Cloud Computing
Architectures and Design Issues

Ozalp Babaoglu, Stefano Ferretti, Moreno Marzolla, Fabio Panzieri
{babaoglu, sferrett, marzolla, panzieri}@cs.unibo.it
Outline

- What is Cloud Computing?
- “A View from Bologna”
- Design Issues
  - Energy Reduction
  - P2P Cloud
  - QoS-aware Clouds
What is Cloud Computing?

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

The NIST Definition of Cloud Computing, Special Publication 800-145
Main Characteristics

- On-demand self-service.
- Broad network access.
- Resource pooling.
- Rapid elasticity.
- Measured service.
Service Models

Infrastructure as a Service (IAAS)

Platform as a Service (PAAS)

Software as a Service (SAAS)

Application

Platform

Infrastructure

Hardware

Amazon EC2

Microsoft Azure, Google AppEngine

Google Apps, SalesForce.com

Data Center

Deployment Model

- **Public Cloud**
  - Cloud infrastructure available to the general public

- **Community Cloud**
  - Cloud infrastructure available to a community of users with shared concerns

- **Private Cloud**
  - Cloud infrastructure available to a single organization

- **Hybrid Cloud**
  - Any combination of the above
Above the Clouds: A Berkeley View of Cloud Computing

Michael Armbrust
Armando Fox
Rean Griffith
Anthony D. Joseph
Randy H. Katz
Andrew Konwinski
Gunho Lee
David A. Patterson
Ariel Rabkin
Ion Stoica
Matei Zaharia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-28
A view from Bologna

- HPC
- Data analytics
- MMOG
- “Private” clouds
- Mobile gateways
- Loosely coupled applications

SLA mgmt
Virtualization
Centralized

SLA mgmt
Virtualization
Federated

SLA mgmt
Virtualization
Peer-to-peer

QoS policy management
Cloud core infrastructure
Virtualization Layer

Applications

Seminari DISI 2012 / Cloud Computing
A view from Bologna

- HPC
- Data analytics

- MMOG
- “Private” clouds

- Mobile gateways
- Loosely coupled applications

SLA mgmt

Virtualization

Centralized

Federated

Peer-to-peer

Cloud core infrastructure

Applications

QoS policy management

Virtualization Layer

Centralized

Cloud API
Energy Reduction through gossiping

- HPC
- Data analytics

- MMOG
- “Private” clouds

- Mobile gateways
- Loosely coupled applications

Virtualization

SLA mgmt

Virtualization

Virtualization

Cloud core infrastructure

QoS policy management

Cloud API

Electricity use breakdown

Source: EPA Report to Congress on Server and Data Center Energy Efficiency

*Toward energy efficient computing, ACM Queue, http://queue.acm.org/detail.cfm?id=1730791*
Virtualization

- Main feature of a Cloud system
  - *Dynamic scalability* (pay-as-you-go economic model)
  - *Virtualization* of resources

- Assumptions
  - Physical resources (servers) are multi-core/multi-processor machines; each server can support up to $C$ VM instances
  - Users request Virtual Machine (VM) instances
  - Users release instances when no longer needed
  - The VM monitor supports *live migration* of VMs

- Goal
  - Minimize energy consumption by *consolidating VMs*
VM Consolidation

(a) Before consolidation

(b) After consolidation of VM₁ and VM₄ to host 2
VM consolidation through gossiping

- Each server hosts the V-MAN daemon
- Daemons maintain an overlay network such that each daemon is connected to at most $K$ other nodes
- Daemons exchange messages only with neighbors
- The overlay is maintained with the *Newscast* algorithm
Example (1)

- Capacity = 4

Host 1

Host 2
Example (1)

- Capacity = 4

I have two VMs
Example (1)

- Capacity = 4

*I send you one*
Example (2)

- Capacity = 4

- Host 1
  - 2 VMs

- Host 2
  - 2 VMs
Example (2)

- Capacity = 4

- I have two VMs
Example (2)

