Evaluation of Cloud Systems

Prof. Florin POP
florin.pop@cs.pub.ro

cHiPSet Training School 2016
New Trends in Modeling and Simulation in HPC Systems
Bucharest, Romania, 21-23 September 2016

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

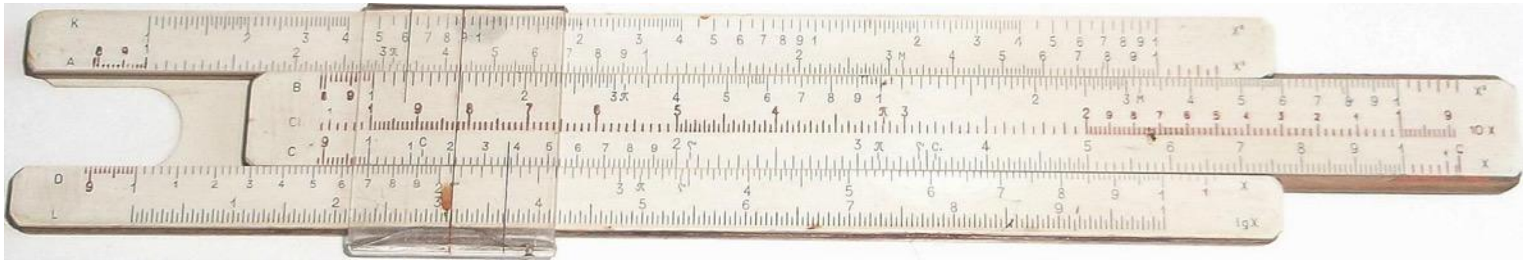
Computer Science & Engineering Department

Evaluation of Cloud Systems

… it will be about metrics,

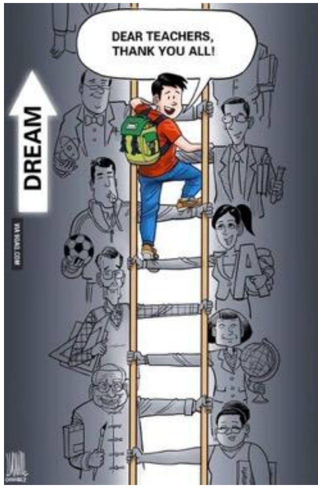
… evaluation metrics for Cloud systems,

… and how to compute their performance.



Acknowledge the work behind this publication

- Prof. Valentin Cristea and Prof. Ciprian Dobre
- Colleagues and PhD Students from our DSLab, especially:
  - Cătălin Negru
  - Mihaela Vasile
  - Cătălina Niță



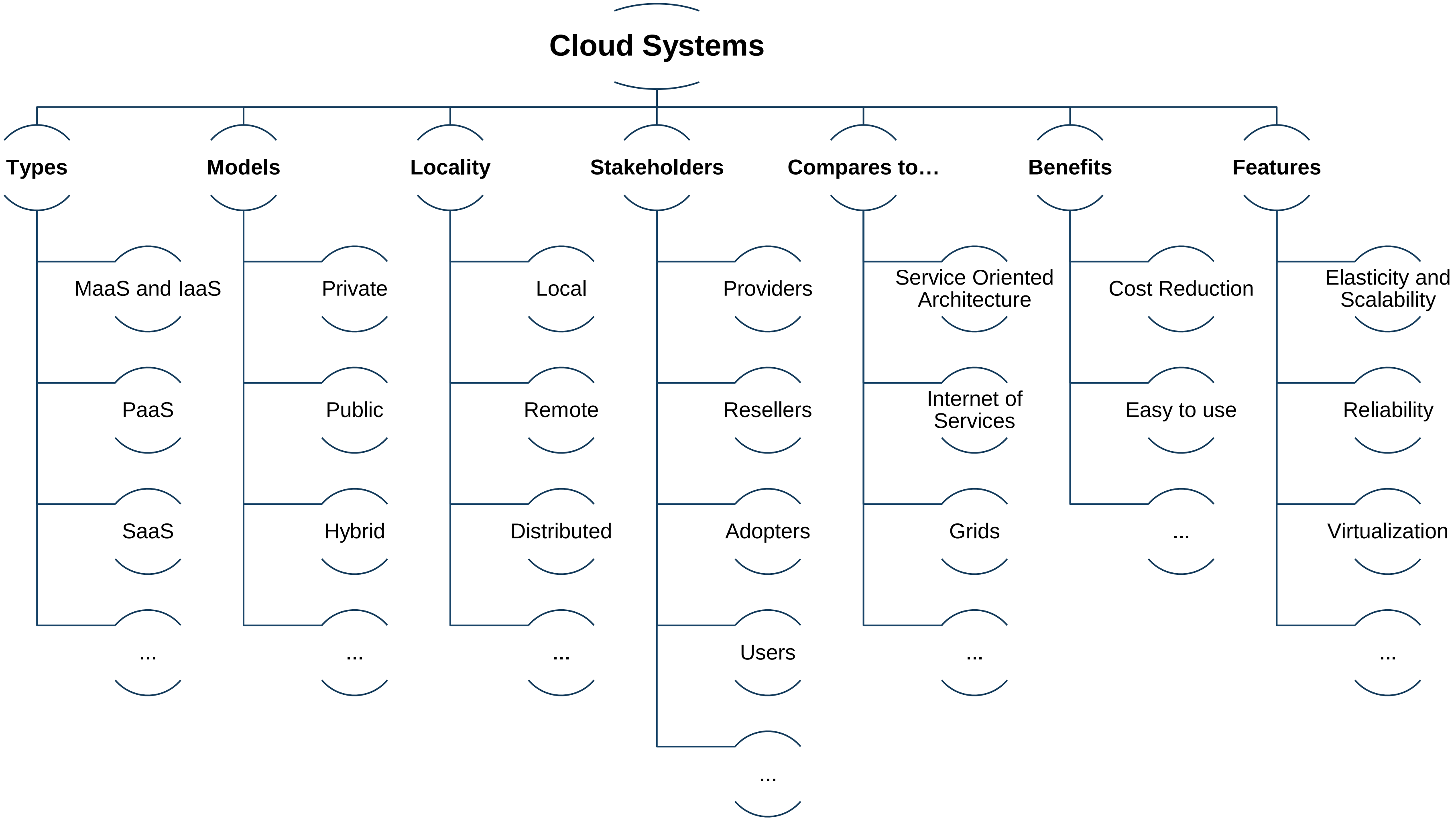
- Our current research interests (brief overview)
  - Big Data & Cyberinfrastructure Platforms and Applications
  - Resource Management and Data Handling in Heterogeneous Distributed Systems
  - Pervasive Systems, Technologies and Application

What is Cloud Computing?

