Cloud Computing Overview

Hong-Linh Truong and Schahram Dustdar

Distributed Systems Group
Vienna University of Technology

truong@infosys.tuwien.ac.at
http://www.infosys.tuwien.ac.at/Staff/truong
 Goals

- Understand the evolution leading to cloud computing
  - Not detailed techniques but basic questions
- Understand cloud definitions and terminologies
  - Note they are evolving
- Understand enabling techniques for clouds
- Capture some open research questions related to “elastic” systems/applications in clouds
  - Cost monitoring and analysis
  - Quality and its compatibility
Overview – basic terms and models
Let's go for some fundamental questions

- If somebody offers you

- Which characteristics about the system/service you would wonder?
Some main characteristics

- Not complete and you can add more
- Why do we care about these characteristics?
### Computing element and storage element – some terms

<table>
<thead>
<tr>
<th>Apps</th>
<th>Apps</th>
<th>Apps</th>
<th>Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Middleware</td>
<td>Middleware</td>
<td>Operating System</td>
<td>Hardware</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Apps</th>
<th>Apps</th>
<th>Apps</th>
<th>Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Server</td>
<td>Workflow Engine</td>
<td>Scheduler</td>
<td></td>
</tr>
<tr>
<td>Enterprise Service Bus</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Apps</th>
<th>Apps</th>
<th>Apps</th>
<th>Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform</td>
<td>Platform</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Infrastructure</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parallel computing with many-core/CPU Systems

- Computing element:
  - SMP (Symmetric Multi-Processor) Machines, Multicore machines

- Characteristics:
  - Time-sharing and space-sharing parallel computing
    - Multiple users/multiple tasks
    - Virtualization

- Limited scalability
- Limited resource pool
- Limited on-demand resource/application provisioning
- Infrastructure ownership: single organization
- Broad access: typically within an organization
Cluster computing

- **Clusters**
  - A collection of interconnected computers
  - Utilized as a single, unified computing resources

- **Characteristics**
  - Single organization

- **Characteristics**
  - Good scalability
  - Fine-grained parallel processing techniques
  - Limited on-demand resource/service/application provisioning
  - Infrastructure ownership: single organization
  - Broad access: within an organization, limited services
Grid computing

- **Computing elements:**
  - A Grid consists of multiple Grid sites
  - Grid site: resources and services in an organization

- **Characteristics**
  - Mainly for scientific applications
  - Mainly free will resource sharing

- **High scalability and performance, loosely-coupled and fine-grained distributed and parallel processing techniques**
- **On-demand resource/service/application provisioning is not guaranteed in general**
- **Infrastructure ownership: distributed across organizations**
- **Broad access: across organizations**
- **Service contract: typically no guarantee**
Utility computing


The Sun Grid Compute Utility is:

- Unique: Sun Grid is the world’s first and only true compute utility.
- Accessible: Sun Grid’s enormous compute power is available via the Web portal at www.network.com.
- Affordable: $1/CPU-hr. You only pay for the compute power consumed.
- Powerful: SunFire dual processor Opteron-based servers with 4 GB of RAM per CPU, Solaris 10 OS, and Sun Grid Engine 6 software.
- Secure: There are multiple layered defenses at every level of Sun Grid. Nothing is more important than the security of your data.
- Easy: Just login, select your application and run the job. It’s that simple. Results arrive via email.

How to use the Network.com Application Catalog

This flash video shows how easy it is to tap into Network.com’s powerful and affordable compute resources and takes viewers through the process of selecting an application, creating and running a job, and downloading the results. » View the Demo

Define Utility Computing

Utility Computing is a business model for computing in which resources (CPU power, storage space, etc.) are made available to the user on an as-needed basis. The goal of the utility computing model is to maximize the efficient use of computing resources and minimize user costs. Users are able to dial up or dial down usage in real time, to meet the varying demands of business.
The IT' Elephant View

Cloud computing - “New Wine” or “New Bottle”

Source: Jeffrey Voas, Jia Zhang, "Cloud Computing: New Wine or Just a New Bottle?," IT Professional, pp. 15-17, March/April, 2009

Figure 1. Computing paradigm shift. Over six distinct phases, computers have evolved from dummy terminals to grids and clouds.
Let's think about a scenario

- A owns only a computing infrastructure (hardware and operating systems)
- B uses A's offers to create a platform (e.g., an application engine)
  - B's does not own the infrastructure
- C's uses B's offers to deploy its applications
  - C does not own the platform
- D's uses C's offers to run C's applications
  - D does not own the application
- etc.
Adding some characteristics

- Thousands of As, Bs, and Cs
- Even a million :-) of D-liked customers around the world
- C's applications: e.g., financial, scientific, and collaboration services,
- B's does not want to rent 1000's CPUs but scale up and down the number of CPUs automatically
- etc.

