During this course

- Treads in IT
- Towards a new data center
- What is Cloud computing?
- Types of Clouds
- Making applications Cloud-ready

Trends in IT

Internet is everywhere

Frequency barrier hit

Management costs increase
Power & Cooling expensive

Increasing prosumarism

Towards a new data center

Business face rapid growth

... regular peaks
- Daily, weekly and yearly patterns

... expected peaks
- Marketing campaigns
  - Christmas
- Tax filling
... unexpected peaks

- Triggered by news/events

Source: Google blog

... and need to provision for such load

- Over-provisioning: wasted resources
- Under-provisioning: lost clients

Legend:
- actual load
- provisioned load

Ideally: auto-scaling

What is Cloud computing?

NIST definition
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.

You do not own computing infrastructure, you rent it

Essential Characteristics
- On-demand self-service
  - User can provision resources without human intervention
- Broad network access
  - Service is accessible through the network
- Resource pooling
  - Resources are pooled to several customers
- Rapid elasticity
  - Can rapidly increase and decrease capacity
- Measured service
  - Resource usage can be monitored, reported

Service models
- SaaS: Software as a Service
  - Ready to use applications
    - E.g.: Gmail, Google Docs, YouTube, Facebook
- PaaS: Platform as a Service
  - Ready to use tools
    - E.g.: Google Apps Engine
- IaaS: Infrastructure as a Service
  - Fundamental computing resources
    - E.g., Amazon EC2, Amazon S3, Windows Azure
### Deployment models
- Private cloud ("my stuff")
  - Institution resources managed in a "Cloud-like" fashion
- Community cloud ("our stuff")
  - Share resources among institutions
- Public cloud ("anyone's stuff")
  - Resources to rent by anybody with a credit card
- Hybrid cloud
  - A combination of the above

### Types of Clouds
- Private cloud ("my stuff")
- Community cloud ("our stuff")
- Public cloud ("anyone's stuff")
- Hybrid cloud

### Infrastructure as a Service (IaaS)
- Fundamental computing resources
  - CPU, RAM, storage, network, etc.
- Mostly in an virtualized way
  - Details of the underlying physical resources are hidden from the user

### Virtualization
- Hypervisors: Xen, VMware, KVM, Microsoft Hyper-V
- Isolates different users
- Flexible infrastructure management:
  - consolidation
  - At VM start: mapping
  - During VM execution: migration

### Public IaaS Clouds
- Amazon: EC2, S3
- Windows Azure
- RackSpace
- E.g.: Amazon EC2
  - 1 year micro instance for free
    - Try it yourself!
    - Small: 0.60 $/hr
    - BXL: 4.60 $/hr
- For comparison:
  - BXL in your own datacenter: 1.50 $/hr (estimated)
  - Abisko @ HPC2N: 0.07 $/core/hr (source: P-O)

### Open-source solutions
- Useful to build private/community/hybrid clouds
- Production-oriented
  - Eucalyptus, CloudStack, OpenStack
- Research-oriented
  - OpenNebula, Nimbus
**IaaS Cloud standards**

- De-facto: Amazon
  - WSDL-based
  - createInstance()
  - startInstances(), stopInstances()
- Open Cloud Compute Interface (OCCI)
  - REST-based

```
POST /compute HTTP/1.1
Category: compute ...
X-OCCI-Attribute: occi.compute.cores=2
X-OCCI-Attribute: occi.compute.hostname="foobar"
HTTP/1.1 201 OK
Location: http://example.com/vms/foovm/
```

**"Cloud-ready" applications**

- Almost any application can run in the Cloud, but they are **not** "Cloud-ready"
- Use API to auto-scale
  - Application needs to be elastic

**1 VM per tier**

Client

Load balancer

Application

Master DB

Replicate

Slave DB 1

Application

Replicate

Load balancer

Application

Master DB

Replicate

Slave DB 1

Application

Replicate

Slave DB n-1

Application

Read

Write
### IaaS pros and cons

- **Pros**
  - General purpose
  - Very flexible

- **Cons**
  - Low-level, fine-grained
  - Need to handle application
    - Deployment
    - Elasticity
    - Failure