- Providing Software as a Service (SaaS) - delivering different type of applications over the Internet.
- More recently also Hardware infrastructure (IaaS and MaaS), platform as a service (PaaS).
- Based on Utility Computing - pay-as-you-go:
  - Infinite resources (as much as you need),
  - Billing (e.g. hourly).

But "… nobody understand the Cloud!"

Brief overview of Cloud Systems



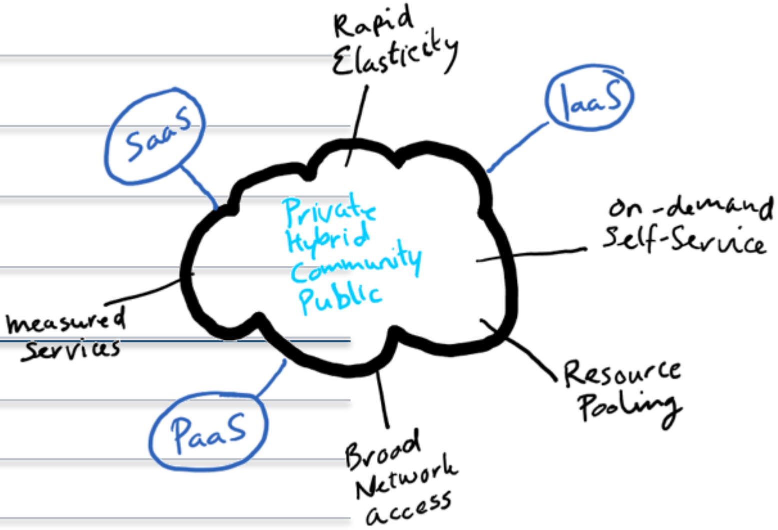
Different Cloud Services and Systems

Application Service (SaaS)	MS Live/Exchange Labs, IBM, Google Apps; Salesforce.com, Quicken Online, Zoho, Cisco
Application Platform	Google App Engine, Mosso, Force.com, Engine Yard, Facebook, Heroku, AWS
Server Platform	3Tera, EC2, SliceHost, GoGrid, RightScale, Linode
Storage Platform	Amazon S3, Dell, Apple, ...

- **Amazon**:
  - Computing in the cloud!
  - EC2 (with S3, SQS and SimpleDB), use Xen VMs,
  - Workflow and Security.
- **Google**:
  - Apps: Python module/API,
  - Working also with IBM.
- **Microsoft**:
  - Azure!
- **Yahoo**:
  - Pipes, Working with Computational Research Laboratories
- **Oracle/IBM/HP** and others…

Cloud Computing Characteristics/Issue

Nonfunctional	Elasticity (ex: Amazon EC2)
	Reliability (ex: Vmware ecosystem)
	Quality of Service (ex: Amazon S3)
	Agility and adaptability (ex: FlexNet)
	Availability (ex: MS Azure)
Economic	Cost reduction
	Pay per use
	Improved time to market
	Return of investment (ROI)
	Turning CAPEX (capital expenditure) into OPEX (operational expenditure)
	Going Green
Technological	Virtualization (ex: Virtual Box)
	Multi-tenancy (ex: MS SQL)
	Security, privacy and compliance
	Data Management (ex: WebSphere)
	APIs and / or Programming Enhancements (ex: Hadoop)
	Tools



More about Clouds

- Why is Cloud becoming a Big Deal?
  - Using high-scale/low-cost providers,
  - Any time/place access via web browser,
  - Rapid scalability; incremental cost and load sharing,
  - Can forget need to focus on local IT.
- Concerns and open issues:
  - Performance, reliability, interoperability
  - SLA negotiation,
  - Control of data, and service parameters,
  - Application features and choices,
  - No standard API – mix of SOAP and REST!
  - Privacy, security, trust…

Performance and SLA

- Agree on performance and availability SLAs
  - For **whom**?
  - What **function**?
  - From **where**?
  - From what **component**?
  - Will have what **performance**?
  - And what **availability**?
  - In what **timeframe**?
- Clearly state your recourses
  - “Using the $100/mo. subscription.”

General Features of Cloud Services (1/4)

- Availability
  - the degree to which a system is in a specified state.
  - *Metrics*: Flexibility, Accuracy, Response time.
- Reliability
  - the power to remain functional with time without malfunction.
  - *Metrics*: Service Constancy, Accuracy of Service, Fault Tolerance, Maturity, Recoverability.
- Efficiency
  - the ratio of the useful work performed by a system to the total energy expended or heat taken in.
  - *Metrics*: Utilization of Resource, Ratio of waiting time, Time behavior.

