapReduce implementation on a cluster of EC2 instances, and S3 and SimpleDB for data storage and data retrieval.

The combination of Google's OpenSocial API and development platform with Google AppEngine is another example that demonstrates how Cloud computing technologies and services can be combined into a more powerful application architecture [38]. OpenSocial applications that do not draw on external server resources are subject to certain limitations, such as narrow data storage capacities or administrative constraints associated with the application container. Integrating an OpenSocial application with an App Engine backend allows to combine the client-side presentation layer with external data and application logic in a straightforward manner.

Both of the before mentioned examples show how services from the same Cloud computing service provider, i.e. Amazon and Google, respectively, can be combined in a reasonable fashion. However, there are good examples of how to combine services from different Cloud computing service providers, as well. Instead of extending OpenSocial applications with an App Engine backend it is also possible to develop Facebook applications that use external server capacities from Cloud computing service providers like Amazon or Joyent [16] [35]. Another prominent example is given by the extension of the Force.com developer platform with Google App Engine [50]. Just like in the examples before, the Force.com programming model and development environment is used to build business applications that draw on stored data and complex application logic residing on App Engine servers.

We have summarized our categorizations in Figure 2 (IaaS), Figure 3 (PaaS), Figure 4 (SaaS), and Figure 5 (HuaaS).

4. Outlook and Future Work

As our survey shows, the Cloud services landscape is very complex and, one could argue, therefore difficult to control centrally by any single party without severe scalability implications. We therefore envision self-organizing social support mechanisms such as federated markets to be developed for the Cloud where local and individual cost and benefit optimizations will evolve the offerings to be more efficient, force consolidation, nurture collaboration and drive serendipitous mash-ups.

To address the identified gap of business support systems that span Cloud layers we have also embarked on an implementation of a mash-up between a market system (Tycoon), a billing system (PayPal), a federated authentication system (OpenID) and a distributed computing platform (Hadoop) that we hope to present at future Cloud conferences.

Although many companies use Hadoop today in combination with other Cloud infrastructure such as EC2, there is no generally available pay-as-you-go zero-install Hadoop service. One reason for this is that Hadoop was designed for in-house intra data center operations with a trusted well-known user base. Efficient sharing of large volumes of data and computational resources in an untrusted multi-tenancy Cloud remains a challenging research topic. We have obtained some initial promising results when integrating Hadoop and Tycoon [51], and the next step is to close the end-to-end Cloud service loop by offering a custom version of the Hadoop scheduler with PayPal and OpenId account management. Such an integrated platform would allow developers to more easily replicate and automate use cases such as the New York Times experiment mentioned in Section 1.2.

Another area of future work is to elaborate on a number of concrete use case scenarios of end-to-end Cloud services to refine and further motivate the use of our stack.

References

[1] 10gen. Scalable high performance data storage for web applications. http://www.10gen.com, Seen: 2009-01-25, 2009.

[2] Akamai Technologies Inc. Akamai edgecomputing. enabling applications that grow your business. http://www.akamai.com/dl/whitepapers/Akamai_Enabling_Apps_Grow_Business_Whitepaper.pdf, Seen: 2009-01-25, 2009.

[3] Amazon. Mechanical turk. https://www.mturk.com/mturk/welcome, Seen: 2009-02-18, 2009.

[4] Amazon Web Services. Amazon webservices homepage. http://aws.amazon.com, Seen: 2008-12-05, 2008.

[5] Amazon Web Services. Building grepptheweb in the cloud, part 1: Cloud architectures. http://developer.amazonwebservices.com/

connect/entry.jspa?externalID=1632, Seen: 2009-01-25, 2008.

[6] S. Andreozzi, N. D. Bortoli, S. Fantinel, A. Ghiselli, G. Rubini, G. Tortone, and M. Vistoli. GridICE: a Monitoring Service for Grid Systems. *Future Generation Computer Systems Journal*, 21(4):229–571, 2005.

[7] AppNexus. Appnexus cloud. http://www.appnexus.com, Seen: 2009-01-25, 2009.