→ does the scenario differ from those for “cluster computing” and “grid computing”?
NIST Cloud definitions

“This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.”


Data-as-a-Service?

Human-as-a-Service?

NIST Cloud Definition

Deployment Model
- Private cloud
  - Community cloud
- Public cloud
- Hybrid cloud

On-demand self-service
- Resource Pooling
- Broad network access
- Rapid Elasticity
- Measured Service

Model
- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

Characteristics
Some enabling techniques

- On-demand self-services
  - Self-*, automatic service composition
- Resource pooling
  - Virtualization, Cluster/Grid techniques, data center management
- Broad network access
  - SOA, mobile, Internet technologies, interoperability APIs
- Rapid elasticity
  - Self-*, resource management, performance monitoring
- Measured service
  - Service contract, monitoring, billing
Infrastructure as a Service

- **NIST IaaS**

  “The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).”
Virtualization is a powerful term: we can apply virtualization techniques virtually for everything!
Platform as a service

- NIST

“The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.”
Platform as a Service

- Enabling techniques
  - Cloud programming techniques
  - On-demand application deployment, composition and execution
  - Job management

- Some questions:
  - Is any new cloud-specific programming language?
  - Can we just use conventional deployment techniques?
Software as a Service

- **NIST**

“The capability provided to the consumer is **to use the provider’s applications running on a cloud infrastructure**. The applications are **accessible from various client devices through** a thin client interface such as a web browser (e.g., web-based email). The consumer **does not manage or control** the underlying cloud infrastructure including network, servers, operating systems, storage, or **even individual application capabilities**, with the possible exception of limited user-specific application configuration settings.”
Software as a Service

- Enabling techniques
  - Web services, Mashup
  - Rich Internet applications
Data-as-a-service

- Not just an infrastructure service for storing data.
Some enabling techniques – monitoring and analysis

- Is that similar to cluster/Grid monitoring and analysis?

Source: http://www.3tera.com/AppLogic/Monitoring.php

Some enabling techniques - service contract


Deployment - Private cloud

- NIST “The cloud infrastructure is **operated solely for an organization.** It may be managed by the organization or a third party and may exist on premise or off premise.”
NIST: “The cloud infrastructure is *shared by several organizations and supports a specific community* that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.”
NIST: “The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.”
NIST: “The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).”
Sky Computing – or Grid of Clouds?

Source: Katarzyna Keahey, Mauricio Tsugawa, Andrea Matsunaga, and Jose Fortes. 2009. Sky Computing. IEEE Internet Computing 13, 5 (September 2009), 43-51. DOI=10.1109/MIC.2009.94 http://dx.doi.org/10.1109/MIC.2009.94

Figure 1. A virtual cluster interconnected with ViNe. An end user employs the Nimbus client, contextualization service, and images available in the marketplace. We can see (1) preparation, (2) deployment, and (3) usage.
Multiple/hybrid cloud systems

- Interoperability protocols for multiple level of abstractions
  - Virtual machines, networks, cloud management APIs
- Security across multiple domains
- Complex data governance and service contract issues
- Complex billing and monitoring

What if a computer is a human?
A human is a „computing element/processor“ that can process many types of tasks, of course, with different cost and quality!
Our focus – building elastic, complex applications in clouds
System and application composition model in the Cloud (1)

- **System and application composition**
  - Includes processes for creating systems/applications from different system/application components

- A cloud-based composition includes **different classes** of system/application components **from different providers**

- Component integration in a cloud system/application composition
  - Tightly coupled: a system/application is built from linking existing libraries, modules, components, e.g., by means of a compiler
  - Loosely coupled: an application is built from a composition of existing programs and services by means of an executable (high-level) language, e.g., workflow of programs and services

