



# **Cloud Computing Hadoop Amazon**

# ?What is Cloud Computing

- Cloud computing is a model for enabling *convenient*, *on-demand network access* to a *shared pool* of *configurable computing resources* (e.g., networks, servers, storage, applications, and services) [Mell\_2009], [Berkely\_2009].
- It can be *rapidly provisioned* and *released* with minimal management effort.
- It provides *high level abstraction* of computation and storage model.
- It has some essential **characteristics**, **service models**, and **deployment models**.

# Essential Characteristics

- ***On-Demand Self Service:***
  - A consumer can unilaterally provision computing capabilities, automatically **without requiring human interaction with each service's provider.**
- ***Heterogeneous Access:***
  - Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous ***thin*** or ***thick*** client platforms.

# Essential Characteristics (cont.)

- ***Resource Pooling:***
  - The provider's computing resources are pooled to serve multiple consumers using a *multi-tenant model*.
  - Different physical and virtual resources dynamically assigned and reassigned according to consumer demand.
- ***Measured Service:***
  - Cloud systems *automatically control* and *optimize* resources used by leveraging a metering capability at some level of abstraction appropriate to the type of service.
  - *It will provide analyzable and predictable computing platform.*

# Service Models

- ***Cloud Software as a Service (SaaS):***
  - 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 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,...
  - ***Examples: Caspio, Google Apps, Salesforce, Nivio, Learn.com.***

## Service Models (cont.)

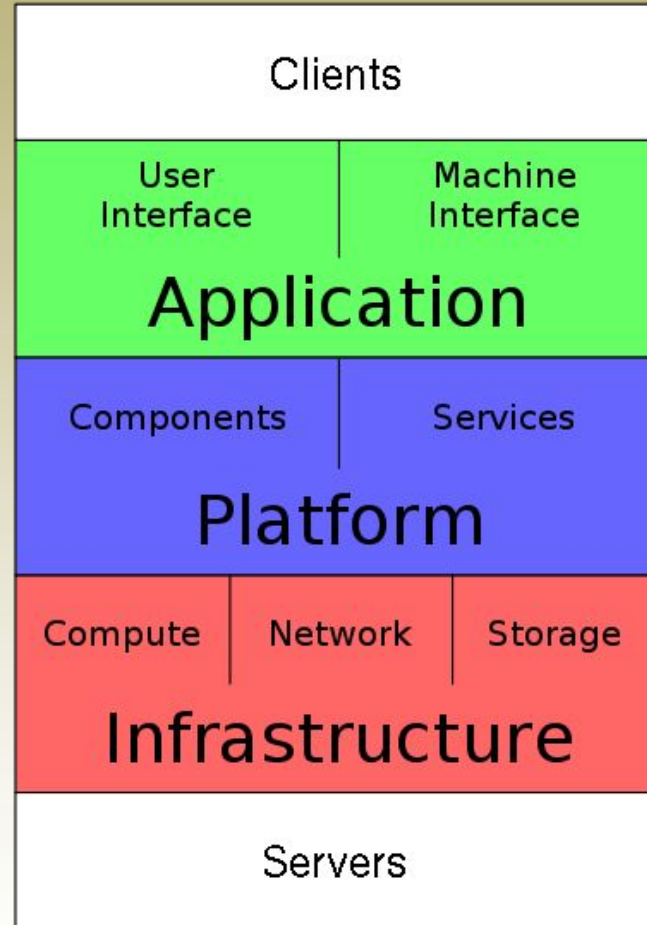
- ***Cloud Platform as a Service (PaaS):***
  - The capability provided to the consumer is to deploy onto the cloud infrastructure *consumer-created* or *acquired applications* created using *programming languages and tools* supported by the provider.
  - The consumer does not manage or control the underlying cloud infrastructure.
  - Consumer has control over the deployed applications and possibly application hosting environment configurations.
  - ***Examples: Windows Azure, Google App.***

# Service Models (cont.)

- ***Cloud Infrastructure as a Service (IaaS):***
  - The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources.
  - 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).
  - ***Examples: Amazon EC2, GoGrid, iland, Rackspace Cloud Servers, ReliaCloud.***



# Service Models (cont.)



Cloud Computing Stack

Service Model at a glance: Picture From [http://en.wikipedia.org/wiki/File:Cloud\\_Computing\\_Stack.svg](http://en.wikipedia.org/wiki/File:Cloud_Computing_Stack.svg)





# Deployment Models



## ***:Private Cloud .***

The cloud 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

## ***:Community Cloud .***

The cloud infrastructure is shared by several organizations and .supports a specific community that has **shared concerns**

It may be managed by the organizations or a third party and may exist on premise or off premise

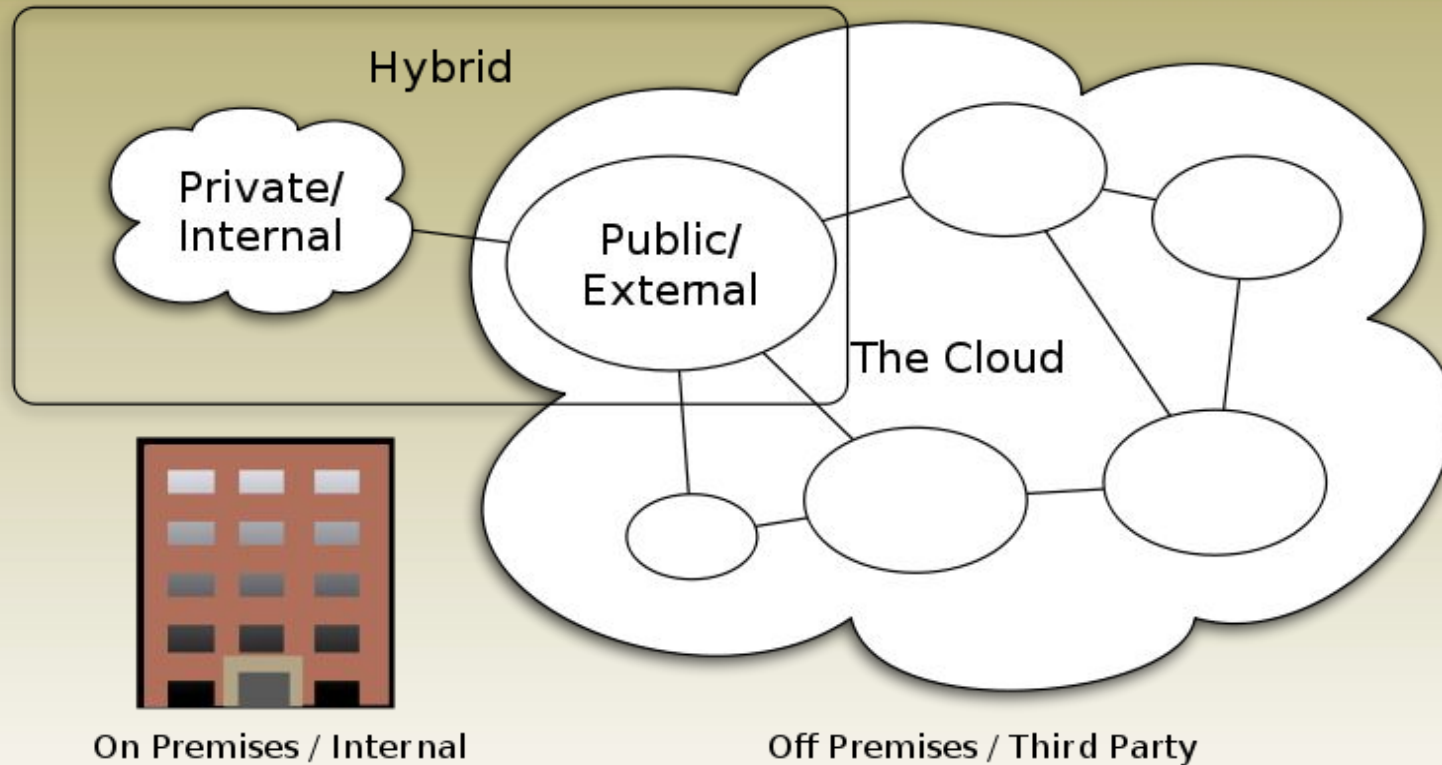


## ***:Public Cloud .***

The cloud infrastructure is made available to the general public or a large industry group and it is owned by an organization selling cloud services –

## ***:Hybrid cloud .***

The cloud infrastructure is a composition of two or more clouds –  
(private, community, or public)



## Cloud Computing Types

CC-BY-SA 3.0 by Sam Johnston

## Advantages of Cloud Computing

- Cloud computing do not need high quality equipment for user, and it is very easy to use.
- Provides dependable and secure data storage center.
- Reduce run time and response time.
- Cloud is a large resource pool that you can buy on-demand service.
- Scale of cloud can extend dynamically providing nearly infinite possibility for users to use internet.



Infrastructure as a Service  
(IaaS)  
Amazon EC2

# ? What is Infrastructure as a Service

- A category of cloud services which provides capability to provision processing, storage, intra-cloud network connectivity services, and other fundamental computing resources of the cloud infrastructure.

Source- [ITU –Cloud Focus Group]

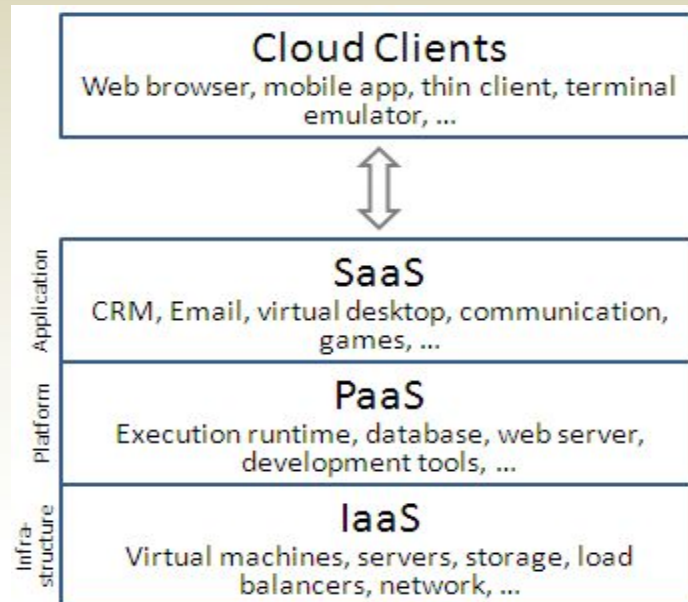


Diagram Source: Wikipedia



# Highlights of IaaS

- On demand computing resources
  - Eliminate the need of far ahead planning
- No up-front commitment
  - Start small and grow as required
  - No contract, Only credit card!
- Pay for what you use
- No maintenance
- Measured service
- Scalability
- Reliability

# ? What is EC2

- Amazon Elastic Compute Cloud (EC2) is a web service that provides resizable computing capacity that one uses to build and host different software systems.
- Designed to make web-scale computing easier for developers.
- A user can create, launch, and terminate server instances as needed, paying by the hour for active servers, hence the term "elastic".
  - Provides scalable, pay as-you-go compute capacity
  - Elastic - scales in both direction