[8] D. Baran. Cloud computing basics. http://www.webguild.org/2008/07/cloud-computing-basics.php, Seen: 2008-11-07, July 2008.

[9] J. Berg, R. Forsythe, F. Nelson, and T. Rietz. Results from a dozen years of election futures markets research. *Handbook of Experimental Economic Results*, 2001.

[10] Bluelock. Bluelock. http://www.bluelock.com/, Seen: 2009-01-25, 2009.

[11] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Symposium on Operating System Design and Implementation*, 2004.

[12] Digg. What is digg? http://www.digg.com/about, Seen: 2009-02-18, 2009.

[13] Django. Django web framework. http://www.djangoproject.com, Seen: 2009-01-23, 2009.

[14] B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer. Xen and the Art of Virtualization. In *Proceedings of the ACM Symposium on Operating Systems Principles*, 2003.

[15] ENKI. Virtual private datacenters. http://www.enkiconsulting.net/virtual-private-data-centers/, Seen: 2009-01-25, 2009.

[16] N. Everett. Hosting facebook applications on amazon ec2. http://developer.amazonwebservices.com/connect/entry.jspa?\entryID=1044, Seen: 2009-01-23, 11 2007.

[17] Fay Chang and Jeffrey Dean and Sanjay Ghemawat and Wilson C. Hsieh and Deborah A. Wallach and Mike Burrows and Tushar Chandra and Andrew Fikes and Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *Symposium on Operating System Design and Implementation*, 2006.

[18] C. Ferris, A. Karmarkar, and P. Yendluri. Basic Profile Version 2.0. Technical report, Web Services Interoperability Organization, 2007.

[19] FlexiScale. Flexiscale cloud computing. http://www.flexiscale.com, Seen: 2009-01-25, 2007.

[20] I. Foster, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, H. Kishimoto, F. Maciel, A. Savva, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. V. Reich. The Open Grid Services Architecture Version 1.0. Technical report, Global Grid Forum, 2004.

[21] GENI. Global environment for network innovations. http://www.geni.net. Seen: 2009-01-23, 2009.

[22] Giuseppa DeCandia and Deniz Hastorun and Madan Jampani and Gunavardhan Kakulapati and Avinash Lakshman and Alex Pilchin and Swaminathan Sivasubramanian and Peter Vosshall and Werner Vogels. Dynamo: Amazon's Highly Available Key-Value Store. In *ACM Symposium on Operating Systems Principles*, 2007.

[23] GoGrid. Gogrid cloud hosting. http://www.gogrid.com, Seen: 2009-01-25, 2009.

[24] Google Inc. Google apps engine. http://www.google.com/apps, Seen: 2008-12-05, 2008.

[25] Google Inc. Google docs. http://docs.google.com/, Seen: 2009-01-24, 2009.

[26] Google Inc. Google maps api. http://code.google.com/apis/maps, Seen: 2009-01-23, 2009.

[27] D. Gottfrid. Self-service, Prorated Super Computing Fun! *The New York Times*, (1), November 2007.

[28] O. M. Group. The Common Object Request Broker: Architecture and Specification, 1998.

[29] T. O. Group. X/Open DCE: Remote Procedure Call, November 1995.

[30] Hadoop. Hadoop homepage. http://hadoop.apache.org. Seen: 2008-11-07, 2008.

[31] H. HBase. Hbase homepage. http://hadoop.apache.org/hbase/, Seen: 2008-11-21, 2008.

[32] Hewlett-Packard. HP Integrated Lights-Out 2 User Guide. Technical report, HP, 2009.

[33] M. Hibler, R. Ricci, L. Stoller, J. Duerig, S. Guruprasad, T. Stack, K. Webb, and J. Lepreau. Large-scale Virtualization in the Emulab Network Testbed. In *Proceedings of the 2008 USENIX Annual Technical Conference*, 2008.

[34] Joyent. Reasonably smart. http://www.joyent.com, Seen: 2009-01-26, 2009.

[35] Joyent Inc. Joyent accelerators for facebook developers. http://www.joyent.com/developers/facebook, Seen: 2009-01-23, 01 2009.

[36] K. Keahey and T. Freeman. Contextualization: Providing one-click virtual clusters. In *eScience 2008*, 2008.