General Features of Cloud Services (2/4)

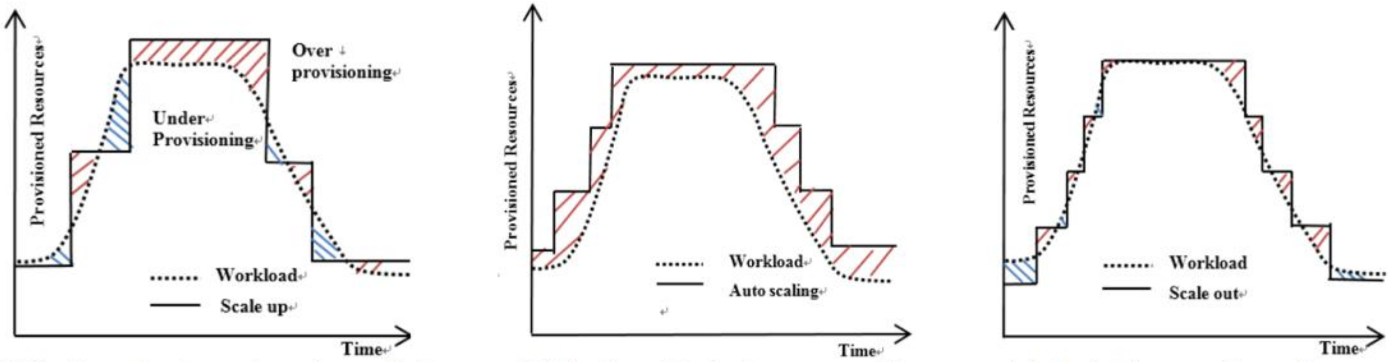
- Reusability
  - the level to which a component may be used in a number of systems or applications.
  - *Metrics*: Readability, Coverage of variability, Publicity.
- Interoperability / Composability
  - the capability to integrate with different standards and technologies.
  - *Metrics*: Service Modularity, Service interoperability, LSSI.
- Adaptability
  - the level of efficiency in adjusting a solution for the utilization in different context.
  - *Metrics*: Coverage of Variability, other performance metrics.

General Features of Cloud Services (3/4)

- Usability
  - the quantity to which a Cloud service could be used by particular consumers to gain certain aims with usefulness.
  - *Metrics*: Operability, Attractiveness, Learnability.
- Modifiability
  - the capability to make modifications to a component rapidly and cost-effectively.
  - *Metrics*: MTTC (Mean Time To Change).
- Sustainability
  - environmental effect of the Cloud service (usual carbon footprint or even energy capable of the Cloud services).
  - *Metrics*: DPPE (Data Centre Performance per Energy) parameter, PUE (Power Usage Efficiency).

General Features of Cloud Systems (4/4)

- Scalability
  - the capability of a system to handle a growing amount of resources and workloads.
  - *Metrics*: Average of assigned resources among the requested resources.



(a) Scale-up instance type (capacity)

(b) Scale-out in instance quantity

(c) Brutal-force auto-scaling

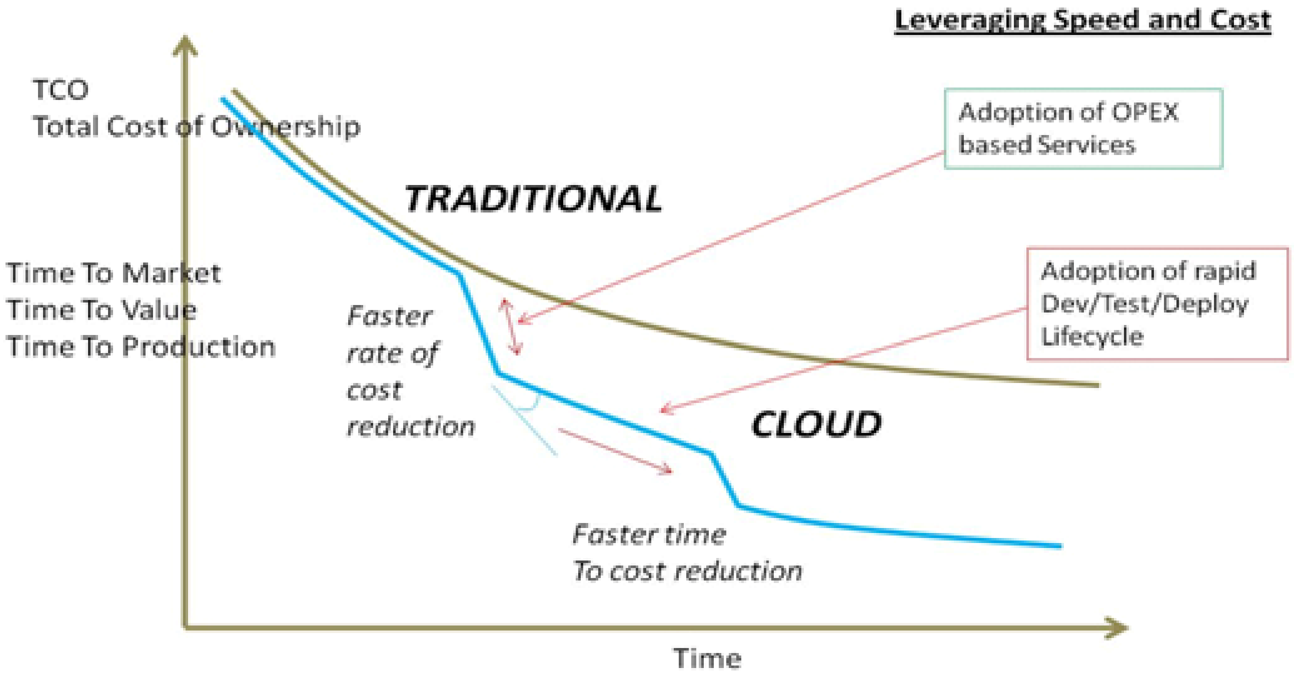
- Elasticity
  - "the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner, such that at each point in time the available resources match the current demand as closely as possible".
  - *Metrics*: Boot Time (second), Suspend Time (second), Delete Time (second), Provision (or Deployment) Time (second), Total Acquisition Time (second).