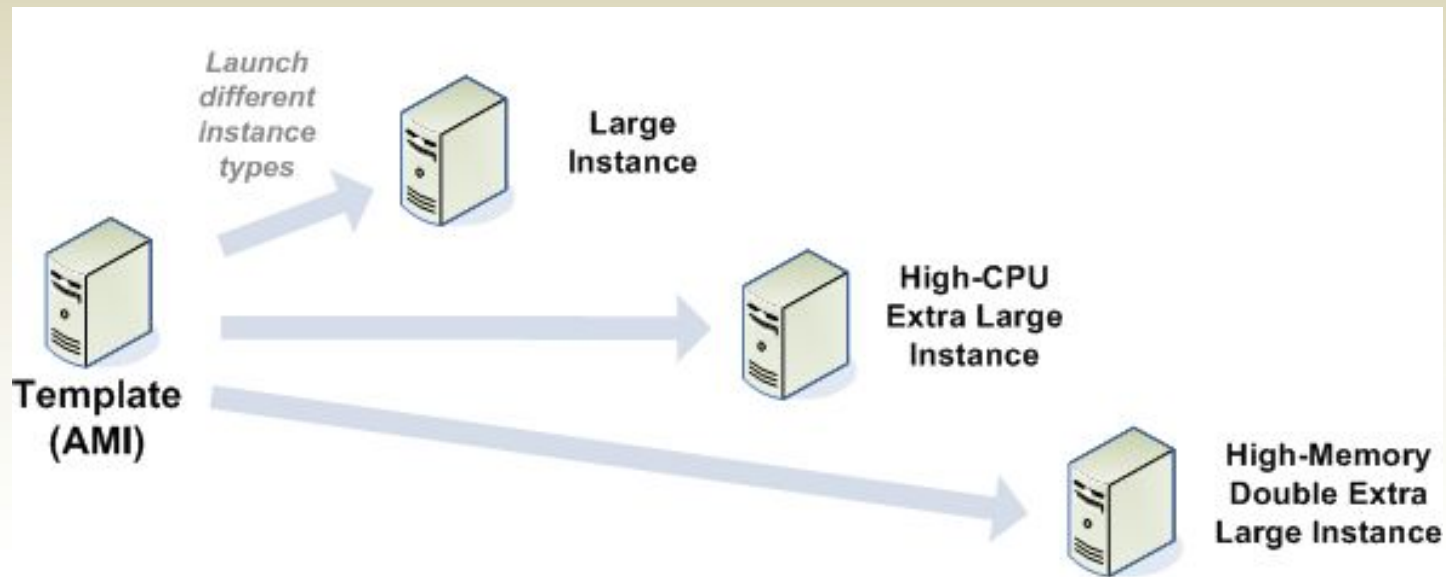
# EC2 Infrastructure Concepts

# EC2 Concepts

- AMI & Instance
- Region & Zones
- Storage
- Networking and Security
- Monitoring
- Auto Scaling
- Load Balancer

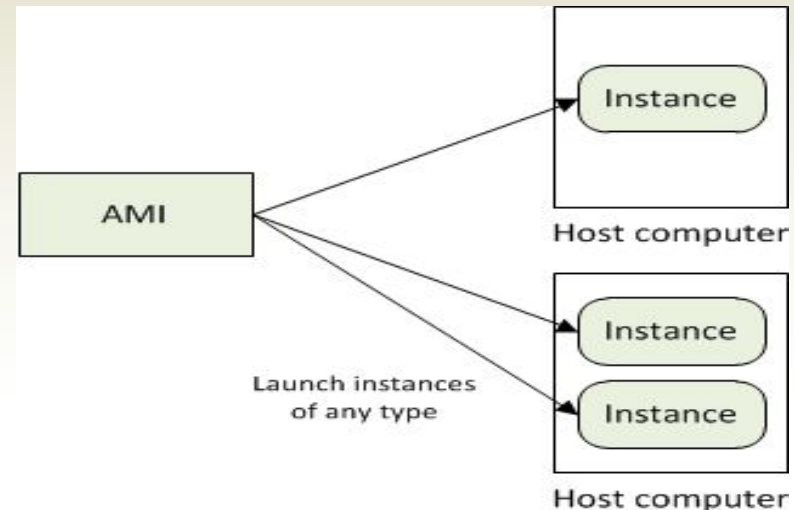
# Amazon Machine Images (AMI)

- Is an immutable representation of a set of disks that contain an operating system, user applications and/or data.
- From an AMI, one can launch multiple instances, which are running copies of the AMI.



# AMI and Instance

- Amazon Machine Image (AMI) is a template for software configuration (Operating System, Application Server, and Applications)
- Instance is a AMI running on virtual servers in the cloud
- Each *instance type* offers different compute and memory facilities



Type	CPU	Memory	Local Storage	Platform	I/O	Name
Small	1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)	1.7 GB	160 GB instance storage (150 GB plus 10 GB root partition)	32-bit	Moderate	m1.small
Large	4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each)	7.5 GB	850 GB instance storage (2 x 420 GB plus 10 GB root partition)	64-bit	High	m1.large
Extra Large	8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each)	15 GB	1690 GB instance storage (4 x 420 GB plus 10 GB root partition)	64-bit	High	m1.xlarge
Micro	Up to 2 EC2 Compute Units (for short periodic bursts)	613 MB	None (use Amazon EBS volumes for storage)	32-bit or 64-bit	Low	t1.micro
High-CPU Medium	5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each)	1.7 GB	350 GB instance storage (340 GB plus 10 GB root partition)	32-bit	Moderate	c1.medium

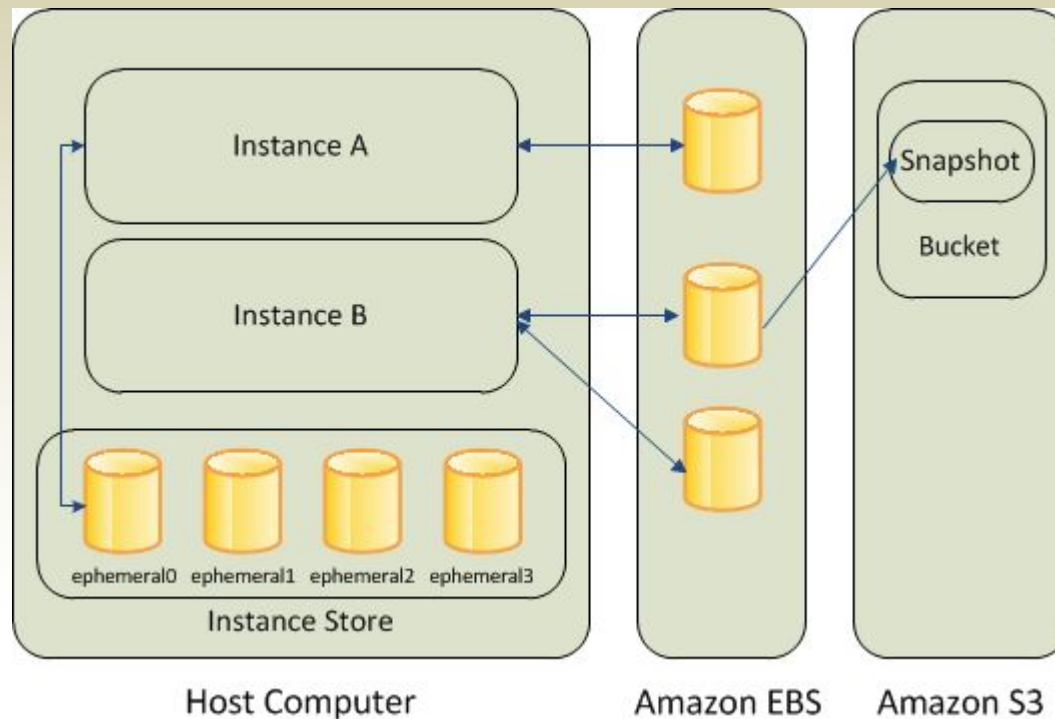


# Region and Zones

- Amazon have data centers in different region across the globe
- An instance can be launched in different regions depending on the need.
  - Closer to specific customer
  - To meet legal or other requirements
- Each region has set of zones
  - Zones are isolated from failure in other zones
  - Inexpensive, low latency connectivity between zones in same region

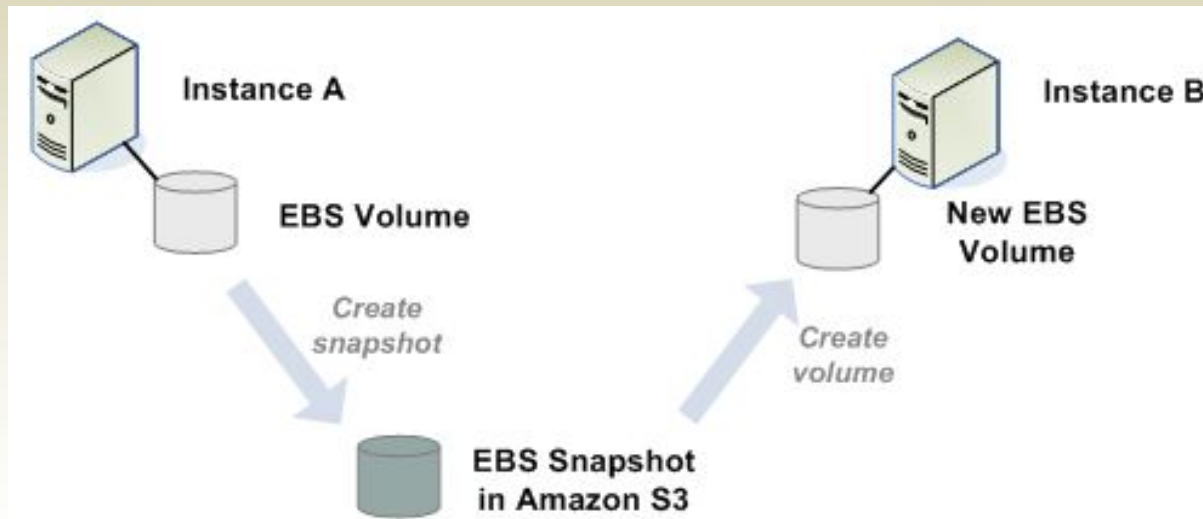
# Storage

- Amazon EC2 provides three type of storage option
  - Amazon EBS
  - Amazon S3
  - Instance Storage



# Elastic Block Store(EBS) volume

- An EBS volume is a read/write disk that can be created by an AMI and mounted by an instance.
- Volumes are suited for applications that require a database, a file system, or access to raw block-level storage.



# Amazon S3

- S3 = Simple storage Service
- A SOA – Service Oriented Architecture which provides online storage using web services.
- Allows read, write and delete permissions on objects.
- Uses REST and SOAP protocols for messaging.

# Amazon SimpleDB

- Amazon SimpleDB is a highly available, flexible, and scalable non-relational data store that offloads the work of database administration.
- Creates and manages multiple geographically distributed replicas of your data automatically to enable high availability and data durability.
- The service charges you only for the resources actually consumed in storing your data and serving your requests.

