Cloud Computing – Lecture 3

Platform as a Service (PaaS)

Pelle Jakovits
Jakovits@ut.ee
Outline

• Introduction to Platform as a Service (PaaS) Cloud model
• Different types of PaaS
• Google App Engine
• Heroku PaaS
• Advantages & disadvantages of PaaS
Cloud Models

SaaS

PaaS

IaaS

http://nolegendhere.blogspot.com.ee/2012/06/presentation-4-5-7.html
Background

• Previous lecture introduced the different cloud computing models (IaaS, PaaS, SaaS)

• IaaS provides computing resources
  – Virtual machines, storage, network.

• User do not need to purchase hardware themselves

• IaaS can utilize resources more efficiently in comparison to each customer using own hardware

• You have worked with OpenStack instances in the lab
Issues with using IaaS

• To deploy applications in IaaS, need to choose and set up:
  – Computing infrastructure
  – Software environment

• User is responsible for:
  – System administration, backups
  – Monitoring, log analysis
  – Managing software updates
  – Stability & scalability of the software environment
Cloud Model complexity

- **Own Hardware**: Applications, Data, Runtime, Middleware, Operating System, Virtualization, Servers, Storage, Networking
- **IaaS**: Applications, Data, Runtime, Middleware, Operating System, Virtualization, Servers, Storage, Networking
- **PaaS**: Applications, Data, Runtime, Middleware, Operating System, Virtualization, Servers, Storage, Networking
- **SaaS**: Applications, Data, Runtime, Middleware, Operating System, Virtualization, Servers, Storage, Networking
Platform as a Service - PaaS

- Complete platform for hosting applications in Cloud
- The underlying infrastructure & software environment is managed for you
- Enables businesses to build and run web-based, custom applications in an on-demand fashion
- Eliminates the complexity of selecting, purchasing, configuring, and managing hardware and software
- Dramatically decreases upfront costs
PaaS Characteristics

• Multi-tenant architecture
• Built-in scalability of deployed software
• Integrated with cloud services and databases
• Simplifies prototyping and deploying startup solutions
• More fine-grained cost model
  – Generally do not pay for unused resources
  – Users only pay for services they use
• Typically introduces vendor lock-in
Different types of PaaS

1. **Fully managed cloud platforms** for web applications
   - Google App Engine, AWS BeanStalk, Windows Azure

2. **Data Processing as a Service**
   - AWS Elastic MapReduce (EMR)

3. **Open-platform PaaS**
   - Cloud Foundry, OpenShift
   - **DotCloud PaaS** (bankrupt) -> but the underlying tech led to Docker

4. **Function as a Service** (FaaS)
   - IBM Functions, Amazon Lambda, Google Cloud Functions
PaaS for web applications

• Typically built on-top of existing IaaS Cloud
• Provides and **manages** all computing resources and services needed for running web applications
• **Google App Engine**, AWS BeanStalk
• Open-Computing Platforms, which are not tied to a single IaaS provider
  – Interoperability and open-source tools
  – E.g. Cloud Foundry, Red Hat OpenShift etc.
• Social App Platforms
  – Develop add-ons for SaaS, such as Google+ or Facebook
  – Integrated API with the SaaS platform
Google App Engine
Google App Engine

- PaaS for developing and hosting web applications in Google-managed data centers
- Easy to build, maintain, and scale applications
- No servers to maintain or configure by yourself
- Upload & Go
- Was created before Google Cloud became public
- Supported languages
  - Python, Java, PHP, Go
Sandboxing

• App Engine distributes requests for applications across multiple web servers
• To prevent one application from interfering with another, the application runs in a restricted "sandbox" environment.
• App Engine application cannot:
  – write to the filesystem. Applications must use Cloud services for storing persistent data. Reading from the filesystem is allowed.
  – respond slowly. Request must be handled within 30 seconds. Otherwise process is terminated.
  – make other kinds of system calls.
App Engine Characteristics

• **Persistent storage** with queries, sorting, and transactions
• App Engine distributes user requests across multiple servers and **scales servers** to meet **dynamic traffic** demands
• **Asynchronous task queues** for performing work outside the scope of a request
• **Scheduled tasks** for triggering events at specified times or regular intervals
• **Integration** with all other Google Cloud services and APIs
• Application runs within its own **secure, sandboxed and reliable** environment
  – Independent of hardware, OS or physical location of the server
Available cloud services

- **Google Cloud SQL** - A fully-managed web service that allows you to create, configure, and use relational databases in Google's cloud.