Performance Features of Cloud Systems

- Communication
  - *Metrics*: Packet Loss Frequency, Connection Error Rate, MPI Transfer bit/Byte Speed, MPI Transfer Delay
- Computation
  - *Metrics*: CPU Load (%), Benchmark OP (FLOP) Rate, Instance Efficiency (% CPU peak)
- Storage
  - *Metrics*: Response time, Latency, Bandwidth, Capacity,
- Memory
  - *Metrics*: Mean Hit Time (s), Memory bit/Byte Speed (MB/s, GB/s), Random Memory Update Rate, Response Time (ms)
- Time
  - *Metrics*: Computation time, Communication time

Economic Features of Cloud Services

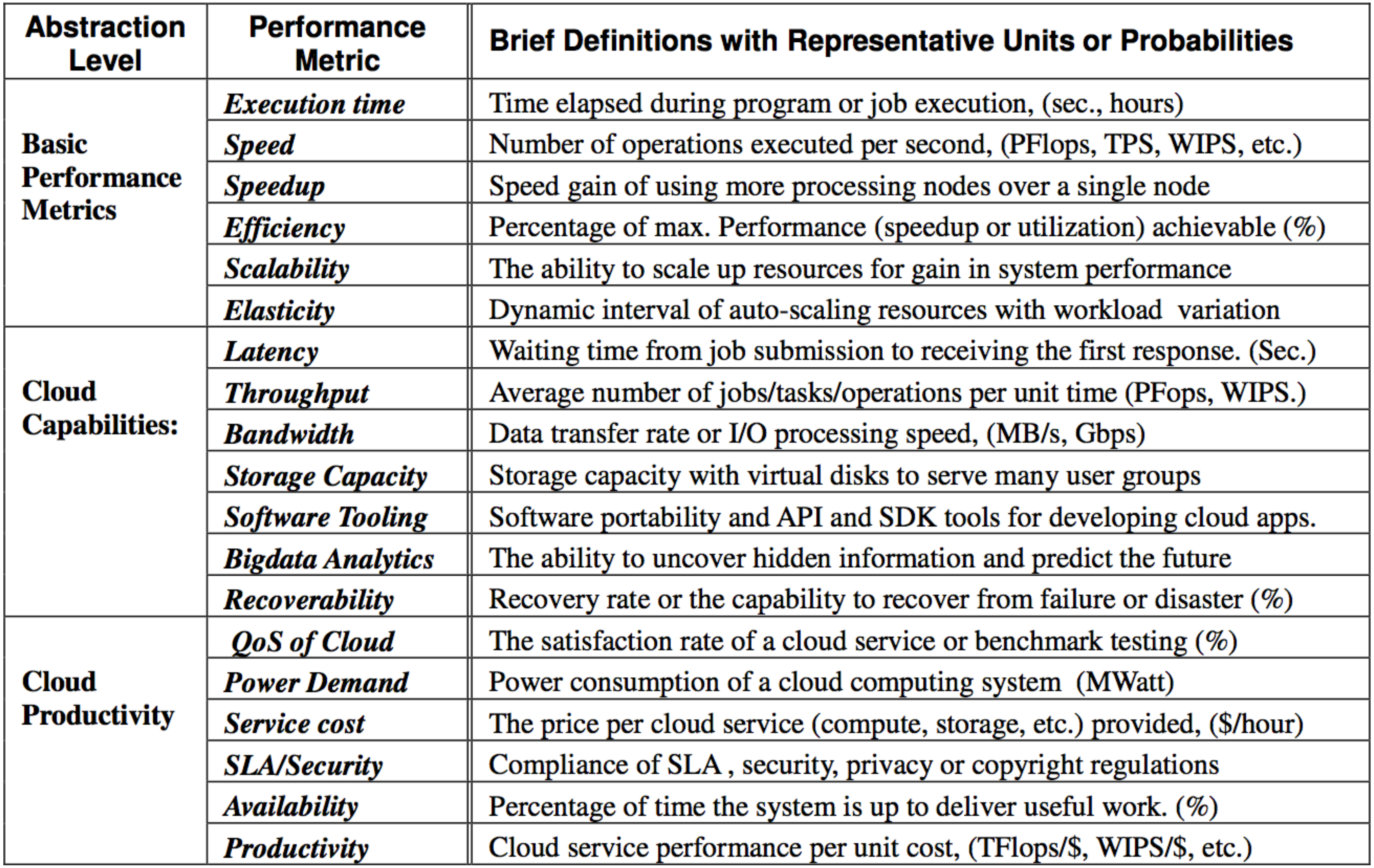
- **Costs**: Total Cost ($), FLOP Cost (cent/FLOP, $/GFLOP), Supported Users on a Fixed Budget, Component Resource Cost ($), Price/Performance Ratio, Cost over a Fixed Time ($/year)


Leveraging Speed and Cost
TCO
Total Cost of Ownership
TRADITIONAL
Adoption of OPEX based Services
Time To Market
Time To Value
Time To Production
Faster rate of cost reduction
Adoption of rapid Dev/Test/Deploy Lifecycle
CLOUD
Faster time To cost reduction
Time

Security Features of Cloud Services

- Data Security
  - *Metrics*: Is SSL Applicable, Communication Latency over SSL, Auditability, Resistance to attacks
- Authentication
  - *Metrics*: Meaning, Sensitivity, Effectiveness, Confidentiality