- Forms of components/services:
  - Software (composite) services, human-based services, social compute units (teams of individuals), hybrid (composite) services (software/robot + humans)
System and application composition model in the Cloud (2)

- Composition structure describes the dependency among components
  - Dependency types: data, control, deployment, non-functional constraints
  - Multiple levels: infrastructures, platforms, and DaaS/SaaS
  - Abstract or executable languages for composition description
- A composition structure can be changed during runtime
  - Dynamic process refinements
- Information about the composition structure is used to select suitable components for constructing and executing systems/applications
- We aim at developing techniques for automating processes
  - Even if a human is a “processor” used in a process
System and application composition model in the Cloud (3)

- Phases
  - Modeling the composition structure and selecting components
  - Constructing and deploying the system/application
  - Executing and refining the system/application

- Some considerations
  - Phases are not conducted in a pipeline
  - Separated or interwoven runtime with design-time
  - Adaptiveness/Elasticity
    - When and based on what?

- Systems/applications must be elastic to serve cloud consumers and they must utilize underlying services and data in an elastic way!
Some considerations – cost issues for complex applications/processes
Example: complex applications in clouds

- Simulate the stiffness of human bones
- Data and computation intensive applications: including sequential and parallel programs (e.g., parfe and paraview), run under batch and interactive modes
- Scientists want to use hybrid clouds: local, partial cloud, and fully cloud
Cost estimation and monitoring needs

“which part of my experiments should be run on the cloud with respect to cost criteria”? “how much does it cost to actually run my application”? 

- Consumers and elastic cloud deployment and management tools need to
  - Decide whether to use local, partial cloud or full cloud storages and machines
  - Optimize based on different trade-offs (cost, performance, urgency, jurisdiction)
  - Achieve dynamic provisioning and resource mapping

- Even in pay-as-you-go model, cost is not just dependent on the performance (execution time) of the application
Current research

- Well-developed Grid and cluster performance monitoring and analysis
  - Several performance tools exist, including monitoring and analysis and prediction
  - Well-researched event representations, instrumentation techniques and performance models for specific application models (OpenMP, MPI, workflows)
  - Several performance benchmarks and performance comparison among different systems
- Lack of cost monitoring tools for cloud computing
  - Simple cost models from cloud providers
  - Mainly performance and cost analysis of specific applications, performance benchmarks for specific cloud systems
- Missing cost estimation and monitoring tools for application models (e.g., composition of different models) in the cloud
- Missing monitoring techniques to deal with elasticity behaviours
Complex issues on cost

- The implication of using cloud machines
  - The concept of data locality is extremely important
    - Moving data close to compute or move computational resources close to data source
  - In the cloud, the cost of storages and machines seem cheap but, with respect to performance and cost, does it make sense to
    - Have data hosted the cloud but not use cloud computational resources (e.g., just share data)?
    - Use cloud computational resources but data is hosted in on-premise storages (e.g., due to law compliance)?
- Diverse cost models offered by different clouds
Complex issues on cost evaluation

- **Layers:**
  - Applications, platforms, machines, storage, networks

- **Activities**
  - Computing, storage, data transfer, application packaging and deployment, temporary and final result cleaning

- **Multiple cloud providers**

- **How to**
  - Correlate the cost elements across layers?
  - Break down cost elements for concrete activities?
  - Evaluate cost on-the-fly
Example – Conceptual architecture

• Leverage our previous knowledge on event representations, application monitoring, performance analysis, dependability analysis

• Employ service-oriented approach
  – RESTful service, JSON and XML event data
Example – Composable cost models

- Data storage: \( M_{ds} = \text{size(total)} \times t_{sub} \times \text{cost(storage)} \)

- Basic computational cost: \( M_{cm} = \text{cost(machine)} \)

- Basic data transfer in cost: \( M_{dfi} = \text{cost(transfer}_{in} \) \)

- Basic data transfer out cost: \( M_{dfo} = \text{cost(transfer}_{out} \) \)

- Basic single data transfer
  \( M_{sd} = \text{size(in)} \times M_{dfi} + \text{size(out)} \times M_{dfo} \)
Example – Composable cost models

- Monitoring cost for sequential/multi-threaded applications

\[ M_{sm} = t_e \times M_{cm} + \text{size(out)} \times M_{dfo} + \text{size(in)} \times M_{dfi} \]

- Estimated cost for sequential/multi-threaded applications

\[ M_{se} = f_{pi} \times M_{cm} + \text{size(out)} \times M_{dfo} + \text{size(in)} \times M_{dfi} \]

→ \( f_{pi} \) is an estimated performance improvement function

- e.g. an ideal estimated \( f_{pi} \) for \( n \) threads based on \( p \) threads experiments is \( (p \times t_e(p))/n \).