# Networking and Security

- Instances can be launched on one of the two platforms
  - EC2-Classic
  - EC2-VPC
- Each instance launched is assigned two addresses a private address and a public IP address.
  - A replacement instance has a different public IP address.
- Instance IP address is dynamic.
  - new IP address is assigned every time instance is launched
- Amazon EC2 offers Elastic IP addresses (static IP addresses) for dynamic cloud computing.
  - Remap the Elastic IP to new instance to mask failure
  - Separate pool for EC2-Classic and VPC
- Security Groups to access control to instance



# Monitoring, Auto Scaling, and Load Balancing

- Monitor statistics of instances and EBS
  - CloudWatch
- Automatically scales amazon EC2 capacity up and down based on rules
  - Add and remove compute resource based on demand
  - Suitable for businesses experiencing variability in usage
- Distribute incoming traffic across multiple instances
  - Elastic Load Balancing



# How to access EC2

- AWS Console
  - <http://console.aws.amazon.com>
- Command Line Tools
- Programmatic Interface
  - EC2 APIs
  - AWS SDK

Amazon S3

Amazon EC2

Amazon VPC

Amazon Elastic MapReduce

Amazon CloudFront

Amazon RDS

Amazon SNS

Navigation

Region: US East

> EC2 Dashboard

INSTANCES

> Instances

> Spot Requests

IMAGES

> AMIs

> Bundle Tasks

ELASTIC BLOCK STORE

> Volumes

> Snapshots

NETWORKING & SECURITY

> Elastic IPs

> Security Groups

> Placement Groups

> Load Balancers

> Key Pairs

My Instances

Launch Instance

Instance Actions

Reserved Instances

Show/Hide

Refresh

Help

Viewing:

All Instances

All Instance Types

	Name	Instance	Type	Status	Lifecycle	Public DNS
<input checked="" type="checkbox"/>	Web Server	i-841948e9	m1.small	running	normal	ec2-67-202-15-66.compute-1.

1 EC2 Instance selected

EC2 Instance: i-841948e9

Description

Monitoring

Tags

AMI ID:	ami-08728661	Zone:	us-east-1b
Security Groups:	80_22_open	Type:	m1.small
Status:	running	Owner:	043708602122
VPC ID:	-	Subnet ID:	-
Virtualization:	paravirtual	Placement Group:	
Reservation:	r-7de68517	RAM Disk ID:	-
Platform:	-	Key Pair Name:	GSG_Keypair
Kernel ID:	aki-407d9529	Monitoring:	basic
AMI Launch Index:	0	Elastic IP:	-
Root Device:	/dev/sda1	Root Device Type:	ebs

Region:	US East (Virginia) ▼	
	Linux/UNIX Usage	Windows Usage
<b>Standard On-Demand Instances</b>		
Small (Default)	\$0.085 per hour	\$0.12 per hour
Large	\$0.34 per hour	\$0.48 per hour
Extra Large	\$0.68 per hour	\$0.96 per hour
<b>Micro On-Demand Instances</b>		
Micro	\$0.02 per hour	\$0.03 per hour
<b>Hi-Memory On-Demand Instances</b>		
Extra Large	\$0.50 per hour	\$0.62 per hour
Double Extra Large	\$1.00 per hour	\$1.24 per hour
Quadruple Extra Large	\$2.00 per hour	\$2.48 per hour
<b>Hi-CPU On-Demand Instances</b>		
Medium	\$0.17 per hour	\$0.29 per hour
Extra Large	\$0.68 per hour	\$1.16 per hour
<b>Cluster Compute Instances</b>		
Quadruple Extra Large	\$1.60 per hour	N/A*
<b>Cluster GPU Instances</b>		
Quadruple Extra Large	\$2.10 per hour	N/A*
* Windows® is not currently available for Cluster Compute or Cluster GPU Instances		

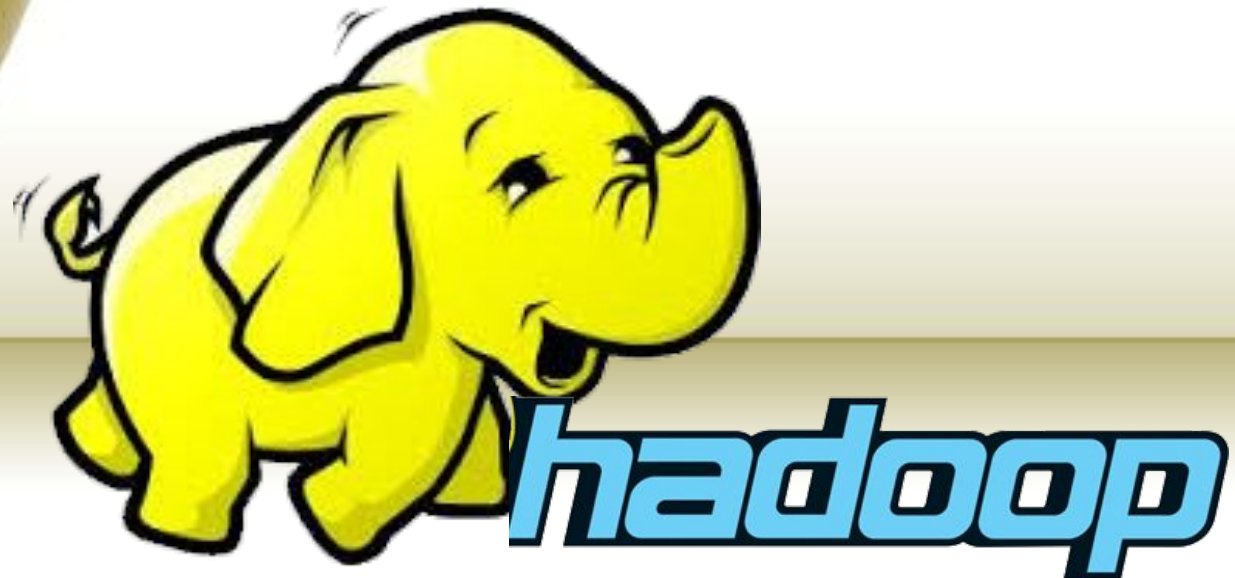
Pricing is per instance-hour consumed for each instance, from the time an instance is launched until it is terminated. Each partial instance-hour consumed will be billed as a full hour.

# References

- Mobile cloud computing: Big Picture by M. Reza Rahimi
- <http://aws.amazon.com/ec2>, <http://docs.aws.amazon.com>
- Amazon Elastic Compute Cloud – User Guide, API Version 2011-02-28.
- Above the Clouds: A Berkeley View of Cloud Computing - Michael Armbrust et.al 2009
- International telecommunication union – Focus Group Cloud Technical Report



# Hadoop, a distributed framework for Big Data

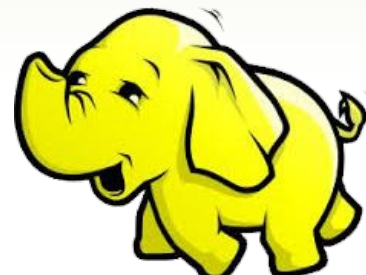






# Introduction

- 1. Introduction: Hadoop's history and advantages**
- 2. Architecture in detail**
- 3. Hadoop in industry**



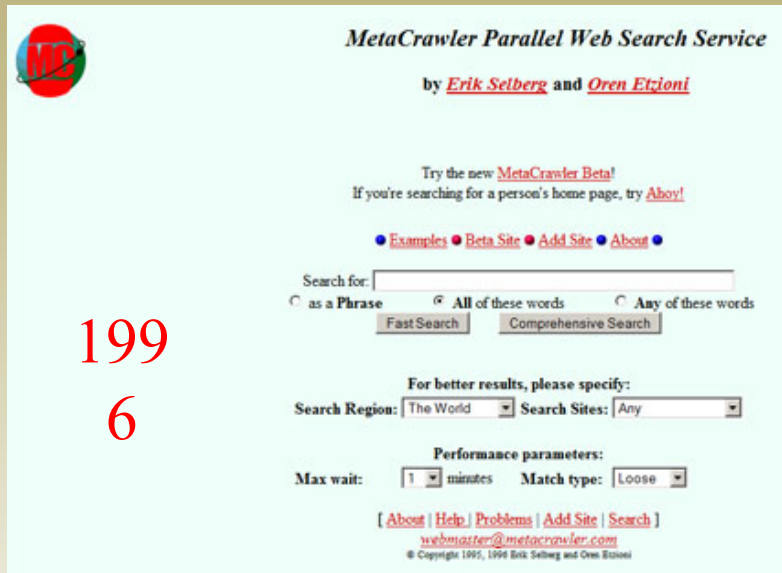
# ?What is Hadoop

- Apache top level project, open-source implementation of frameworks for reliable, scalable, distributed computing and data storage.
- It is a flexible and highly-available architecture for large scale computation and data processing on a network of commodity hardware.
- Designed to answer the question: **“How to process big data with reasonable cost and time?”**





# Search engines in 1990s



**MetaCrawler Parallel Web Search Service**  
by [Erik Selberg](#) and [Oren Etzioni](#)

Try the new [MetaCrawler Beta!](#)  
If you're searching for a person's home page, try [Abov!](#)

● [Examples](#) ● [Beta Site](#) ● [Add Site](#) ● [About](#) ●

Search for:   
☐ as a Phrase ☒ All of these words ☐ Any of these words

For better results, please specify:  
 Search Region:  Search Sites:

Performance parameters:  
 Max wait:  minutes Match type:

[ [About](#) | [Help](#) | [Problems](#) | [Add Site](#) | [Search](#) ]  
[webmaster@metacrawler.com](mailto:webmaster@metacrawler.com)  
 © Copyright 1995, 1996 Erik Selberg and Oren Etzioni

199  
6



**excite** search reviews city.net new live reference?

excite home maps news people finder

**Excite Search:** twice the power of the competition.  
 What:    
 Where:

**Excite Reviews:** site reviews by the web's best editorial team.

• <a href="#">Arts</a>	• <a href="#">Entertainment</a>	• <a href="#">Money</a>	• <a href="#">Regional</a>
• <a href="#">Business</a>	• <a href="#">Health</a>	• <a href="#">News &amp; Reference</a>	• <a href="#">Science</a>
• <a href="#">Computing</a>	• <a href="#">Hobbies</a>	• <a href="#">Personal Pages</a>	• <a href="#">Shopping</a>
• <a href="#">Education</a>	• <a href="#">Life &amp; Style</a>	• <a href="#">Politics &amp; Law</a>	• <a href="#">Sports</a>

199  
6



**HELP** WIRED NEWS HOTWIRED WIRED MAGAZINE SUCK.COM

**THE WRED Search Center**

look for:

for more options use [SuperSearch](#)

Date:   
 Content:

Include media type:  
☐ image ☐ audio ☐ video ☐ Shockwaves

Return Results:

[CITIZEN FORUM](#)

Search: The Web  
 Usenet  
 Top News Sites  
 Classifieds  
 Domain Names  
 Stocks  
 Discussion Groups  
 ShareWare

Find:  
 Businesses  
 People  
 Email Addresses

**Sandbox** WIRED Entertainment  
 Shop WIRED  
 Holiday Gift Guide

**SOMETHING HAS SURVIVED.**  
 Find more here

Log On  
 Cybertron  
 Outpost  
 Microsoft®  
 Expedia™ Travel  
 ON SALE

199  
7



**LYCOS** It's amazing where Go Get It will get you.

Find:

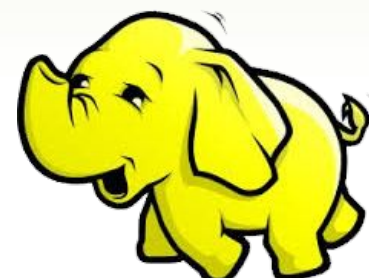
[Enhance your search.](#)

[New Search](#) • [Top News](#) • [Sites by Subject](#) • [Top 5% Sites](#) • [City Guide](#) • [Pictures & Sounds](#)  
[PeopleFind](#) • [Point Review](#) • [Road Maps](#) • [Software](#) • [About Lycos](#) • [Club Lycos](#) • [Help](#)