### Platform as a Service (PaaS)

- Provides programming languages, libraries, services and tools
- **Higher level than IaaS**
- Special purpose
- Less flexible

- **Examples**
  - For web applications: Google Apps Engine
  - For BigData processing: MapReduce

### Google Apps Engine

- Develop and host web applications
- Take advantage of Google technologies for scalability, resilience
- **Python, Java, Go**
- Several APIs
  - Images, Mail, Log
  - Blobstore, Datastore, BigTable

### Hello Google Apps Engine

```python
import webapp2

class MainPage(webapp2.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'text/plain'
        self.response.write('Hello, webapp2 World!')

app = webapp2.WSGIApplication([('/', MainPage)], debug=True)
app.yaml
application: helloworld
version: 1
runtime: python27
api_version: 1
threadsafe: true
handlers:
  - url: /.*
    script: helloworld.app
```

**Try it, it's free!**

### MapReduce

- A programming model for processing large data sets (BigData)
  - Difficult due to scalability, failure, etc.
- Useful for: Indexing, Financial Analysis, etc.
- Used by Google, Yahoo, Facebook
- Open source implementation: Hadoop

### MapReduce: Logical View

- Operates on several key-value spaces:
  1. Input, 2. Intermediate, 3. Output
- **Map**: (key_i, value_i) -> (key_o, value_o)
- **Reduce**: (key_o, list(value_o)) -> list(value_o)
MapReduce: Word count

```java
map(String name, String document):
    for each word w in document:
        emit (w, 1)

reduce(String word, list partialCounts):
    sum = 0
    for each partialCount in partialCounts:
        sum += partialCount
    emit (word, sum)
```

Spark

- **General-purpose** BigData engine
- Most operations in-memory
- Allows interactive use

```python
file = spark.textFile("hdfs://...")
errors = file.filter(lambda line: "ERROR" in line)
# Count all the errors
errors.count()
# Count errors mentioning MySQL
errors.filter(lambda line: "MySQL" in line).count()
# Fetch the MySQL errors as an array of strings
errors.filter(lambda line: "MySQL" in line).collect()
```

Spark: More Examples

```python
file = spark.textFile("hdfs://...")
errors = file.filter(lambda line: "ERROR" in line)
# Count all the errors
errors.count()
# Count errors mentioning MySQL
errors.filter(lambda line: "MySQL" in line).count()
# Fetch the MySQL errors as an array of strings
errors.filter(lambda line: "MySQL" in line).collect()
```

Clouds vs. Grids

<table>
<thead>
<tr>
<th></th>
<th>Clouds</th>
<th>Grids</th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin</td>
<td>Industry</td>
<td>Academic</td>
</tr>
<tr>
<td>Economic model</td>
<td>Public: pay-per-use</td>
<td>Other: domain-specific, (e.g., grant models)</td>
</tr>
<tr>
<td>Size of allocation</td>
<td>Many &quot;small&quot; users</td>
<td>Few &quot;big&quot; users</td>
</tr>
<tr>
<td>Waiting time</td>
<td>On-demand</td>
<td>Queueing</td>
</tr>
<tr>
<td>Duration</td>
<td>Long (months)</td>
<td>Short (&lt; 4h)</td>
</tr>
<tr>
<td>Usual workload</td>
<td>Interactive</td>
<td>Long-lived services, Non-interactive Jobs</td>
</tr>
<tr>
<td>Organization</td>
<td>One provider</td>
<td>Multiple providers</td>
</tr>
</tbody>
</table>

Final thoughts

Cloud issues

- **Privacy**
  - What happens to your data?
- **Security**
  - Sometimes VM provide insufficient isolation
- Standardization is lagging behind
  - Important to prevent vendor lock-in
  - IaaS: less of a problem
  - PaaS: major problem
Conclusion

- IT trends
  - Frequency barrier
  - Infrastructure complexity
  - Omnipresent network connection
- What are Clouds: rent computing resources, rapid elasticity
- Types of Clouds: IaaS, PaaS, SaaS
- Making an application Cloud ready
- Challenges remaining