- Our assumption: we rely on external knowledge for this, e.g., by using external tools/scientist's knowledge/benchmarks
Example – Composable cost models

- Monitoring cost for parallel/MPI programs on multiple machines
  \[ M_{pm} = n \times t \times M_{cm} + \text{size(out)} \times M_{dfo} + \text{size(in)} \times M_{dfi} \]

- Estimated cost for parallel/MPI programs on multiple machines
  \[ M_{se} = n \times f \times M_{cm} + \text{size(out)} \times M_{dfo} + \text{size(in)} \times M_{dfi} \]

- Monitoring cost for workflow – sum of costs for all activities
  \[ \sum \left( \text{size(in}_i \right) \times M_{dfi} \right) + \sum \left( \text{size(out}_i \right) \times M_{dfo} \right) + \sum \left( M_{cm} \times t_e \left( \text{machine}_i \right) \right) \]
Example – Composable cost models

Estimated cost for workflows by using the hierarchical structure workflow region model

- Cost of all regions:

$$\sum cost(wr_i)$$

- Cost for a region

$$cost(wr) = \sum cost(activity_j)$$

Using $M_{mp}$, $M_{sm}$ and $M_{sd}$ for parallel activity, sequential activity, and data transfer
Example – Cost estimation

- Assumption: knowledge from scientists and possible external prediction/benchmark tools
- Estimation based on events generated from dependency graphs
  - Nodes: storages, machines, abstract application execution models (sequential, parallel, workflow)
  - Edges: data and control flows
  - Nodes and edges include performance information and payment models (subscription/pay-as-you-go)
Contract compatibility issues for complex applications
Complex context and quality properties

Data resource

- Person Context
- Device Context
- Usage Context

- Composite Service
- Web Service
- Resource

- Consumer
- Provider/Integrator

- Quality
  - QoS
  - QoD
DaaS concerns and contracts
Characteristic of software components and data in the Cloud

- Different classes of components
  - virtual machines versus system/application components
  - licensed software components, pay-per-use software components, user-provided components
- Different degrees of quality of services and quality of data
- Different business models
- Different intellectual property rights
- Different law enforcements
Compatibility issues for data and service in clouds

- Types of compatibilities
  - Compatibility with consumer requirements
  - Compatibility among application components in an application

- Compatibility
  - Functionality and non-functional parameters

- Design versus runtime compatibility
  - Let's consider “system/application components” delivered under the service model
    - So we can focus more on runtime aspect and cloud business model

→ Compatibility must be ensured when processes/applications are elastic
Two “compatibility worlds”

- User interface, Web services consumers
- Service/Data Composition and Execution
  - FCA (Function Compatibility Analyzer)
  - NCA (Non-functional Compatibility Analyzer)
- Service/Data Discovery and Management
- Service/Data Information
- Monitoring

Functional parameters
- Contractual terms

Services and contracts
- Compatible services/data
- Contractual terms
- Compatible contracts
Being able to utilizing multiple services in an elastic way

- A composite service/process involves multiple services from different providers under different contracts
  - Not just contract negotiation between consumer and service in a point-to-point manner.
- A service contract
  - establishes the understanding between a service consumer and a service provider;
  - specifies conditions on non-functional parameters (NFPs)/concerns, such as: Quality of Service, Business terms, Context terms, License terms, Quality of data
- Several standard/proposed languages for service contract descriptions, e.g., WSLA, WSOL, ODRL-S, WS-Policy)
Compatibility for being elastic

- Service contract compatibility needs to consider:
  - three aspects – **service APIs, data APIs, and provided data concerns**;
  - a rich set of contract properties (e.g., QoS, **Data quality**, **Business, License and Context terms**);
  - several service contract specification languages (e.g., WSLA, WSOL, ODRL-S) together.

