Open-source platform-as-a-service requirements and implementation challenges

Dana Petcu

West University of Timisoara
& Institute e-Austria Timisoara
Romania

web.info.uvt.ro/~petcu
Overview

1. The problem: vendor-lock-in in Clouds
2. The solution? Interoperability and portability
3. The role of open-source platforms
4. mOSAIC case study
5. Research challenges
6. Further steps
Vendor lock-in, due to proprietary APIs

How to protect my application from lock-in?
Interoperability and portability: challenges

Scenarios using multiple Clouds

Multi-dimensional problem

**POLICY:**
Federate, communicate between providers

**RUNTIME:**
Migration support

**DESIGN:**
Abstract the programmatic differences
Interoperability and portability: approaches

Levels

- Business
- Semantic
- Appl & service
- Management
- Techs & infrastr
- Image & data
- Network

E.g.
- Strategies, regulations, mode of use
- Function calls and responses
- Automation, configuration
- Standards in deployment & migration
- Protocols for requests/responses
- Pre-deployment, work-loads
- Allocation, admission

Techs

- Domain specific lang.
- Semantic repositories
- Abstraction layers
- Standards
- Open protocols
- Open APIs
- Open

E.g.
- Automated translation in code
- UCI
- Mediators, frameworks (SLA@SOI)
- OVF/DMTF, CDMI/SNIA
- OCCI, Deltacloud
- jClouds, libcloud, OpenStack
PaaS – the worst case!

- each PaaS provider offers a special flavor in its design
- not all the features that are expected
  - rarely debug facilities,
  - no Security-as-a-Service
  - often not for private clouds
- portability is possible only between a small no.PaaSs
  - in case of open-source clones of the proprietary ones
- the problem escalates with the increase of no.PaaSs
  - increased no. in last two years
Deployment of Cloud techs at PaaS level: two types

Platform Service | Hosting | Integrated solution

- Well knows examples:
  - Google’s AppEngine, Microsoft’s Azure, RackSpace’s CloudSites, Amazon’s Beanstalk, Salesforce’s Force.com, Joyent’ SmartPlatform

- Open-source only to develop apps to allow customization

- Appl support – different approaches:
  - Deploy code to specific VMs (Azure, Beanstalk)
  - Develop using rules, platform deal with deployment (AppEngine, Heroku)
  - Create metadata to be interpreted by PaaS at run-time (Force.com, OrangeScapes)

Platform Software | Software service | Deploy-based solution

- Deployment of middleware in data centers
- Easy way to deal with portability and interoperability (framework category)
- Open-source have the potential to impact the market as...
  - PVM/Parallel
  - Globus/Grid
## Open-source Platform Software

<table>
<thead>
<tr>
<th>Product</th>
<th>AppScale</th>
<th>Cloud Foundry</th>
<th>ConPaaS</th>
<th>mOSAIC</th>
<th>OpenShift</th>
<th>TyphoonAE</th>
<th>WaveMaker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>Univ. California</td>
<td>VMWare</td>
<td>Contrail Consortia</td>
<td>mOSAIC Consortia</td>
<td>RedHat</td>
<td>Tobias Rodäbel</td>
<td>VMWare</td>
</tr>
<tr>
<td>State</td>
<td>1.5/Jul 2011</td>
<td>0.x , Beta</td>
<td>0.1/Sep 2011</td>
<td>0.5/Jun’12, Beta</td>
<td>Production</td>
<td>0.2/Dec 2010/beta</td>
<td>6.4.4/Dec 2011</td>
</tr>
<tr>
<td>Languages</td>
<td>Python, Java, Go</td>
<td>Java, Ruby,Node.js, Go</td>
<td>PHP</td>
<td>Java, Python</td>
<td>Java, Python, Perl, PHP, Ruby</td>
<td>Python</td>
<td>Java</td>
</tr>
<tr>
<td>Data Support</td>
<td>HBase, Redis, Hypertable, MySQL Cluster, Cassandra, Voldemort, MongoDB, Memcached, DB</td>
<td>MongoDB, SQLFire, PotsgreSQL, Redis</td>
<td>Scalaris, MySQL, XtreemFS</td>
<td>Riak, MemcachedDB, Redis, MySQL</td>
<td>MySQL, MongoDB, Amazon RDS</td>
<td>MongoDB, MySQL, Berkeley DB J</td>
<td>Amazon S3, Rackspace</td>
</tr>
<tr>
<td>OS</td>
<td>Ubuntu, CentOS on Xen, KVM</td>
<td>VMWare image</td>
<td>XtreemOS image</td>
<td>mOS image, Linux</td>
<td>Red Hat Virtualization</td>
<td>Debian, Ubuntu</td>
<td>VMWare image</td>
</tr>
<tr>
<td>Messaging</td>
<td>Channel</td>
<td>RabbitMQ</td>
<td>Own design</td>
<td>RabbitMQ</td>
<td>Own design</td>
<td>RabbitMQ, ejabberd, Channel</td>
<td>Own design</td>
</tr>
<tr>
<td>Clouds tested</td>
<td>Amazon EC2, Eucalyptus</td>
<td>VMWare</td>
<td>Own testbed</td>
<td>Amazon EC2, Eucalyptus, OpenNebula, Flexiscale</td>
<td>RightScale Rackspace, Smart-Cloud, Amazon</td>
<td>Google</td>
<td>EC2, Rackspace, OpSource, Eucalyptus</td>
</tr>
<tr>
<td>Interface</td>
<td>CLI, Web</td>
<td>CLI</td>
<td>Web</td>
<td>CLI,Web, REST</td>
<td>CLI,REST</td>
<td>CLI</td>
<td>Studio</td>
</tr>
</tbody>
</table>
# Open-source Platform Software