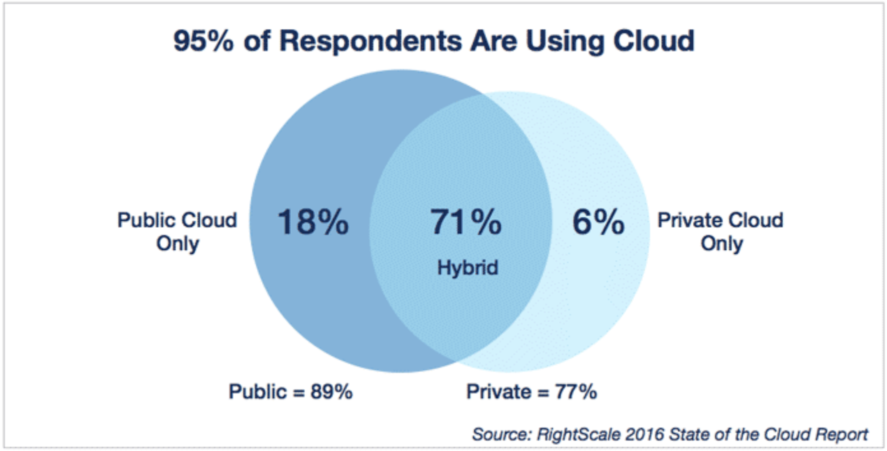
Putting all together



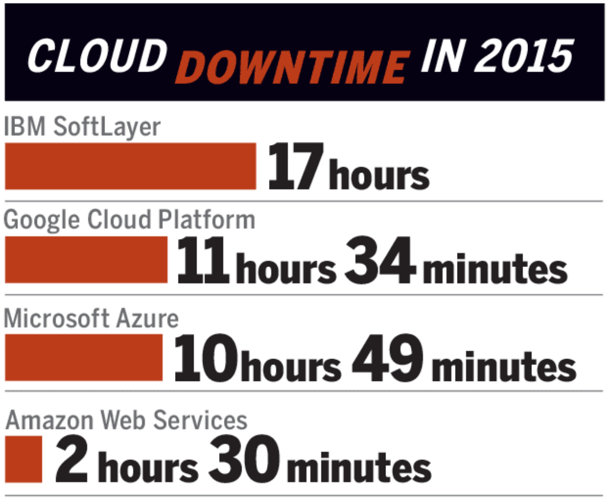
Abstraction Level	Performance Metric	Brief Definitions with Representative Units or Probabilities
Basic Performance Metrics	***Execution time***	Time elapsed during program or job execution, (sec., hours)
	Speed	Number of operations executed per second, (PFlops, TPS, WIPS, etc.)
	Speedup	Speed gain of using more processing nodes over a single node
	Efficiency	Percentage of max. Performance (speedup or utilization) achievable (%)
	Scalability	The ability to scale up resources for gain in system performance
	Elasticity	Dynamic interval of auto-scaling resources with workload variation
Cloud Capabilities:	***Latency***	Waiting time from job submission to receiving the first response. (Sec.)
	Throughput	Average number of jobs/tasks/operations per unit time (PFops, WIPS.)
	Bandwidth	Data transfer rate or I/O processing speed, (MB/s, Gbps)
	Storage Capacity	Storage capacity with virtual disks to serve many user groups
	Software Tooling	Software portability and API and SDK tools for developing cloud apps.
	Bigdata Analytics	The ability to uncover hidden information and predict the future
	Recoverability	Recovery rate or the capability to recover from failure or disaster (%)
Cloud Productivity	***QoS of Cloud***	The satisfaction rate of a cloud service or benchmark testing (%)
	Power Demand	Power consumption of a cloud computing system (MWatt)
	Service cost	The price per cloud service (compute, storage, etc.) provided, ($/hour)
	SLA/Security	Compliance of SLA , security, privacy or copyright regulations
	Availability	Percentage of time the system is up to deliver useful work. (%)
	Productivity	Cloud service performance per unit cost, (TFlops/$, WIPS/$, etc.)

HWANG, ET AL, CLOUD PERFORMANCE MODELING AND BENCHMARK EVALUATION OF ELASTIC SCALING STRATEGIES (TPDS, 2015)

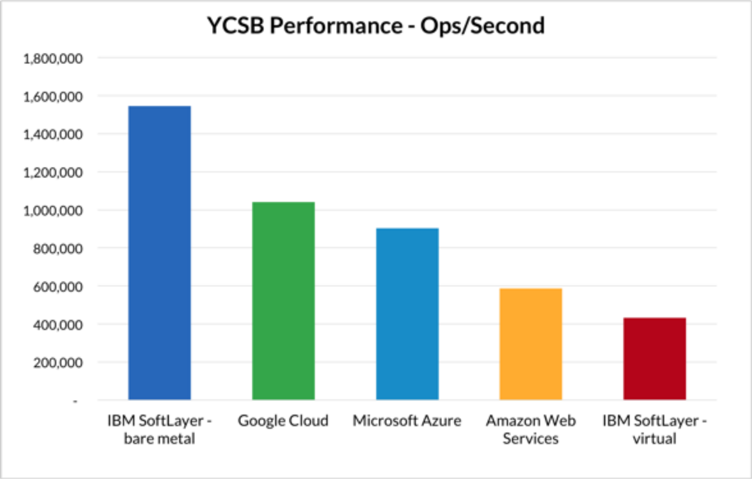
Clouds Performance in Numbers

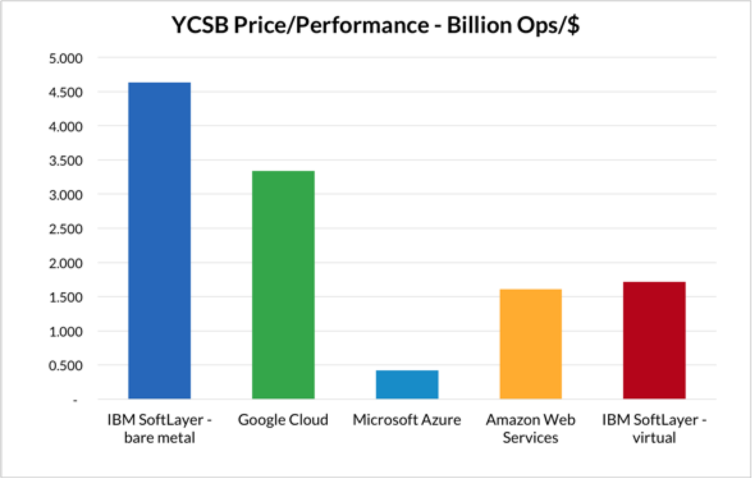


Source: RightScale 2016 State of the Cloud Report



SOURCE: CLOUDHARMONY





https://www.voltdb.com/blog/cloud-benchmark

Test: https://cloudharmony.com

Simulation in CloudSim

- CloudSim provides a generalized and extensible simulation framework that enables modeling, simulation, and experimentation of emerging Cloud computing infrastructure and application services
- Developed CLOUDS Laboratory -> Computer Science and Software Engineering Department of the University of Melbourne
- CloudSim Toolkit 3.0 released at Jan 13, 2012