[37] K. Lai, L. Rasmusson, E. Adar, S. Sorkin, L. Zhang, and B. A. Huberman. Tycoon: an Implemention of a Distributed Market-Based Resource Allocation System. *Multiagent and Grid Systems*, 1(3):169–182, Aug. 2005.

[38] L. LiaBraaten. Building and opensocial app with google app engine. http://code.google.com/intl/de-DE/apis/opensocial/articles/appengine-0.8.html, Seen: 2009-01-23, 09 2008.

[39] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in college networks, 2008.

[40] Microsoft. Azure services platform. http://www.microsoft.com/azure, Seen: 2009-01-23, 2009.

[41] Microsoft. Office live. http://www.officelive.com, Seen: 2009-01-23, 2009.

[42] S. Microsystems. Project caroline. http://research.sun.com/projects/caroline, Seen: 2009-01-23, 2009.

[43] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. Eucalyptus open-source cloud-computing system. In *CCA08: Cloud Computing and Its Applications*, 2008.

[44] OCC. The open cloud consortium. http://www.opencloudconsortium.org, Seen: 2009-01-23, 2009.

[45] Open Cirrus. Cloud computing testbed. http://www.opencirrus.org, Seen: 2009-01-23, 2009.

[46] OpenId Foundation. Openid homepage. http://www.openid.net, Seen: 2008-12-05, 2008.

[47] K. Park and V. S. Pai. CoMon: A Mostly-Scalable Monitoring System for PlanetLab. *Operating Systems Review*, 40(1), 2006.

[48] Rackspace. Rackspace managed hosting. `http://www.rackspace.com`, Seen: 2009-01-25, 2009.

[49] K. Rangan, A. Cooke, M. Dhruv, J. Post, N. Schindler, J. Fidacaro, W. Mohan, G. A. B. III, and J. Vleeschhouwer. The cloud wars: $100+ billion at stake. Technical report, Merrill Lynch, 2008.

[50] Salesforce.com Inc. Force.com for google app engine: Connecting the clouds. `http://developer.force.com/appengine`, Seen: 2009-01-23, 2008.

[51] T. Sandholm and K. Lai. MapReduce Optimization using Regulated Dynamic Prioritization. In *ACM SIGMETRICS'09: International Conference on Measurement and Modeling of Computer Systems*, 2009. to appear.

[52] Sanjay Ghemawat and Howard Gobioff and Shun-Tak Leung. The Google File System. In *ACM Symposium on Operating Systems Principles*, 2003.

[53] B. Sotomayor, R. Montero, I. M. Llorente, and I. Foster. Capacity Leasing in Cloud Systems using the OpenNebula Engine. In *CCA08: Cloud Computing and its Applications*, 2008.

[54] A. I. Technologies. Hangout. `http://fun.auciti.com`, Seen: 2009-01-23, 2009.

[55] Terremark. Infinistructure. `http://www.terremark.com/services/managed-hosting.aspx`, Seen: 2009-01-25, 2009.

[56] J. Varia. Cloud architectures. Technical report, Amazon Webservices, 2008.

[57] XMind. Cloud xmind social brainstorming and mind mapping. `http://share.xmind.net/zhenjl/cloud`, Seen: 2009-01-25, 2009.

[58] YouTube. Company history. `http://www.youtube.com/t/about`, Seen: 2009-02-18, 2009.