- Capacity = 4

Host 1

Host 2

Send me one
Performance assessment

- We implemented V-MAN using the cycle-driven simulator engine provided by PeerSim (peersim.sf.net)
- Parameters:
  - $K=20$ (each node maintains a list of 20 neighbors)
  - $C=8$ (maximum capacity of each host is 8 VMs)
  - Topology is managed using Newscast
  - Length of each simulation run is 20 steps
  - Results are averages of 10 independent simulation runs
- Results:
  - $F_0$ Fraction of empty hosts
  - $F_{0,\text{opt}}$ Optimal fraction of empty hosts
Static System
Animation

Step 000

Seminari DISI 2012 / Cloud Computing
Dynamic system
Dynamic System with failures

Time step

$F_0$  -1000 servers  +2000 servers
A P2P Cloud

- HPC
- Data analytics

- MMOG
- “Private” clouds

- Mobile gateways
- Loosely coupled applications

SLA mgmt

Virtualization

SLA mgmt

Virtualization

SLA mgmt

Virtualization

Self-managing P2P Cloud

- Assemble a Cloud out of individual devices
  - E.g. low-power devices such as set-top boxes, ADSL modems, …
  - New business model to harness the computational power of otherwise idle devices

- Individual devices leave and join, but the Cloud keeps a coherent structure anyway
  - No central controller
Self-managing P2P Cloud

P2P Cloud—Goals

- Implement fully decentralized monitoring and Management capabilities
- Allocate $x\%$ of available nodes for a given task
- Allocate at least $n$ node for a given task
- How many nodes are currently busy?
- How many compute hours have been consumed by user $X$?
P2P Cloud—Architecture

Node

User Interface

P2PCS Daemon

Node-to-Node Interface

User
P2P Cloud—Architecture

Instance Management API | Monitoring API | Storage API
---|---|---
Dispatcher | T-Man | Aggregation Service
Slicing Service | | Monitoring System | Storage System

== implemented modules

Authentication / Authorization layer

Bootstrapping Service | Peer Sampling Service
P2P Cloud—Architecture

Gather an initial set of nodes to start the message exchange
P2P Cloud—Architecture

Provide each node with a list of peers to exchange messages with.
P2P Cloud—Architecture

- Instance Management API
- Monitoring API
- Storage API
- Dispatcher
- T-Man
- Slicing Service
- Peer Sampling Service
- Aggregation Service
- Monitoring System
- Storage System

Rank the nodes according to one attribute (e.g., 5% of the total n. of nodes; 1% fastest nodes; ...)
P2P Cloud—Architecture

Compute global measures (e.g., network size) using local message exchange
P2P Cloud—Architecture

<table>
<thead>
<tr>
<th>Implemented modules</th>
<th>Deployed modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance Management API</td>
<td>Monitoring API</td>
</tr>
<tr>
<td>T-Man</td>
<td>Storage API</td>
</tr>
<tr>
<td>Peer Sampling Service</td>
<td>Monitoring System</td>
</tr>
<tr>
<td>Aggregation Service</td>
<td>Storage System</td>
</tr>
<tr>
<td>Authentication / Authorization layer</td>
<td></td>
</tr>
</tbody>
</table>

Build an overlay network with a given topology (e.g., tree, ring, mesh...)

Slicing Service

Bootstrapping Service

Build an overlay network with a given topology (e.g., tree, ring, mesh...)

Instance Management API

Monitoring System

Aggregation Service

Storage System

System Management API

Instance Management API

Monitoring System

Aggregation Service

Storage System
Aggregation example
Computing the mean

\[ \frac{X+Y}{2} \]
P2PCS: Building subclouds

(a) 

(b) 

(c) 

slice 1

slice 2
P2PCS API

- **run-nodes subcloud_id number**
  - Creates a subcloud with *number* nodes; *subcloud_id* is set as the name of the newly created subcloud

- **terminate-nodes subcloud_id nodename1 ... nodenameN**
  - Removes the named nodes from the subcloud with given id

- **add-new_nodes subcloud_id number**
  - Adds *number* nodes to the subcloud identified by *subcloud_id*. The new nodes are chosen without any particular criteria

- **describe-instances nodename**
  - Prints a human-readable description of the given node

- **monitor-instances**
  - Return the global size of the Cloud using the aggregation service

- **unmonitor-instances**
  - Stops printing the global size of the Cloud
QoS-aware Clouds