http://www.cloudbus.org/cloudsim/

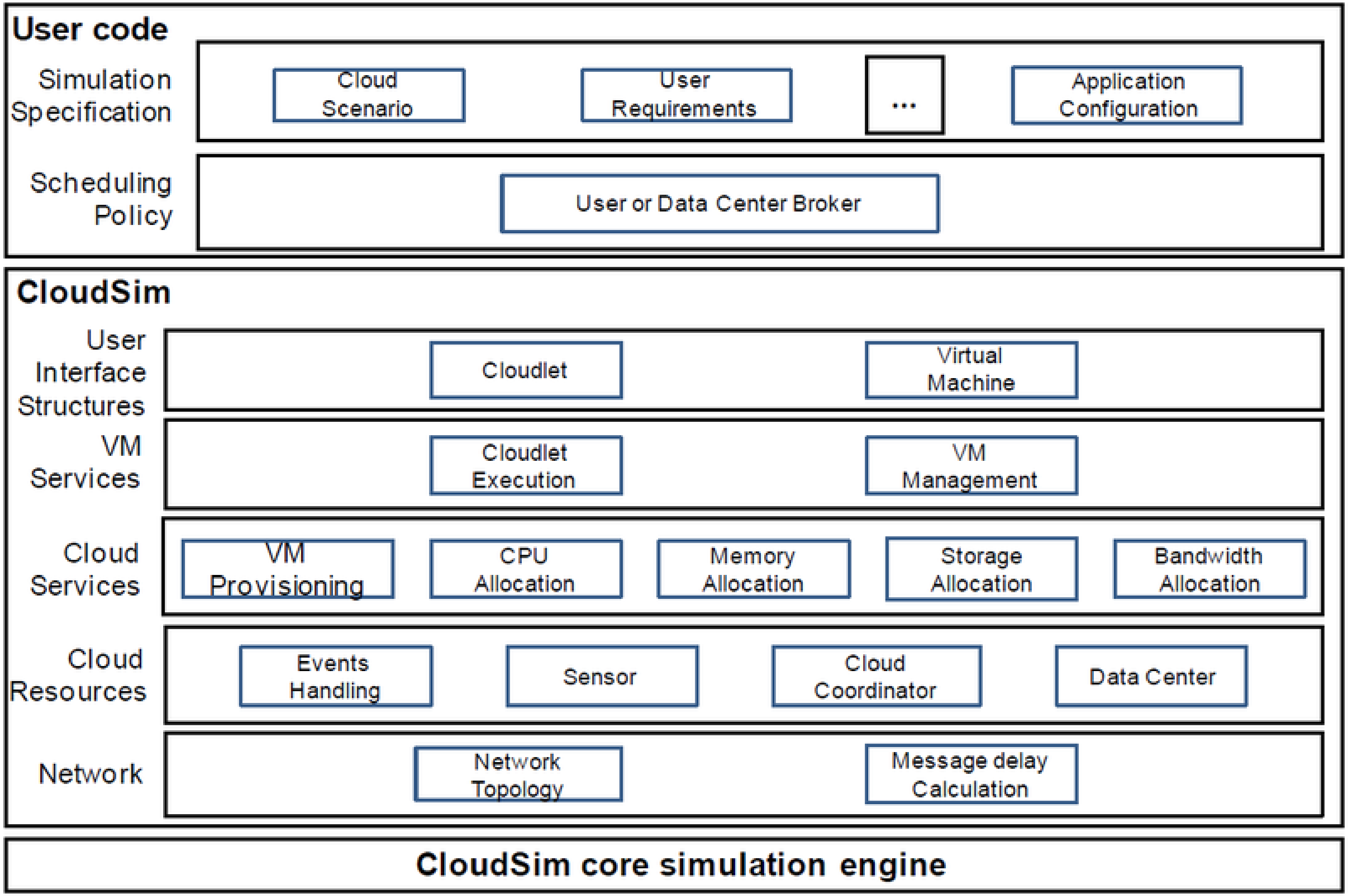
CloudSim

- Support for modeling and simulation of large scale Cloud computing data centers (high an)
- Energy-aware computational resources
- Support for data center network topologies and message-passing application
- Support for dynamic insertion of simulation elements, stop and resume of simulation
- Support for user-defined policies for allocation of hosts to virtual machines and policies for allocation of host resources to virtual machines

Why CloudSim?

- Cloud resource provisioning
- Energy-efficient management of data center resources
- Support for Optimization
- Limitation: no GUI.

CloudSim Architecture



User code

Simulation Specification: Cloud Scenario | User Requirements | ... | Application Configuration

Scheduling Policy: User or Data Center Broker

CloudSim

User Interface Structures: Cloudlet | Virtual Machine

VM Services: Cloudlet Execution | VM Management

Cloud Services: VM Provisioning | CPU Allocation | Memory Allocation | Storage Allocation | Bandwidth Allocation