<table>
<thead>
<tr>
<th>Product</th>
<th>CloudFoundry</th>
<th>mOSAIC</th>
<th>OpenShift</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development support</strong></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Dedicated to web apps or general</td>
<td>Web apps</td>
<td>General</td>
<td>Web apps</td>
</tr>
<tr>
<td>Desktop Cloud Simulator</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>API access</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Support standard programming libs</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Impact on web application architecture</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Complexity of porting web application</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Standard support tools</td>
<td>Spring Tools</td>
<td>No</td>
<td>JBoss, Zend</td>
</tr>
<tr>
<td>Thread access</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MySQL</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Allows to choose stack components</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Allow to pull data out</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Debugging mode</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Deployment support</strong></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Lock-in when building own Cloud</td>
<td>Yes (VMWare)</td>
<td>No</td>
<td>Yes (RHE)</td>
</tr>
<tr>
<td>Web server (e.g. Tomcat)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Build-in-balancer</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Auto-scaling app server</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Auto-scaling database</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Performance analytics</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Support multiple Cloud providers</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Agreements SLA</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Deploy with a special tool</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Support Private Cloud</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Allows to add third party components</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Execution support</strong></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Command line (CLI)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Web console</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Access to logs via web</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Web based monitoring</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Multitenant</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
mOSAIC’s marketing motto: “Flying through the Clouds”
**mOSAIC as R&D collaboration effort**

- [www.mosaic-cloud.eu](http://www.mosaic-cloud.eu)

**Consortium:**
1. Second University of Naples, Italy
2. Institute e-Austria Timisoara, Romania
3. European Space Agency, France
4. Terradue SRL, Italy
5. AITIA International Informatics, Hungary
6. Tecnalia, Spain
7. Xlab, Slovenia
8. University of Ljubljana, Slovenia
9. Brno University of Technology, Czech Republ.

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>September 2011</td>
<td>1st API implement. (Java)</td>
<td></td>
</tr>
<tr>
<td>September 2012</td>
<td>1st stable PaaS, 2nd API impl. (Python)</td>
<td></td>
</tr>
<tr>
<td>March 2013</td>
<td>Full software package</td>
<td></td>
</tr>
</tbody>
</table>
# Layered architecture

## Open-source and deployable PaaS

### Cloud adaptors
- Hosting services support
  - Amazon
  - Flexiscale
  - Arctur
  - CloudBurst
  - GoGrid
  - Hostko
  - Rackspace
  - CloudSigma
  - CHS
- Deployable services support
  - Eucalyptus
  - OpenNebula
  - DeltaCloud
  - OpenStack
  - HDFS
- Other Cloud hosting, deployable services

### mOSAIC PaaS and IaaS
- **Application support**
  - API implementations
    - Java cloudlets
    - Python cloudlets
    - Java connectors
    - Python connectors
    - Demo applications
  - Application tools
    - Eclipse plug-ins
    - Frontends (cmdl, web)
    - Network backends
    - Configuration tools
    - PortableTestbedCluster
  - Service discoverer
  - Semantic engine
    - Semantic query
    - Reasoner
    - Maintainer
    - Search engine
    - Ontology
  - Application service components
    - SLA
    - Network
    - Benchmark
    - Deployment
    - Interoperability
    - mOS
  - Infrastructure support
    - Cloud Agency
      - MTP
    - Mediator
    - Meter
    - Archiver
    - Tier agents
    - Vendor agents
  - OS repository: https://bitbucket.org/mosaic

### Software platform support
- Platform’s core components
  - Register & Discover
  - Packager & Deployer
  - Provisioner & Maintain
  - Scheduler & Scaler
  - Interoperability
  - mOS
- Application service components
  - SLA
  - Network
  - Benchmark
- Application support components
  - Deployable COTS
  - Drivers

### Infrastructure support
- Cloud adaptors
  - mOSAIC’s proof-of-the-concept applications
    - Earth Observation application
    - Intelligent maintenance system
    - Information extraction
    - Model exploration service
    - Analysis of structures
    - User community developed applications
Application lifecycle