- QoS, Business, License and Context terms differently influence data/control flows of the service composition.

<table>
<thead>
<tr>
<th>Quality of Service (QoS)</th>
<th>control flow</th>
<th>data flow</th>
<th>independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Context</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Business</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>License</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Data and control flows in contract compatibility evaluation
SeCO$_2$ – modeling and mapping service contracts

- **Problem**: Heterogeneity in service contract specifications.

- Three types of languages for the specification of service contract properties:
  - **Type A** (e.g., ODRL-S): includes languages allowing the specification of predefined properties.
  - **Type B** (e.g., WSLA): includes languages allowing the specification of user-defined properties.
  - **Type C** (e.g., WSOL): includes languages allowing the specification of properties defined in user ontologies.

- Ontology alignment tools cannot be used to fully automate the mapping between different specifications.
SeCO\textsubscript{2} – modeling and mapping service contracts

- **Solution:** SeCO\textsubscript{2} makes service contracts comparable through the wrapping to specifications (i.e., *SeCO Policies*) built on a common meta-model
  - without loss of information;
  - by means of the *SeCO Reference Ontology* and predefined mapping rules;
  - supporting the use of lexical databases (e.g., *WordNet*) and ontology alignment tools (e.g., *H-match*).
SeCO$_2$ – mapping service contracts

- Specifications in **Type A** are wrapped applying fixed mapping rules.
- Specifications in **Type B** and **Type C** can require interactions with service providers to handle the absence of knowledge (i.e., mapping rules).
  - The definition of new mapping rules is supported by lexical databases and ontology alignment tools.
Evaluating service contract compatibility: activities and flows

Service Contract Mapping

Service Contract Compatibility Evaluation (at the service level as a whole)
Evaluating service contract compatibility

- **Input:**
  - service composition description in terms of data and control flows;
  - contracts of the services involved in the composition.

- **Output:**
  - compatible/incompatible service contract properties.

- The compatibility is checked considering
  - semantic relations among values associated with qualitative contract properties;
  - constraint operators used to define quantitative contract properties;
  - data and control flows of the service composition.
## Compatibility evaluation rules

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Data Flow</th>
<th>Control Flow</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Location</strong></td>
<td>Service Context</td>
<td></td>
<td></td>
<td>Partnership</td>
</tr>
<tr>
<td><strong>Pricing</strong></td>
<td>Business</td>
<td>X</td>
<td></td>
<td>Compatible value list</td>
</tr>
<tr>
<td><strong>Payment (for data usage)</strong></td>
<td>Business</td>
<td>X</td>
<td></td>
<td>Binary, Ternary</td>
</tr>
<tr>
<td><strong>Payment (for service usage)</strong></td>
<td>Business</td>
<td></td>
<td>X</td>
<td>Binary, Ternary</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>QoS</td>
<td></td>
<td>X</td>
<td>Binary, Ternary</td>
</tr>
<tr>
<td><strong>Permissions</strong></td>
<td>License</td>
<td></td>
<td>X</td>
<td>Subsumption</td>
</tr>
<tr>
<td><strong>Data Ownership</strong></td>
<td>License</td>
<td></td>
<td>X</td>
<td>Compatible value list</td>
</tr>
</tbody>
</table>

- But how this way of compatibility evaluation can be used for dealing with multiple forms of services at runtime for elastic processes?
Summary

- To distinguish cloud computing with other computing models
  - Built based on several existing enabling techniques
  - New “business models” in the Internet are the key driver that leads to the development of several new techniques
- Utilizing cloud data and service in order to be elastic
  - Access to a very large number of resources concurrently
  - But cost and quality issues must be addressed at runtime and during the refinement of applications/processes

- For being elastic, we need to solve much more open problems. See some in our next lectures
References

- Jeffrey Voas, Jia Zhang, "Cloud Computing: New Wine or Just a New Bottle?", IT Professional, pp. 15-17, March/April, 2009
- Hong-Linh Truong, Schahram Dustdar "On Evaluating and Publishing Data Concerns for Data as a Service", APSCC 2010
Thanks for your attention!

Hong-Linh Truong
Distributed Systems Group
Vienna University of Technology
Austria

truong@infosys.tuwien.ac.at
http://www.infosys.tuwien.ac.at