- **Datastore** - A schemaless object datastore providing robust, scalable storage for your web application, a rich data modeling API, and a SQL-like query language called GQL.

- **Blobstore** - Allows your application to serve large data objects, such as video or image files, that are too large for storage in the Datastore service.

- **Users** - Allows applications to sign in users with Google Accounts or OpenID, and address these users with unique identifiers.
Available services

- **Search** - Allows your application to perform Google-like searches over structured data: plain text, HTML and geographic locations.

- **Memcache** - A distributed, in-memory data cache to improve application performance.

- **Logs** - Provides programmatic access to application and request logs from within your application.

- **Remote** - Lets external applications transparently access App Engine services. For example, to access a production datastore from an app running on local machine.
App Engine Environment
Integration with Cloud services
Deployment Life Cycle

Write Code → Test Locally

Admin via Web Console ↔ Push to Google servers
Free Google Cloud Account

• **NB!** Not required to try Google App Engine
  – However, this may be disabled starting from April 2020
• $300 Credit for 12 months
• Maximum limit of 8 cores at a time for instances
• **Need credit card to sign up**
  – Won't be billed until upgrade to paid account
  – Only the portion of usage above the App Engine free daily quota is charged against credit
## App Engine Daily Free Quotas

<table>
<thead>
<tr>
<th>Requests</th>
<th>Daily quota</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Outgoing Bandwidth</strong></td>
<td>1 GB</td>
</tr>
<tr>
<td><strong>Incoming Bandwidth</strong></td>
<td>unlimited</td>
</tr>
<tr>
<td><strong>Frontend Instance Hours</strong></td>
<td>28 Instance Hours</td>
</tr>
<tr>
<td><strong>Backend Instance Hours</strong></td>
<td>9 Instance Hours</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Storage</th>
<th>Daily quota</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cloud Datastore Read Operations</strong></td>
<td>50K Ops</td>
</tr>
<tr>
<td><strong>Cloud Datastore Small Operations</strong> (allocate IDs or keys-only queries)</td>
<td>Unlimited</td>
</tr>
<tr>
<td><strong>Cloud Datastore Stored Data</strong></td>
<td>5 GB</td>
</tr>
<tr>
<td><strong>Blobstore Stored Data</strong></td>
<td>5 GB</td>
</tr>
<tr>
<td><strong>Number of Indexes</strong></td>
<td>200</td>
</tr>
</tbody>
</table>
Heroku PaaS

• Was one of the first PaaS services offered on the market
• Provides fully managed reliable containers for applications
• Supports Node, Ruby, Java, PHP, Python, Go, Scala or Clojure
• Heroku is based on containers – **Heroku Dynos**
  – Containers are based on Linux kernels and Ubuntu images
Heroku Dynos

Queue add-on

Database add-on

Web dynos

Worker dynos

An Application
Heroku containers - Dynos

• Isolated and virtualized Linux containers
• Sleeps after 30 mins of inactivity
• Monthly free dyno hours
  – Unverified accounts: 550 free dyno hours
  – Verified accounts: additional 450 free dyno hours
• 512 MB RAM per dyno
• Up to 5 free apps (unverified) or 100 (verified)

• Web Dyno – serves web requests
• Worker Dyno – processes tasks
Scaling Dynos

- Like changing num replicas in Docker Swarm service
- Developers can use Manual scaling
- Configure Autoscaling
  - e.g. based on **Desired p95 Response Time**

Source: https://devcenter.heroku.com/articles/scaling#dyno-formation
Heroku Marketplace - plugins

- Additional managed services provided by Heroku or other providers
  - Many different plugins
  - Subscription plans are usually per month.
- Heroku Postgres - Managed database service
  - Free quota: Row Limit 10,000, Storage Capacity 1 GB
- ElasticSearch - RESTful search engine
- Bucketeer: Access S3 from Heroku
- Kafka: CloudKarafka (5 topics for free)
- Many other types: Monitoring, Logging, Email/SMS, Caching, Content Management, Search Metrics and Analytics, Testing, Messaging and Queueing, Alerts and Notifications, User Management, Development Tools, Document, Image, Video Processing, CI/CD
Heroku cost model