1) Cloudlets + mOS software modules
2) Application descriptor
3) Deployable Apps
4) Deployment descriptor
5) Monitoring
How to use it? One scenario:

- **Write component-based application**
  - Languages: Java or Python
  - Communications through message passing
  - Respect the event-driven style of programming
  - Find the proper functionalities with the Semantic Engine

- **Debug your application on the desktop or on-premise server(s)**
  - Within Eclipse
  - Use Personal Testbed Cluster using VirtualBox for the VMs

- **Deploy your application in a Cloud**
  - Assisted by Cloud Agency and Broker (with SLAs)

- **Monitor & modify the applications**
  - Control the life-cycle of the components (start/stop/replace)

**Need help?**

*Follow documentation and YouTube demos (search “mOSAIC Cloud computing”)*
Research issues related to mOSAIC’s PaaS

❖ Auto-scaler

❖ Scheduler

❖ Naming service
  ▪ S. Panica, Distributed Resource Identification Service for Cloud Environments, submitted to WISE 2012 in June
Auto-scaler (PhD stud. B. Caprarescu)

- **Problem:** Most existing auto-scaling solutions are centralized

- **Solution:**
  - based on a P2P architecture
  - one autonomic service is deployed on each VM

- **DEPAS algorithm:**
  - each VM probabilistically decides to add new nodes or remove itself.
  - probability is computed based on an estimation of the average system load
  - the average system load is approximated by each node with the average load of itself and its neighbors
Auto-scaler (PhD stud. B. Caprarescu)

- **Simulation results** in ADAPTIVE procs
- **On Amazon EC2** in COMPUTING j.
- **Test conclusions:**
  - After a period of adaption, DEPAS allocates a right capacity (between the optimum capacity at max load and optimum capacity at min load)
  - The delay in adaptation is cased by the relatively high duration of load monitoring timeframe and cycle duration
  - The benefit comes in the high stability (no oscillations) which is impressing for a decentralized algorithm
Scheduler (PostDoc Marc Frincu)

- **Problem:**
  - component-based apps can encounter failures of components
  - a scaled application can span its components on several nodes
  - finding the optimal no. component types needed on nodes so that every type is present on every allocated node
  - cost restrictions and threshold for no. nodes

- **Application to:**
  - Highly available Web 2.0 applications

- **Novelty:**
  - Most of the approaches schedule VMs not components

- **Scheduling algorithms:**
  - One that produces an optimal solution in case when the load of every component is known
  - One that produces a sub-optimal solution and relies on a GA to allocate components in case the component load is unknown
Scheduler (PostDoc Marc Frincu)

- **Test goals:**
  - Ability to achieve high availability measured through reliability indicator
  - Heterogeneity of the load on every node; expect load close to max of resource capacity

![Graph showing reliability and optimal allocation difference](image-url)
Naming service (PhD Stud Silviu Panica)

**Problem:**
- Usually, common naming services are implemented in a centralized manner.
- DNS service uses a static configuration scheme and it is therefore not suitable to be deployed as a PaaS component.

**Solution:**
- distributed naming and resource identification service
- allows a client to consume or communicate with a resource without having to deal with resource location or discovery protocols
- create a DS on top of the DNS service that manages the naming ops (register, update and retrieve) in a distributed manner and by using the current DNS service as a backend.
What’s next?

 Full integration & testing & benchmarking & promotion of mOSAIC PaaS - deadline Spring 2013

 Develop further the open-source and deployable PaaS:

  ▪ Improve platform services - automated management - in the frame of [Ro] AMICAS (http://amicas.hpc.uvt.ro), 2012-2014
  ▪ Develop HPC services in Cloud using parts of the mOSAIC’s PaaS in the frame of [EC-FP7] HOST (http://host.hpc.uvt.ro), 2012-2014

 Develop top-level services based on the PaaS:


 Develop applications that are using the PaaS:

  ▪ Scientific cloud services [in submission]
Forthcoming HOST event: Workshop on HPC Services
Date: 27-29 September, 2012
Place: West University of Timisoara, Romania
In conjunction with: SYMASC 2012 – 18th Symposium on Symbolic and Numeric Algorithms for Scientific Computing
http://symasc12.info.uvt.ro

Topics include:
- Parallelization of compute- or data-intensive tasks in scientific applications
- Cloud computing and applications
- Multicore/multiprocessor architectures and GPU support for scientific applications
- Programming paradigms/tools/environments for high-performance scientific computing

Including:
- HPC training
- Grid tutorial

Important dates:
- Full paper submissions: 23 July 2012
- Notification: 15 August 2012
- Publication: selected papers will be included in SYMASC 2012 proceedings (IEEE CPS), extended versions of selected papers to published in SCIE journal

http://host.hpc.uvt.ro/avnt/sackle/