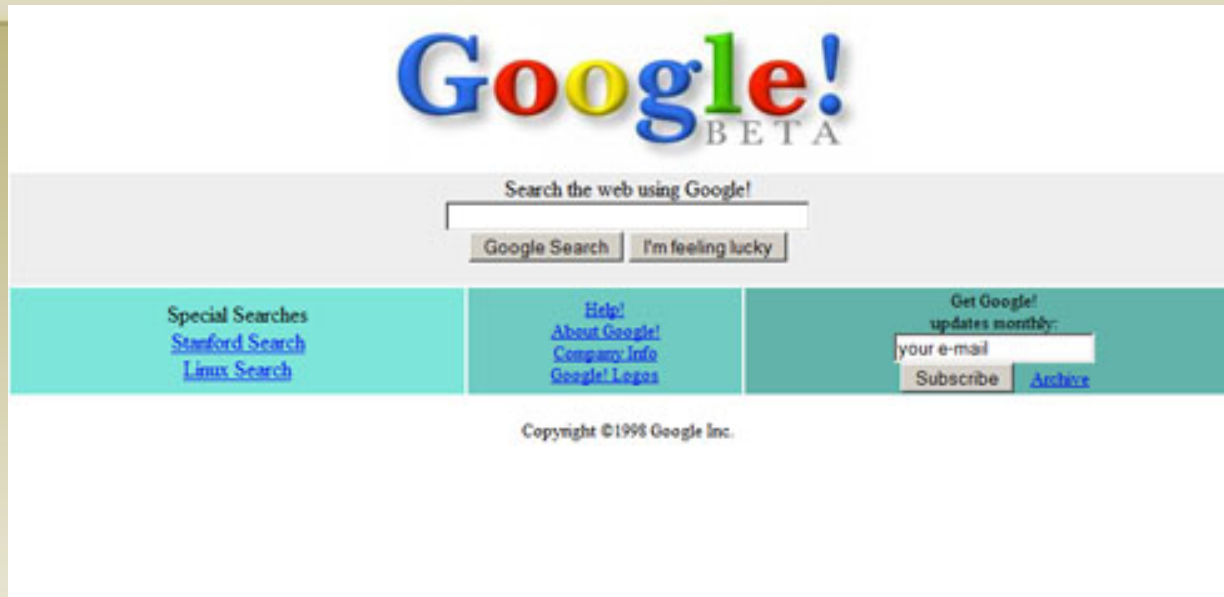
[Add Your Site to Lycos](#)

Copyright © 1996 Lycos™, Inc. All Rights Reserved.  
 Lycos is a trademark of Carnegie Mellon University.  
[Questions & Comments](#)

199  
6



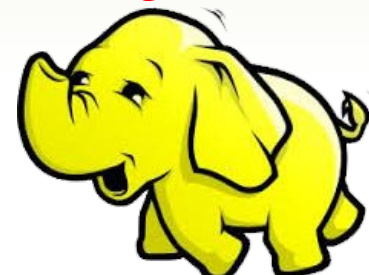
# Google search engines



199  
8



201  
3



# Hadoop's Developers

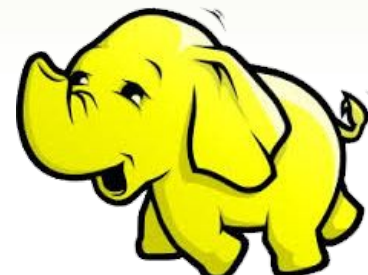


**2005:** Doug Cutting and Michael J. Cafarella developed Hadoop to support distribution for the Nutch search engine project.



The project was funded by Yahoo.

**2006:** Yahoo gave the project to Apache Software Foundation.



# Google Origins

2003

## The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung  
Google\*



2004

## MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat  
jeff@google.com, sanjay@google.com

Google, Inc.



2006

## Bigtable: A Distributed Storage System for Structured Data

Fuy Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach  
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber  
{fuy,jeff,sanjay,wilson,dean,hsieh,tushar,shar,gruber}@google.com

Google, Inc.

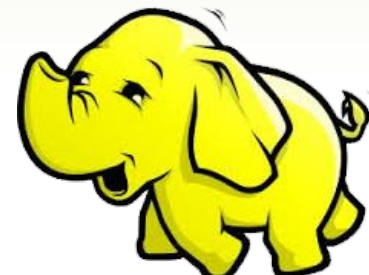
### Abstract

Bigtable is a distributed storage system for managing  
structured data that is designed to scale to a very large  
petabytes of data across thousands of commodity  
servers. Many projects at Google store data in Bigtable,  
including web indexing, Google Earth, and Google Fi-  
re. These applications place very different demands  
on Bigtable, both in terms of data size (from URLs to  
maps for satellite imagery) and latency requirements.

Bigtable achieves scalability and high performance, but Big  
provides a different interface than such systems. Big  
does not support a full relational data model; instead,  
it provides clients with a simple data model that sup-  
ports dynamic control over data layout and format, al-  
lows clients to reason about the locality properties of  
data represented in the underlying storage. Data is  
indexed using row and column names that can be arbitrary  
strings. Bigtable also treats data as uninterpreted str-

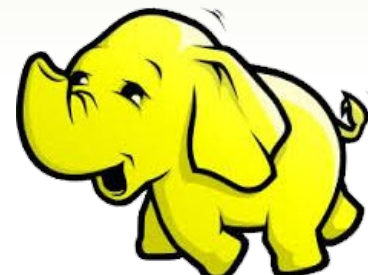


A P A C H E  
**HBASE**



# Some Hadoop Milestones

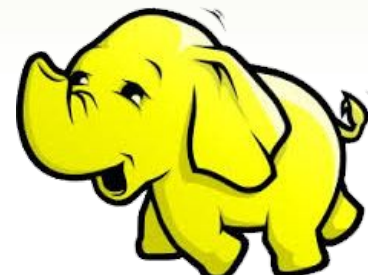
- **2008 - Hadoop Wins Terabyte Sort Benchmark** (sorted 1 terabyte of data in 209 seconds, compared to previous record of 297 seconds)
- 2009 - Avro and Chukwa became new members of Hadoop Framework family
- 2010 - Hadoop's Hbase, Hive and Pig subprojects completed, adding more computational power to Hadoop framework
- **2011 - ZooKeeper Completed**
- **2013 - Hadoop 1.1.2 and Hadoop 2.0.3 alpha.**
  - Ambari, Cassandra, Mahout have been added



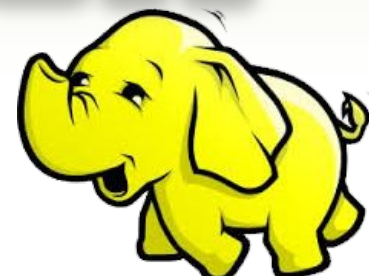
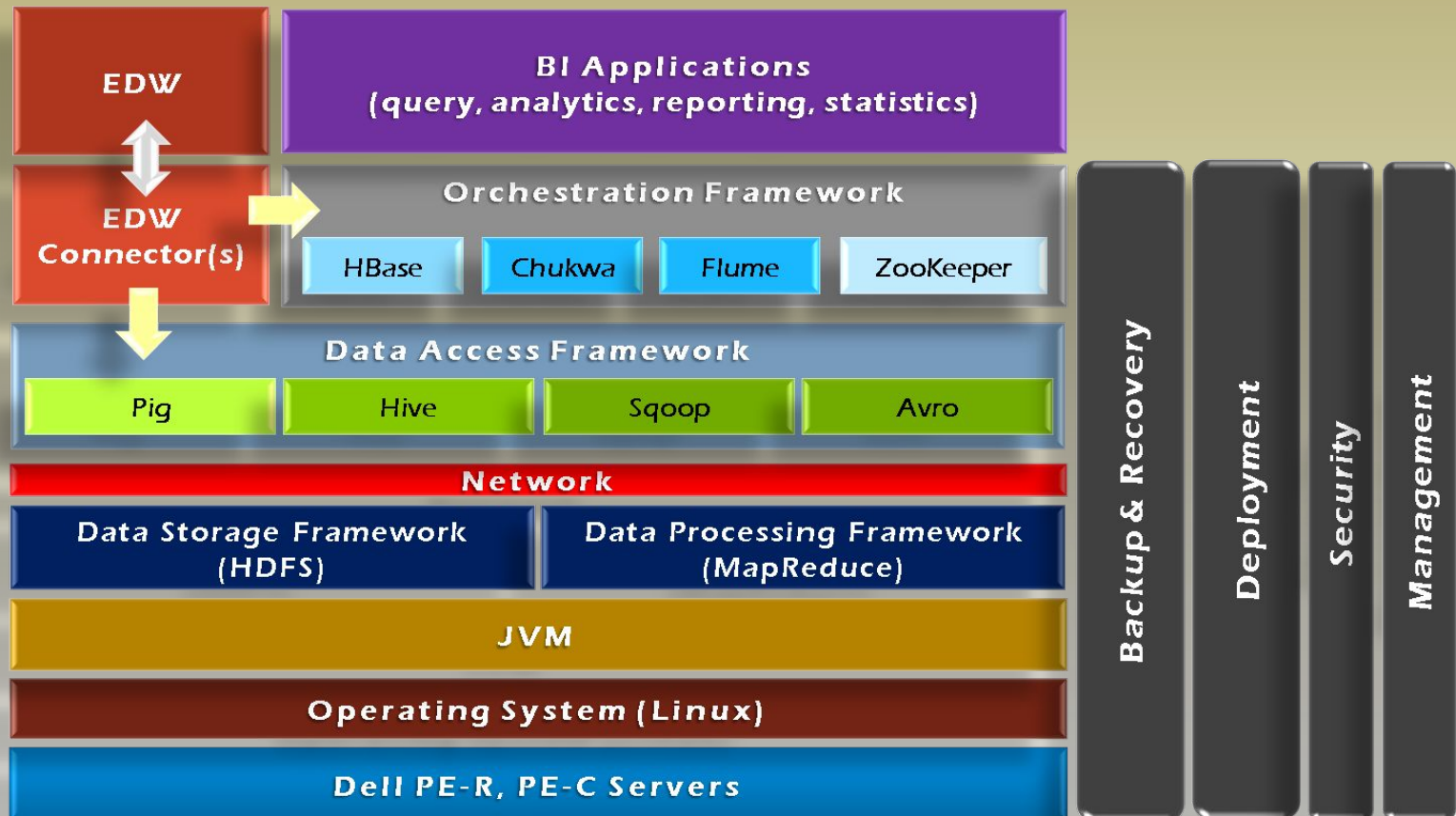


# ?What is Hadoop

- An open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license.
- Abstract and facilitate the storage and processing of large and/or rapidly growing data sets
  - Structured and non-structured data
  - Simple programming models
- High scalability and availability
- Use commodity (cheap!) hardware with little redundancy
- Fault-tolerance
- Move computation rather than data