- Dynos can be significantly more expensive in comparison to running containers in IaaS
- Dyno: $25 per dyno per month (500MB RAM)
- AWS: $4.2 ($0.0058 hour) per t2.nano instance per month (500 MB RAM)
- Cost of a fully managed services
- Less administration and operations cost
- Other services often bill by fixed quota subscription
  - Heroku plugins: DB, messaging
  - Managed Postgre: $50 month for 4GB RAM, 64GB storage, 120 connections
AWS Elastic Beanstalk

• Platform as a Service for web applications
• Languages: Java, .NET, PHP, Node.js, Python, Ruby, Go
• Platforms: Docker, Apache, Nginx, Passenger, and IIS
• Upload code & deploy
• Automatically handles deployment, capacity provisioning, load balancing, auto-scaling, application health monitoring
• More manual control available (and required)

https://aws.amazon.com/elasticbeanstalk/
Function as a Service (FaaS)

- Often also referred to as Serverless
- Each deployed "application" is a single Function
- Functions are independent from each other
  - Scaled, managed and billed separately
  - Can be written in different languages

- **Event driven execution** based on triggers and preconditions:
  - **Trigger Event:** New image uploaded to S3
  - **Precondition:** File size is larger than 10 MB
  - **Execute:** Resize_image(filePath)

- Functions can be composed into larger applications
Serverless/FaaS examples

• **AWS Lambda**
  - Run code/functions in AWS without managing infrastructure or software environment
  - Pricing is based on number of **requests** and **GB-Sec "Memory-Duration"**
  - Free: 1M **requests** a month. After: $0.20 per 1M
  - Free: 400,000 **GB-Sec**. After: $0.000017 per 1 **GB-SsC**

• **Apache OpenWhisk**
  - Open source serverless cloud platform
  - Used also by IBM Bluemix FaaS
  - Event, trigger & rule based execution
  - Supports any language*
    - JavaScript, Swift, Python or PHP function, Java or any binary executable.
Data Processing as a Service

• PaaS for data processing applications
• Service provider manages the data processing cluster
• User specifies required computing resources and uploads the application
• Typically consist of a Hadoop cluster and includes a selection of Hadoop Ecosystem frameworks and tools

• AWS Elastic MapReduce
• Google Cloud DataProc
Amazon Elastic MapReduce

• Service for requesting Amazon-customized Hadoop clusters on-demand
• Provides web interface and command-line tools for running Hadoop jobs on EC2 cloud
• Supports MapReduce, Spark, Hive, Pig, Flink
• Data can be stored in HDFS, Amazon S3, Dynamo DB, Redshift
• Native support for spot instances

http://aws.amazon.com/elasticmapreduce/
Amazon EMR Jobs

• EMR Provides a managed and elastic platform for running Hadoop data processing tasks

• User must:
  – Upload MR job jar & input data to S3
  – Define the size of the EMR cluster to be launched
  – Create an EMR Job Flow as a sequence of data processing steps
  – Wait for the completion
  – Examine the results

• EMR monitors job and shuts machines after completion
IaaS vs PaaS Pricing Model

• IaaS
  – Per instance hour
  – Per storage volume/month
  – Per network bandwidth/month
  – Per additional services (Static IP, Autoscaling)

• PaaS
  – More fine-grained cost model
  – Individual prices for each Cloud Service
  – Small applications can be hosted for free
PaaS Advantages

- Do not have to manage low level resources and services
- Many services ready to use in a plug-in fashion without any configuration or setup
- Provider handles most of the non-functional requirements of your applications
- Scaling is automatically managed by the platform
- Easier and more agile application deployment
  - Simplifies prototyping and launching software startup apps/services
- More fine-grained cost model than in IaaS
  - Pay only for resource which are used
- Platform provider has the best knowledge to optimize the services running on the underlying hardware
Disadvantages of PaaS

- **Not in full control** over:
  - Computing resources (Intel vs AMD, GPU’s, FPGA, ...)
  - Software and library versions
  - Service configuration
- Available programming languages are sometimes limited
- **Vendor lock-in**
- Offered services may not be flexible enough for user needs
- Have to **fully trust** in the PaaS provider
  - Billing accuracy, Security, Reliability, Data ownership
- More complex cost estimation
  - What happens when application exceeds billing quotas?
  - What happens when payments fail?
That's All

• This weeks practice session is:
  – Heroku PaaS: Creating and deploying applications

• Next lecture
  – Function as a Service (Saas) - Serverless
References


• Google Cloud Platform https://cloud.google.com/

• Chakkrit Tantithamthavorn, Introduction to Google App Engine, https://www.slideshare.net/klainfo/introduction-to-google-app-engine-13223789