Cloud Resources: Events Handling | Sensor | Cloud Coordinator | Data Center

Network: Network Topology | Message delay Calculation

CloudSim core simulation engine

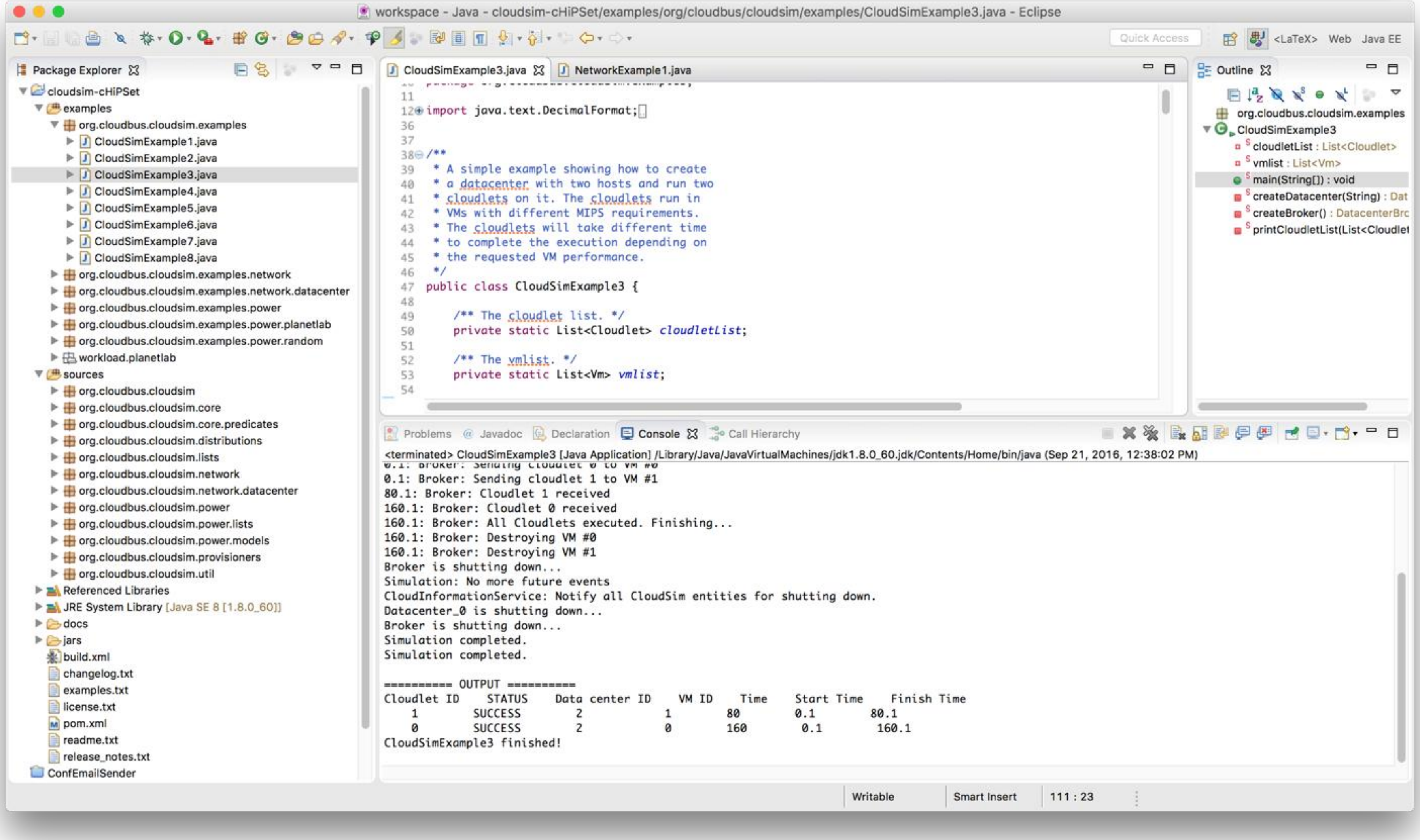
CloudSim - Setting up Development Environment

- Minimal requirements
  - Java Development Kit (already installed Java 1.8)
  - Eclipse IDE (classic – already installed, Eclipse Neon)
- Minimal knowledge
  - basic understanding of how to program in Java
  - basic OOP concept
- To download CloudSim packages use the following link:

  https://code.google.com/archive/p/cloudsim/downloads
- Downloading the common maths file use this link

  http://apache.javapipe.com/commons/math/binaries/

Quick look over running CloudSim Environments



workspace - Java - cloudsim-cHiPSet/examples/org/cloudbus/cloudsim/examples/CloudSimExample3.java - Eclipse
Quick Access
<LaTeX>
Web
Java EE
Package Explorer
cloudsim-cHiPSet
examples
org.cloudbus.cloudsim.examples
CloudSimExample1.java
CloudSimExample2.java
CloudSimExample3.java
CloudSimExample4.java
CloudSimExample5.java
CloudSimExample6.java
CloudSimExample7.java
CloudSimExample8.java
org.cloudbus.cloudsim.examples.network
org.cloudbus.cloudsim.examples.network.datacenter
org.cloudbus.cloudsim.examples.power
org.cloudbus.cloudsim.examples.power.planetlab
org.cloudbus.cloudsim.examples.power.random
workload.planetlab
sources
org.cloudbus.cloudsim
org.cloudbus.cloudsim.core
org.cloudbus.cloudsim.core.predicates
org.cloudbus.cloudsim.distributions
org.cloudbus.cloudsim.lists
org.cloudbus.cloudsim.network
org.cloudbus.cloudsim.network.datacenter
org.cloudbus.cloudsim.power
org.cloudbus.cloudsim.power.lists
org.cloudbus.cloudsim.power.models
org.cloudbus.cloudsim.provisioners
org.cloudbus.cloudsim.util
Referenced Libraries
JRE System Library [Java SE 8 [1.8.0_60]]
docs
jars
build.xml
changelog.txt
examples.txt
license.txt
pom.xml
readme.txt
release_notes.txt
ConfEmailSender
CloudSimExample3.java
NetworkExample1.java
Outline
org.cloudbus.cloudsim.examples
CloudSimExample3
cloudletList : List<Cloudlet>
vmlist : List<Vm>
main(String[]) : void
createDatacenter(String) : Dat
createBroker() : DatacenterBrc
printCloudletList(List<Cloudlet
Problems
Javadoc
Declaration
Console
Call Hierarchy
Writable
Smart Insert
111 : 23

```java
import java.text.DecimalFormat;

/**
 * A simple example showing how to create
 * a datacenter with two hosts and run two
 * cloudlets on it. The cloudlets run in
 * VMs with different MIPS requirements.
 * The cloudlets will take different time
 * to complete the execution depending on
 * the requested VM performance.
 */
public class CloudSimExample3 {

    /** The cloudlet list. */
    private static List<Cloudlet> cloudletList;

    /** The vmlist. */
    private static List<Vm> vmlist;