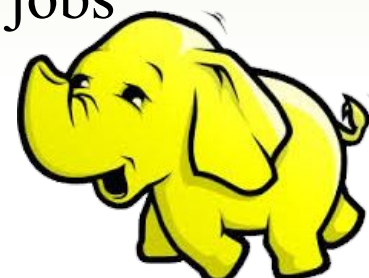
# Hadoop Framework Tools



# Hadoop MapReduce Engine

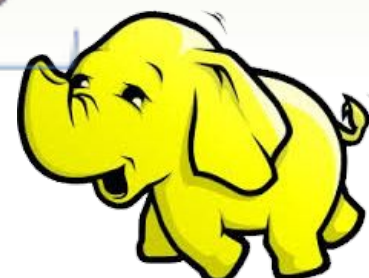
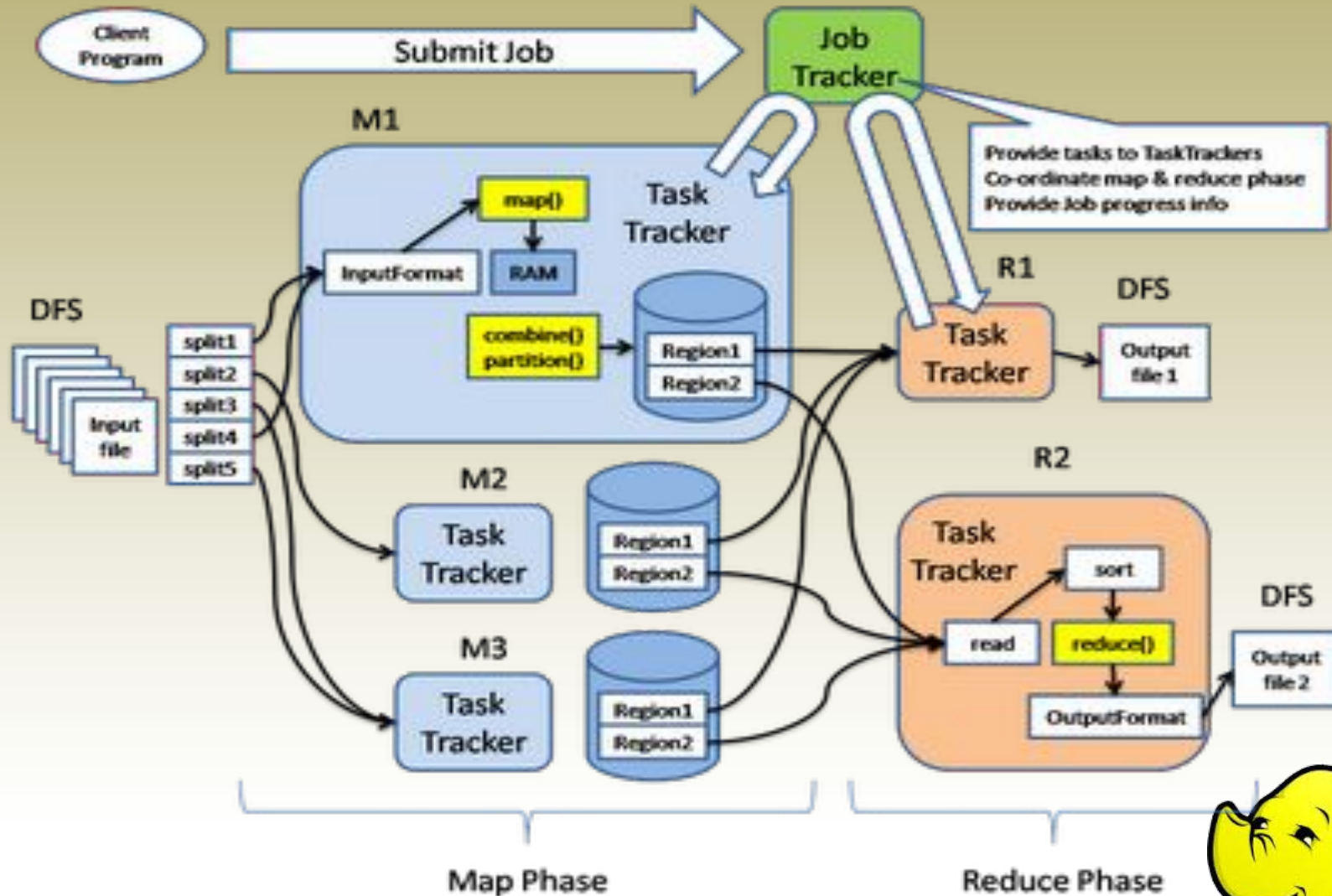
## A MapReduce Process (org.apache.hadoop.mapred)

- JobClient
  - Submit job
- JobTracker
  - Manage and schedule job, split job into tasks;
  - Splits up data into smaller tasks(“Map”) and sends it to the TaskTracker process in each node
- TaskTracker
  - Start and monitor the task execution;
  - reports back to the JobTracker node and reports on job progress, sends data (“Reduce”) or requests new jobs
- Child
  - The process that really executes the task





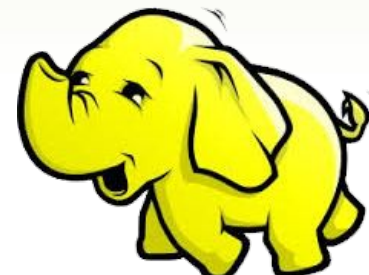
# Hadoop's Architecture: MapReduce Engine



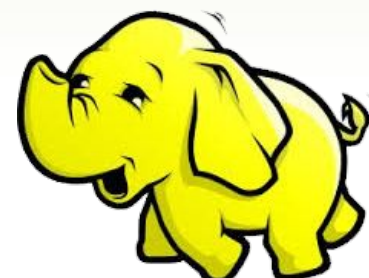
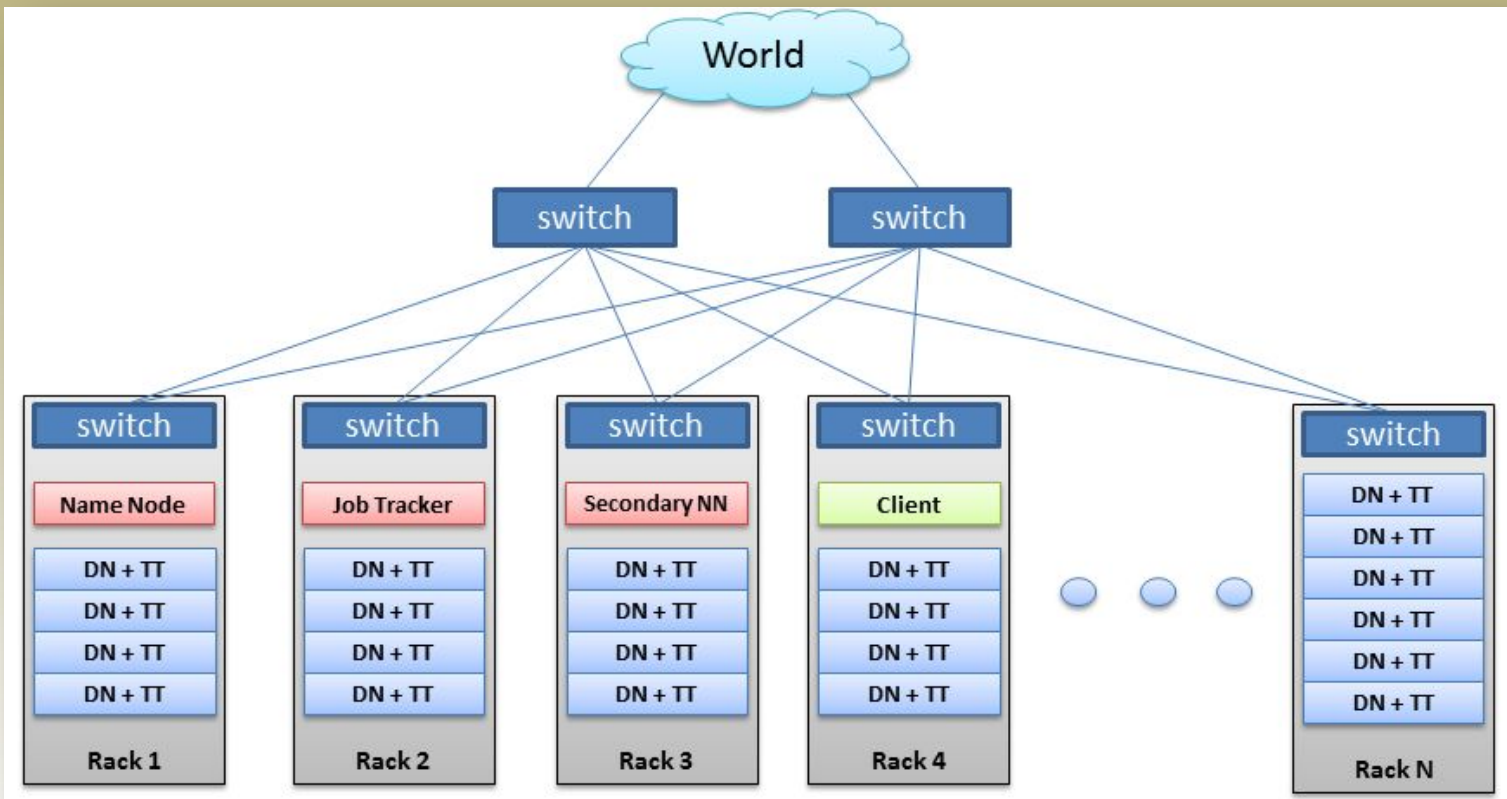


# Hadoop's MapReduce Architecture

- Distributed, with some centralization
- Main nodes of cluster are where most of the computational power and storage of the system lies
- Main nodes run TaskTracker to accept and reply to MapReduce tasks, Main Nodes run DataNode to store needed blocks closely as possible
- Central control node runs NameNode to keep track of HDFS directories & files, and JobTracker to dispatch compute tasks to TaskTracker
- Written in Java, also supports Python and Ruby

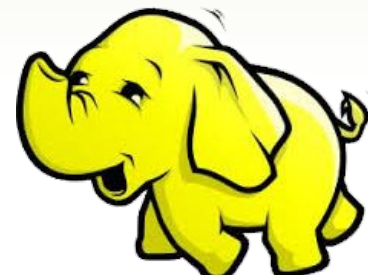


# Hadoop's Architecture



# Hadoop Distributed FileSystem

- Tailored to needs of MapReduce
- Targeted towards many reads of filestreams
  - Writes are more costly
- Open Data Format
  - Flexible Schema
  - Queryable Database
- Fault Tolerance
  - High degree of data replication (3x by default)
  - No need for RAID on normal nodes
- Large blocksize (64MB)
- Location awareness of DataNodes in network



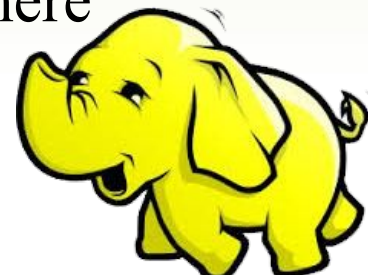
# HDFS

## NameNode:

- Stores metadata for the files, like the directory structure of a typical FS.
- The server holding the NameNode instance is quite crucial, as there is only one.
- Transaction log for file deletes/adds, etc. Does not use transactions for whole blocks or file-streams, only metadata.
- Handles creation of more replica blocks when necessary after a DataNode failure