- HPC
- Data analytics

- MMOG
- “Private” clouds

- Mobile gateways
- Loosely coupled applications

SLA mgmt

Virtualization

SLA mgmt

Virtualization

SLA mgmt

Virtualization

Applications
Cloud API
QoS policy management
Cloud core infrastructure
Virtualization Layer

S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, E. Turrini, QoS-aware Clouds, in Proc. 3rd Int. Conf. on Cloud Computing (IEEE Cloud 2010), Miami (USA), IEEE, July 2010.
Motivations

• QoS: crucial factor for the success of cloud computing providers
  • if not delivered as expected, it may impact the provider’s reputation

• Compliance to SLA
  • **SLA**: legally binding contract stating the QoS guarantees required by cloud customer
  • typically includes max response time, throughput, error rate
  • may include non functional requirements such as timeliness, scalability, availability
  • we address response time, only
SLA Example

<ContainerServiceUsage name="HighPriority" requestRate="100/s">
    <Operations>
        ...
    </Operations>
    ...
</ContainerServiceUsage>

• Customer obligations and rights
  – Maximum request rate from the client application to the virtual execution environment in the cloud
  – Operations the application is allowed to invoke
SLA Example

<ServerResponsabilities serviceAvailability="0.99" efficiency="0.95" efficiencyValidity="2">
  <OperationPerformance name="HighPriority" maxResponseTime="1.0s">
    <Operations>
      ...
    </Operations>
  </OperationPerformance>
  ...
</ServerResponsabilities>

• Responsibilities of a service running in the cloud
SLA Example

Probability that the service is available over a predefined time period

• Responsibilities of a service running in the cloud
SLA Example

<ServerResponsibilities serviceAvailability="0.99" efficiency="0.95"
  efficiencyValidity="2">
  <OperationPerformance name="HighPriority" maxResponseTime="1.0s">
    <Operations>
      ...
    </Operations>
  </OperationPerformance>
  ...
</ServerResponsibilities>

• Responsibilities of a service running in the cloud

Fraction of SLA violations that can be tolerated, within a predefined time interval, before the service provider incurs a penalty
SLA Example

```xml
<ServerResponsabilities serviceAvailability="0.99" efficiency="0.95"
  efficiencyValidity="2">
  <OperationPerformance name="HighPriority" maxResponseTime="1.0s">
    <Operations>
      ...
    </Operations>
  </OperationPerformance>
  ...
</ServerResponsabilities>
```

• Responsibilities of a service running in the cloud

Required service responsiveness
SLA Example

- SLA between the service provider and the cloud infrastructure hosting that service
QoS-aware Cloud Architecture

- Main components
  - Load balancer
  - Monitoring service
  - SLA policy engine
  - Configuration service
Load Balancer

- Implements the load dispatching and balancing functionalities
- Receives requests from clients and dispatches them to virtual resources, balancing the load
- Incorporates a SLA Policy Engine which analyzes logs of the Monitoring Service to identify SLA violations
Monitoring Service

- Monitors the environment to detect QoS deviations from what specified in the SLA
- The component within the Load Balancer monitors incoming requests and related responses
- There is a Monitoring Service instance for each virtual resource
Configuration Service

- Responsible for both configuration and run-time re-configuration of the application hosting environment
- Dynamically resizes the resources for the service by adding/removing them as needed
Experimental Evaluation

- Tool implements *principal components*
  - Load balancing policies
  - QoS handling policies
- Assumptions: required hosting SLA efficiency = 95%, VM allocation time = 2s
  - Other tests carried out with VM allocation time = 6s, 10s
  - VM allocation can take up to 400s
Preliminary Results

Load *progressively increased* until reaching 90 requests per sec, then progressively decreased.

**a) Response time**
- VMs allocated as load increases and released as it decreases.

**a) Violation Rate**
- Peaks occur → a new VM is added.
Preliminary Results

Augmented load till reaching 13 VMs

**a) Response time**
- VMs allocated as load increases and released as it decreases

**a) Violation Rate**
- Peaks occur → a new VM is added
Conclusions

- We described some recent results in the area of Cloud Computing
  - “Bologna View” of Cloud Computing
  - Energy reduction through VM consolidation
  - P2P Clouds
  - QoS management in Clouds
- Work in progress
  - QoS management using performance models @ runtime (e.g., QoS-aware energy reduction)
  - Cloud-enabled applications (mobility)
  - ...