```

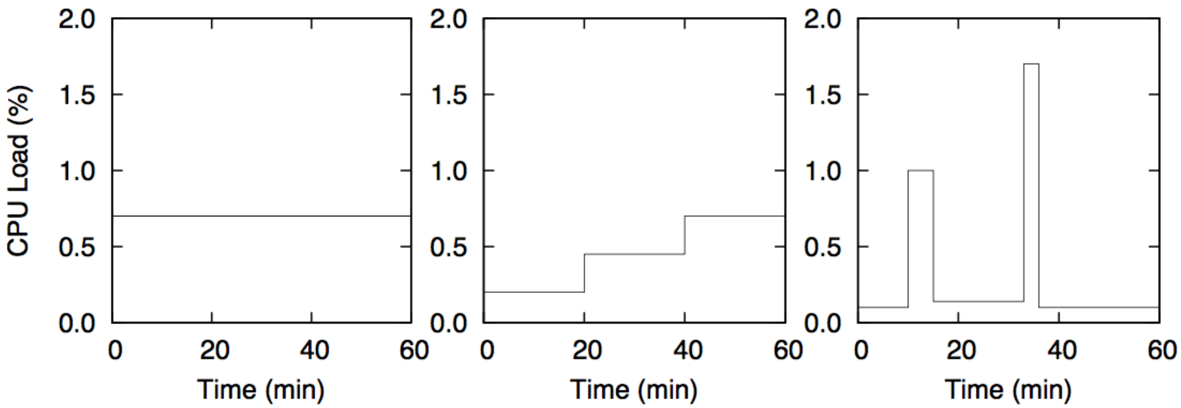
```
<terminated> CloudSimExample3 [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (Sep 21, 2016, 12:38:02 PM)
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
80.1: Broker: Cloudlet 1 received
160.1: Broker: Cloudlet 0 received
160.1: Broker: All Cloudlets executed. Finishing...
160.1: Broker: Destroying VM #0
160.1: Broker: Destroying VM #1
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    1          SUCCESS        2              1      80       0.1         80.1
    0          SUCCESS        2              0      160      0.1         160.1
CloudSimExample3 finished!
```

TODO – Programming in CloudSim

- Create a simulation (in CloudSim) with the following parameters
  - 2 hosts: $Host_1$ and $Host_2$
  - $Host_1$ has m_1 VMs and $Host_2$ has m_2 VMs (heterogeneous) with
    - $m = m_1 + m_2 \leq 10$.
  - Create (by configuring CloudLet) 3 types of workloads: uniform, increasing, bursty with $n \gg m$ tasks.
  - //submit cloudlet list to the broker: broker.submitCloudletList(*cloudletList);*
  - **Advanced (optional):** Schedule each workload on the set of *m* VMs using the following scheduling strategies:
    - *Random* and *Round Robin.*
    - `public class Scheduler extends DatacenterBroker`
  - Measure at least 2 performance metrics.

Workload Generation

- Workload Characterization
  - CPU-Intensive workload;
  - Memory-Intensive workload;
  - I/O-Intensive workload;
  - Mixture of Memory and I/O-Intensive workload.
- Workload Patterns
  - Uniform
  - Increasing
  - Bursty



Performance Metrics (1/4)

- Job Wait Time (*WT*)
  - The time each job waits in the queue before execution
- Job Response Time (*ReT*)
  - The time between the job arrival in Broker, and the receipt of a report from the virtual resource it was executed on.
- Workload Makespan (*MS*)
  - The interval between the time that the first job in the workload arrives, and the time that the execution results of the last job in the workload have been received

$$MS(W) = t_{lc} - t_{fa}$$

- Job Slowdown (*JSD*)
  - the ratio of the actual runtime in the cloud and the runtime in a dedicated environment.

Performance Metrics (2/4)

- Workload Speedup One (SU_1)
  - the ratio between its makespan and the sum of its job runtimes in a dedicated environment.

$$SU_1 = \frac{MS(W)}{\sum_{i \in W} t_{Ri}}$$

- Workload slowdown infinite (SD_∞)
  - represents the slowdown against an infinitely large system

$$SU_\infty = \frac{MS(W)}{max_{i \in W}\{t_{Ri}\}}$$

Performance Metrics (4/4)

- Cost Efficiency (C_{eff})
  - the ratio of the charged and actual cost

$$C_{eff}(W) = \frac{C_c(W)}{C_a(W)}$$

- Utility (U)
  - is a compound metric that rewards low performance overheads and low cost

$$U(W) = \frac{SU_1(W)}{C_a(W)}$$

Performance Metrics (3/4)

- Actual Cost (C_a)
  - the aggregated amount of time that each instance participating in the workload execution has been running for

$$C_a = \sum_{i \in leased\, Vms} (t_{stop}(i) - t_{start}(i))$$

- Charged cost (C_c)
  - follows Amazon's pricing policy for EC2. Amazon charges per hour of use of each leased instance

$$C_c = \sum_{i \in leased\, Vms} \lceil t_{stop}(i) - t_{start}(i) \rceil$$

Time for Questions…

- Can I get all my data from you?
- Is the code I write to customize it portable?
- Can you tell me where my servers are?
- Is the app legally usable from anywhere in the world?
- What kinds of SLA and availability reports do you have?
- How do I dispute my bill, and what proof do you have?
- What privacy guarantees do you have in place?
- What APIs do you offer, how are they supported, and where are the docs?
- Can I keep users on an older version while I train them on the new one?
- Can I back up and restore configurations?

Time for Answers…

Infrastructure transparency (we need to see where data lives after all)

Portability and dependency (a whole new kind of vendor lock-in)

Portfolio management tools (too many Cloud tools to deal with)

Cloud *becomes* the middleman

Social networking (shared apps have shared users)

Security (much easier to do bad things when an account is compromised)

Competitive advantage (Don’t Cloud what makes you special)

Thank you!

Florin POP
florin.pop@cs.pub.ro
http://florinpop.ro