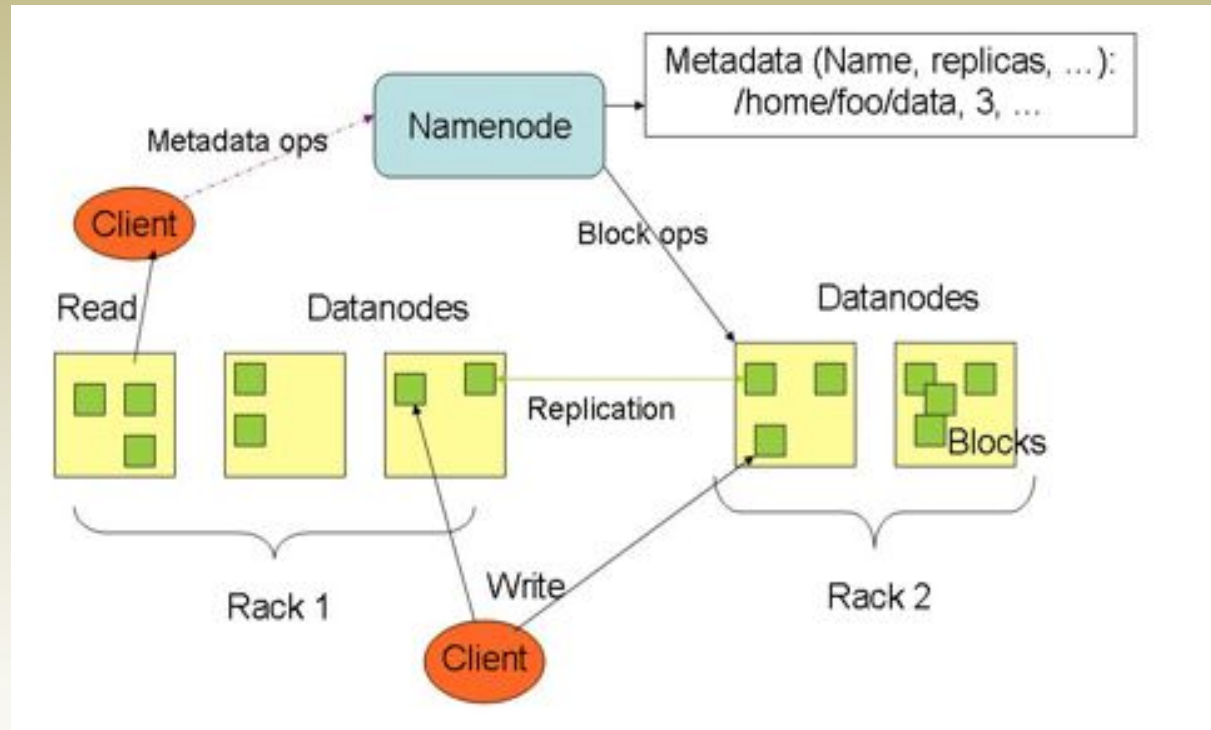
## DataNode:

- Stores the actual data in HDFS
- Can run on any underlying filesystem (ext3/4, NTFS, etc)
- Notifies NameNode of what blocks it has
- NameNode replicates blocks 2x in local rack, 1x elsewhere





# HDFS



# HDFS Replication

## Replication Strategy:

- One replica on local node
- Second replica on a remote rack
- Third replica on same remote rack
- Additional replicas are randomly placed
- Clients read from nearest replica

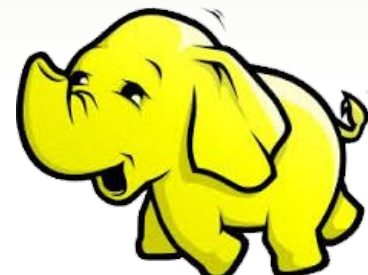
## Use Checksums to validate data – CRC32

- File Creation
  - Client computes checksum per 512 byte
  - DataNode stores the checksum
- File Access
  - Client retrieves the data and checksum from DataNode
  - If validation fails, client tries other replicas

- Client retrieves a list of DataNodes on which to place replicas of a block
- Client writes block to the first DataNode
- The first DataNode forwards the data to the next DataNode in the Pipeline
- When all replicas are written, the client moves on to write the next block

# Hadoop Usage

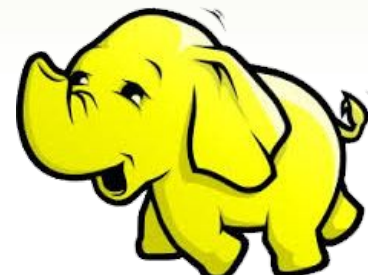
- Hadoop is in use at most organizations that handle big data:
  - Yahoo!
    - Yahoo!'s Search Webmap runs on 10,000 core Linux cluster and powers Yahoo! Web search
  - Facebook
    - FB's Hadoop cluster hosts 100+ PB of data (July, 2012) & growing at  $\frac{1}{2}$  PB/day (Nov, 2012)
  - Amazon
  - Netflix
- Key Applications
  - Advertisement (Mining user behavior to generate recommendations)
  - Searches (group related documents)
  - Security (search for uncommon patterns)





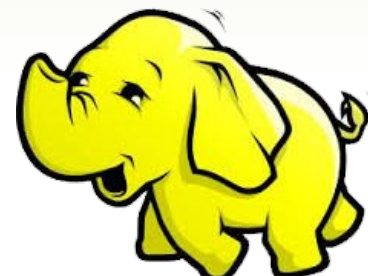
# Hadoop Usage

- Non-realtime large dataset computing:
  - NY Times was dynamically generating PDFs of articles from 1851-1922
  - Wanted to pre-generate & statically serve articles to improve performance
  - Using Hadoop + MapReduce running on EC2 / S3, converted 4TB of TIFFs into 11 million PDF articles in 24 hrs



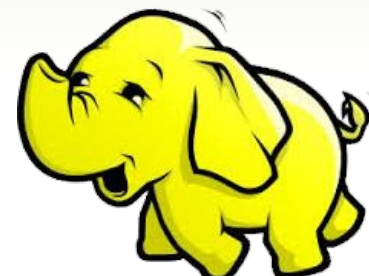
# Hadoop Usage: Facebook Messages

- Design requirements:
  - Integrate display of email, SMS and chat messages between pairs and groups of users
  - Strong control over who users receive messages from
  - Suited for production use between 500 million people immediately after launch
  - Stringent latency & uptime requirements



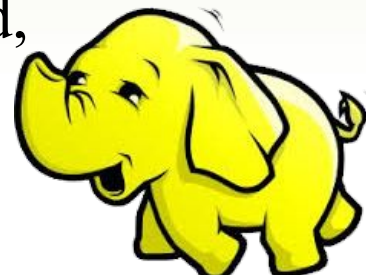
# Hadoop Usage: Facebook Messages

- System requirements
  - High write throughput
  - Cheap, elastic storage
  - Low latency
  - High consistency  
(within a single data center good enough)
  - Disk-efficient  
sequential and random read performance



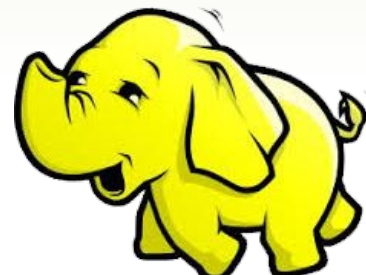
# Hadoop Usage: Facebook Messages

- Classic alternatives
  - These requirements typically met using large MySQL cluster & caching tiers using Memcache
  - Content on HDFS could be loaded into MySQL or Memcached if needed by web tier
- Problems with previous solutions
  - MySQL has low random write throughput... BIG problem for messaging!
  - Difficult to scale MySQL clusters rapidly while maintaining performance
  - MySQL clusters have high management overhead, require more expensive hardware



# Hadoop Usage: Facebook Messages

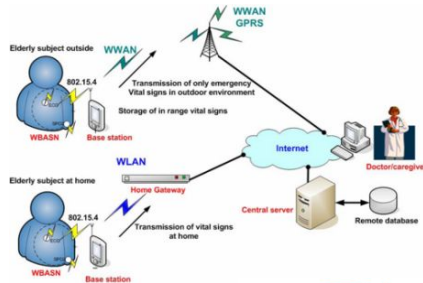
- Facebook's solution
  - Hadoop + HBase as foundations
  - Improve & adapt HDFS and HBase to scale to FB's workload and operational considerations
    - Major concern was availability: NameNode is SPOF & failover times are at least 20 minutes
    - Proprietary "AvatarNode": eliminates SPOF, makes HDFS safe to deploy even with 24/7 uptime requirement
    - Performance improvements for realtime workload: RPC timeout. Rather fail fast and try a different DataNode