Organization	Service or tool	Description	Layer
Amazon	Elastic Compute Cloud (EC2)	Virtual servers	IaaS > RS > VRS
	Dynamo	Key-value storage system	IaaS > RS > HIS
	Simple Storage Service (S3)	Storage buckets	IaaS > IS > BIS
	SimpleDB	Database-as-a-Service	IaaS > IS > HIS
	CloudFront	Content Delivery	IaaS > IS > HIS
	SQS	Queueing services	IaaS > IS > HIS
AppNexus	AppNexus Cloud	Virtual servers	IaaS > RS > VRS
Bluelock	Bluelock Virtual Cloud Computing	Virtual servers	IaaS > RS > VRS
	Bluelock Virtual Recovery	Disaster Recovery	IaaS > IS > HIS
Emulab	Emulab Network Testbed	Network testbed	IaaS > RS > PRS
ENKI	ENKI Virtual Private Data Centers	On-demand virtual data center resources	IaaS > RS > VRS
EU Reservoir project	Open Nebula	Open source virtual infrastructure engine	IaaS > RS > VRS
FlexiScale	FlexiScale Cloud Computing	Virtual servers	IaaS > RS > VRS
GoGrid	Cloud Hosting	Virtual servers	IaaS > RS > VRS
	Cloud Storage	Disk storage	IaaS > IS > BIS
Google	Google Big Table	Distributed storage system	IaaS > IS > HIS
	Google File System	Distributed file system	IaaS > IS > BIS
HP	iLO	Lights out management	IaaS > RS > PRS
	Tycoon	Market-based system for managing compute resources in clusters	IaaS > RS > VRS
Joyent	Accelerator	Virtual servers	IaaS > RS > VRS
	Connector	Pre-configured virtual servers	IaaS > IS > HIS
	BingoDisk	Disk storage	IaaS > IS > BIS
Nirvanix	Nirvanix Storage Delivery Network	Disk storage	IaaS > IS > BIS
OpenFlow	OpenFlow	Network simulation	IaaS > IS > BIS
Rackspace	Mosso Cloud Sites	Pre-configured virtual servers	IaaS > IS
	Mosso Cloud Storage	Disk storage	IaaS > IS > BIS
	Mosso Cloud Servers	Virtual servers	IaaS > RS > VRS
Skytap	Skytap Virtual Lab	Virtual IT lab environment	IaaS > IS
Terremark	Infinistructure	Virtual servers	IaaS > RS > VRS
The Globus Alliance	Nimbus	Open source toolkit to turn a cluster into an IaaS cloud.	IaaS > RS > VRS
UCSB	EUCALYPTUS	Open source implementation of Amazons EC2	IaaS > RS > VRS
10gen	Mongo DB	Database for cloud storage	IaaS > IS > HIS
	Babble Application Server	Web application server for cloud deployments	IaaS > IS > HIS

Figure 2: Infrastructure-as-a-Service providers.

Organization	Service or tool	Description	Layer
Akamai	EdgePlatform	Content, Site, Application Delivery	PaaS Exec. Env.
Facebook	Facebook Platform	Development tools and execution environment for social networking applications	PaaS
Google	App Engine	Scalable runtime environment for Python Web applications	PaaS
Microsoft	Azure	Development environment and runtime for Microsoft applications	PaaS
	Live Mesh	Platform to sync, share and access a wide range of devices with Microsoft operating systems	PaaS
NetSuite	SuiteFlex	Toolkit to customize NetSuite online business applications	PaaS
Salesforce	Force.com	Build and deliver on-demand business applications	PaaS
Sun	Caroline	Horizontally scalable platform for the development and deployment of Internet services.	PaaS
Zoho	Zoho Creator	Toolkit to build and deliver on-demand business applications	PaaS

Figure 3: Platform-as-a-Service providers.

Organization	Service or tool	Description	Layer
Google	Google Docs	Online office suite	SaaS
	Google Maps API	The Google Maps API lets developers embed Google Maps in their own web pages with JavaScript.	SaaS > BAS
	OpenSocial	A common API for social applications across multiple websites.	SaaS > CAS
OpenID Foundation	OpenID	Distributed system to allow users to have a single digital identity across the Internet.	SaaS > BAS
Microsoft	Office Live	Online office suite	SaaS
Salesforce	Salesforce.com	Customer Relationship Management	SaaS

Figure 4: Software-as-a-Service providers.

Organization	Service or tool	Description	Layer
Amazon	Mechanical Turk	Scalable workforce	HuaaS > CS
Digg.com	Digg.com	News aggregation	HuaaS > CS > IAS
The University of Iowa	Iowa Electronic Markets	Future markets based on economic and political events	HuaaS > CS > IAS
Youtube	Youtube	Video portal	HuaaS > CS

Figure 5: Human-as-a-Service providers.