# Cloud Computing for Mobile and Pervasive Applications

## Sensory Based Applications



## Mobile Social Networks and Crowdsourcing



## Location Based Services (LBS)



## Augmented Reality



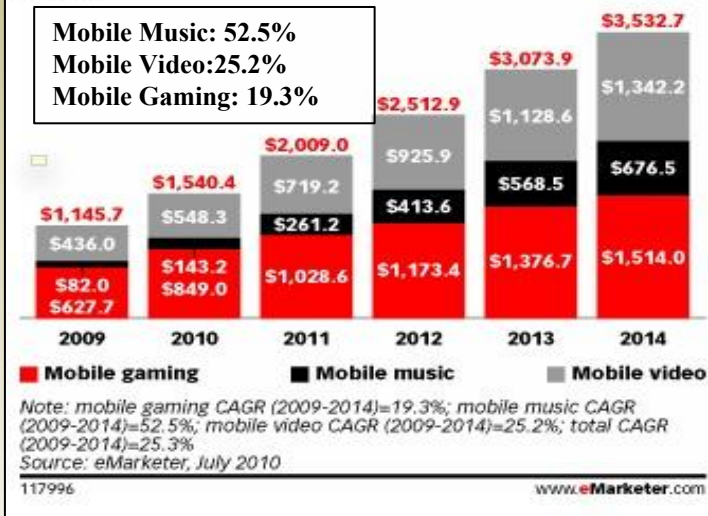
## Multimedia and Data Streaming



## US Mobile Content Revenues, by Segment, 2009-2014

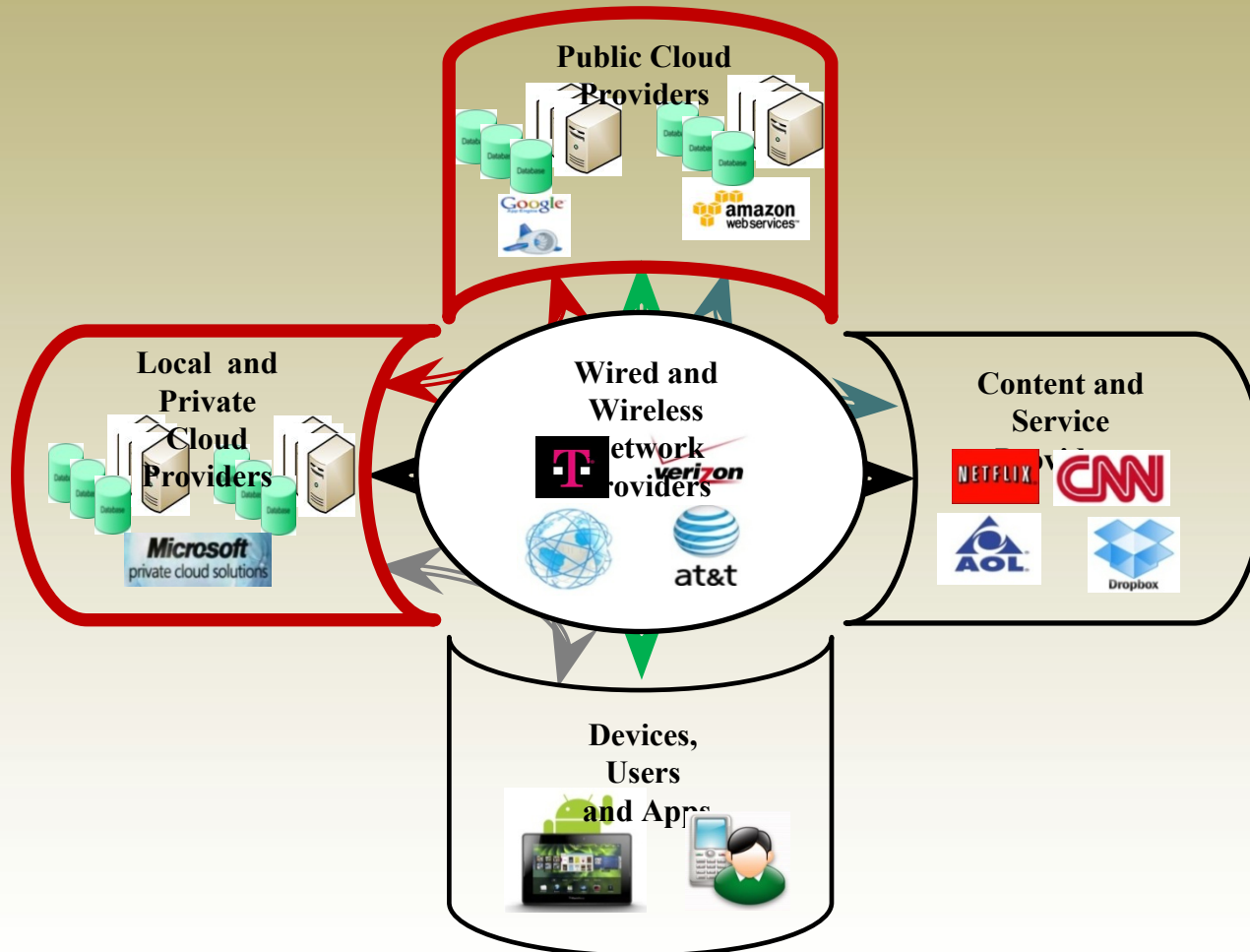
millions

Mobile Music: 52.5%  
Mobile Video: 25.2%  
Mobile Gaming: 19.3%



Due to limited resources on mobile devices,  
we need **outside resources** to empower mobile apps.

# Mobile Cloud Computing Ecosystem



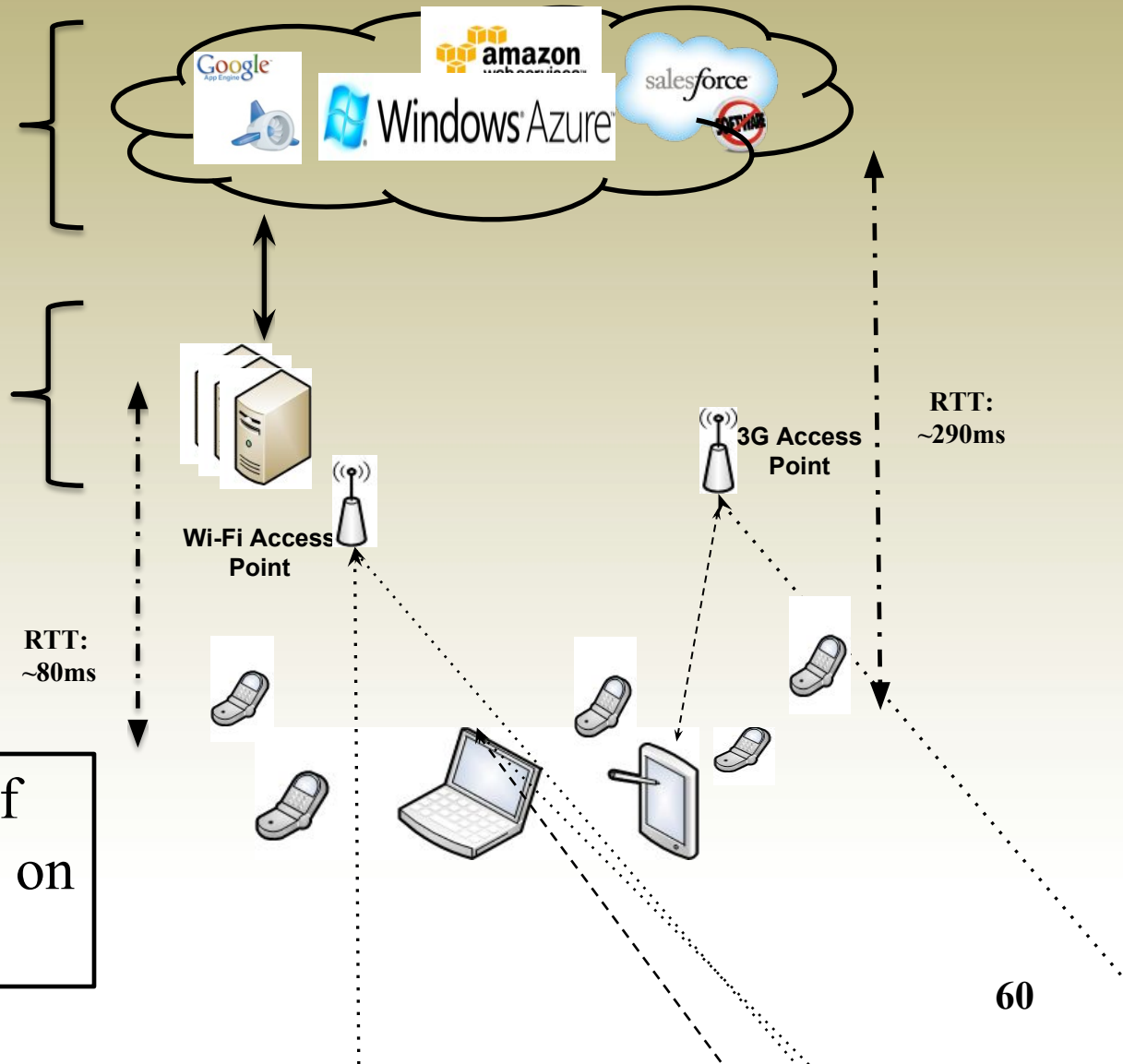
# Tier Cloud Architecture-2

## Tier 1: Public Cloud

(+) Scalable and Elastic  
(-) Price, Delay

## Tier 2: Local Cloud

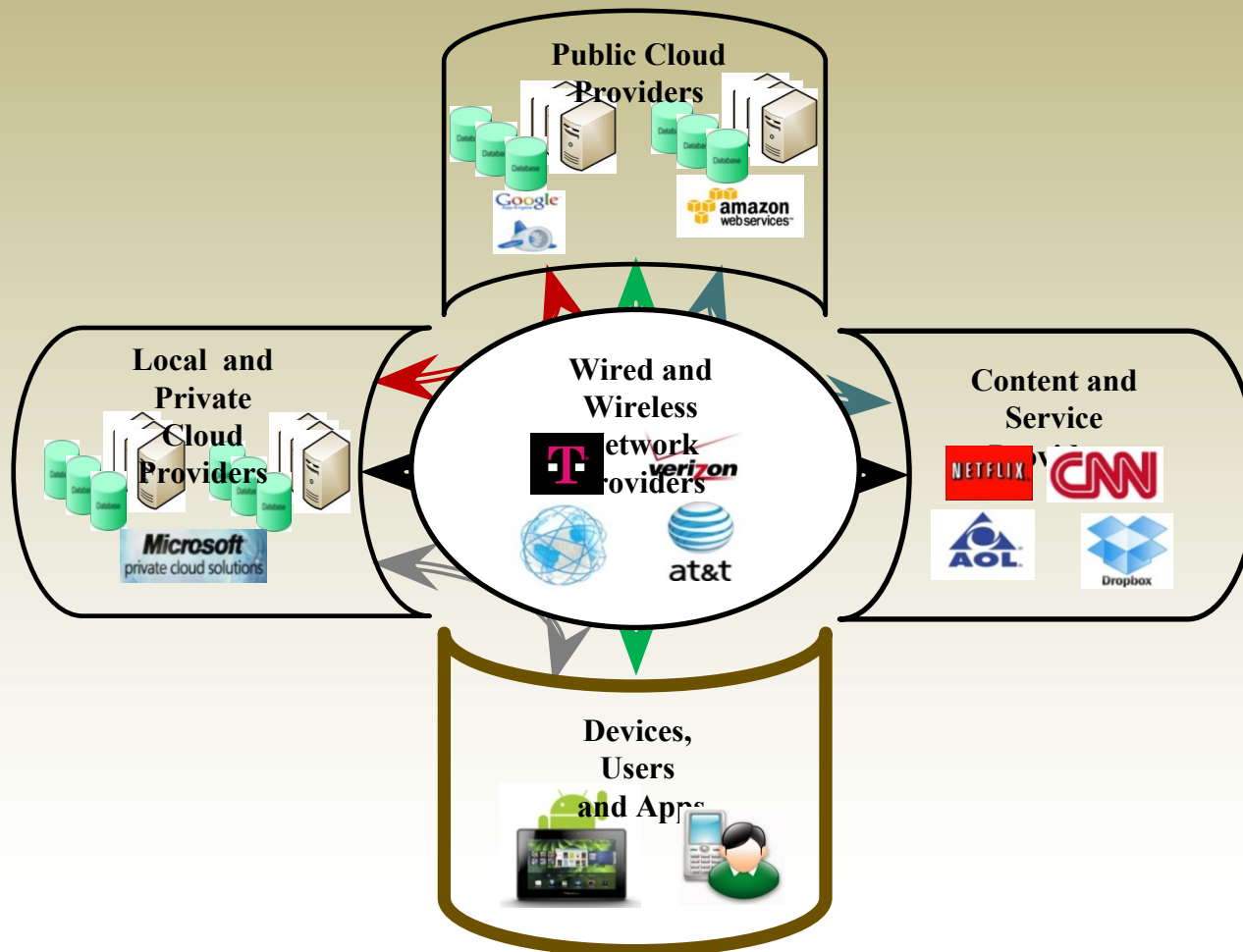
(+) Low Delay, Low Power,  
(-) Not Scalable and Elastic



**IBM:** by 2017 61% of enterprise is likely to be on a tiered cloud



# Mobile Cloud Computing Ecosystem



How can we **Optimally** and **Fairly** assign services to **mobile users** using a **2-tier cloud architecture** (knowing user mobility pattern) considering **power** consumed on mobile device, **delay** users experience and **price** as the main criteria for optimization.

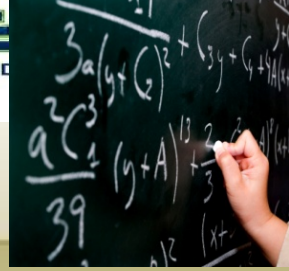
**Modeling Mobile  
Apps**

**Mobility-Aware  
Service Allocation  
Algorithms**

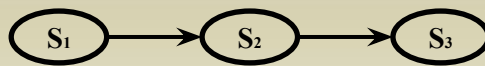
**Scalability**

**Middleware  
Architecture and  
System Design**

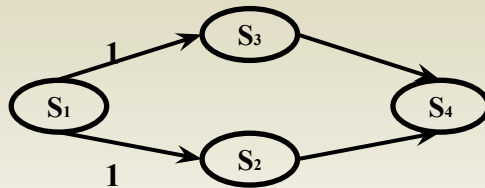
# Modeling Mobile Applications as Workflows



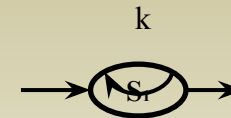
- Model apps as consisting of a series of **logical** steps known as a **Service with** different composition patterns:



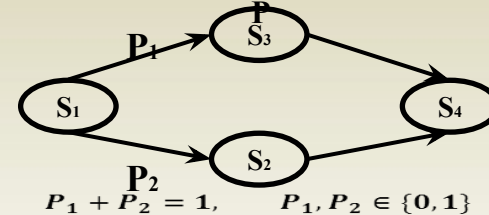
SEQ



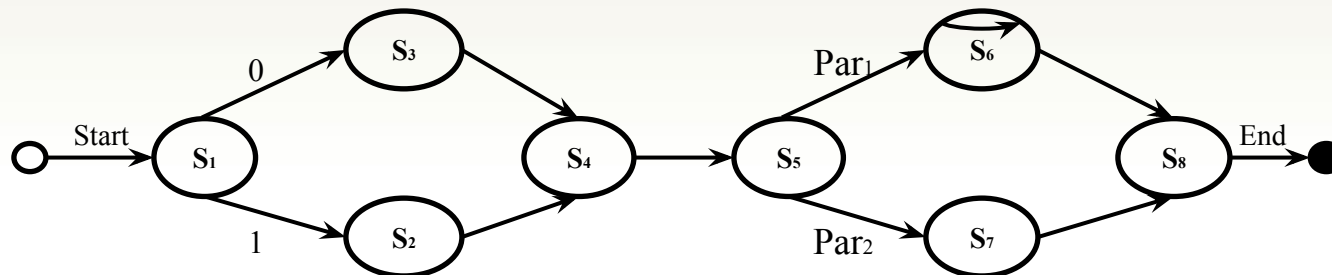
AND: CONCURRENT  
FUNCTIONS



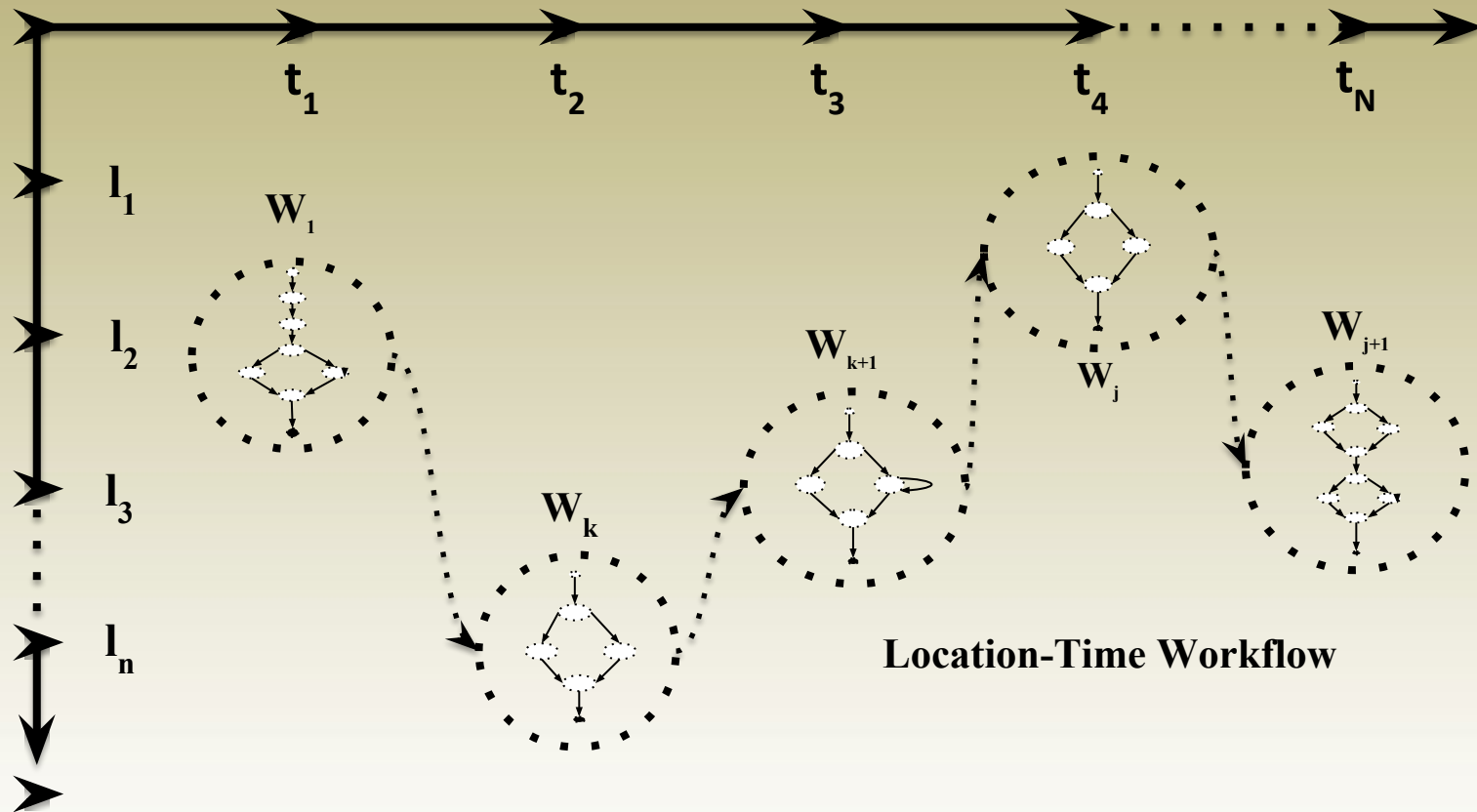
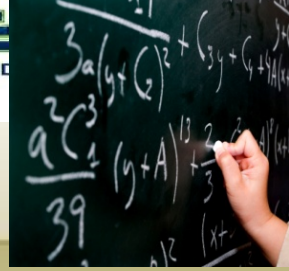
LOO



XOR: CONDITIONAL  
FUNCTIONS



# Modeling Mobile Applications as Workflows



Location-Time Workflow

- It could be formally defined as:

$$W(u_k)_T^L \stackrel{\text{def}}{=} (w(u_k)_{t_{m_1}}^{l_{n_1}}, w(u_k)_{t_{m_2}}^{l_{n_2}}, \dots, w(u_k)_{t_{m_k}}^{l_{n_k}})$$



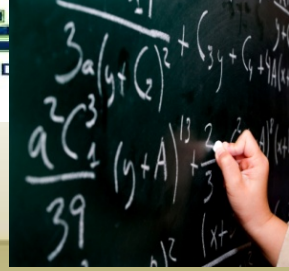
# Quality of Service (QoS)

- The QoS could be defined in **two** different Levels:
  - **Atomic service level**
  - **Composite service level or workflow level.**
- Atomic service level could be defined as (for power as an example):

--	--

- The **workflow QoS** is based on different patterns.

QoS	SEQ	AND (PAR)	XOR (IF-ELSE-THEN)	LOOP



# Normalization

- different QoSes have different dimensions (Price->\$, power->joule, delay->s)
- We need a **normalization** process to make them **comparable**.

The normalized power, price and delay is the real number in **interval [0,1]**.

The higher the normalized QoS the better the execution plan is.

$$\|W(u_k)_{power}\| \stackrel{\text{def}}{=} \begin{cases} \frac{W(u_k)_{power}^{max} - W(u_k)_{power}^{min}}{W(u_k)_{power}^{max} - W(u_k)_{power}^{min}}, W(u_k)_{power}^{max} \neq W(u_k)_{power}^{min} \\ 1, \text{ else} \end{cases}$$

$$\|[W(u_k)_T^L]_{power}\| \stackrel{\text{def}}{=} \begin{cases} \frac{[W(u_k)_T^L]_{power}^{max} - [W(u_k)_T^L]_{power}^{min}}{[W(u_k)_T^L]_{power}^{max} - [W(u_k)_T^L]_{power}^{min}}, [W(u_k)_T^L]_{power}^{max} \neq [W(u_k)_T^L]_{power}^{min} \\ 1, \text{ else} \end{cases}$$

**M. Reza. Rahimi**, Nalini Venkatasubramanian, Sharad Mehrotra and Athanasios Vasilakos, "MAPCloud: Mobile Applications on an Elastic and Scalable 2-Tier Cloud Architecture", In the 5th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2012), USA, Nov 2012.

# Optimal Service Allocation for Single Mobile User



## Fairness Utility

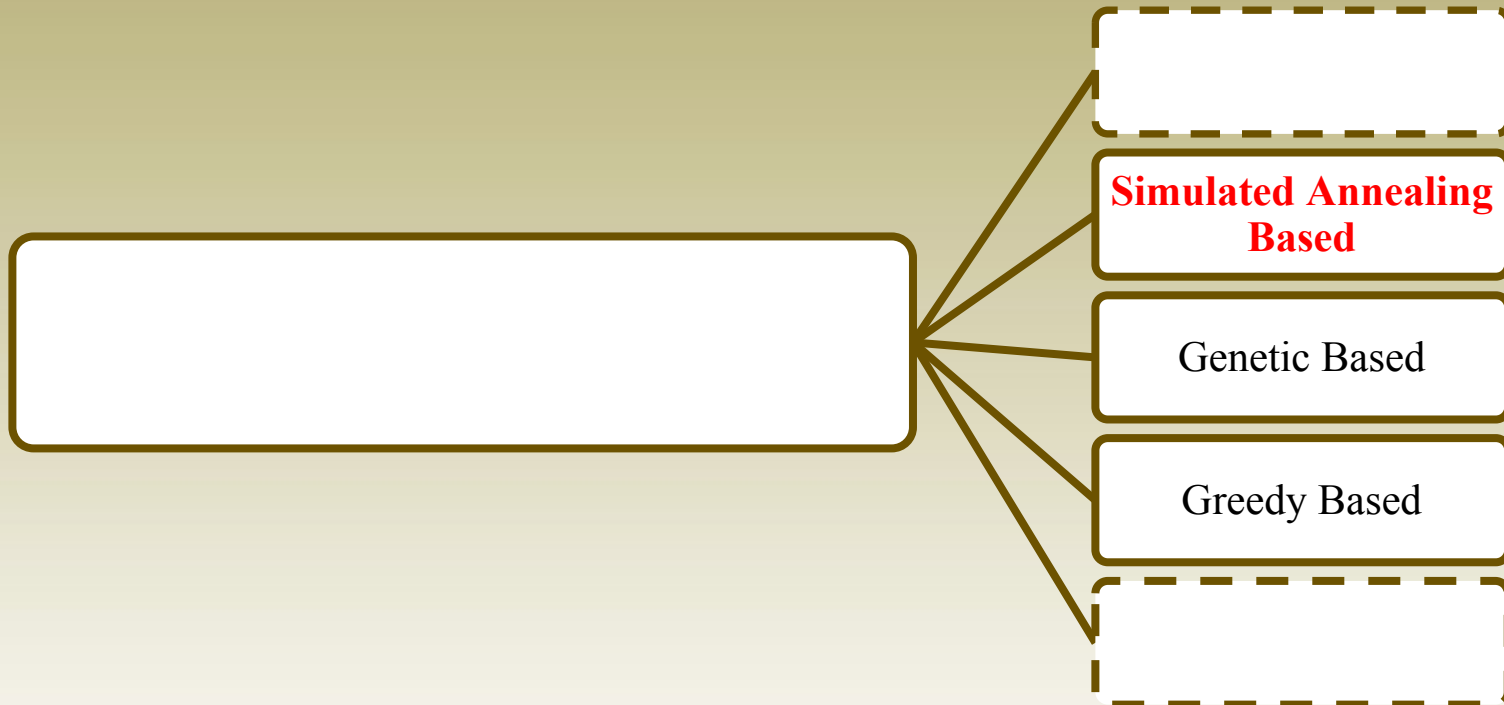
$$\bullet \quad \max \frac{1}{|U|} \sum_{u_k} \min \{ \| [W(u_k)_T^L]_{power} \|, \| [W(u_k)_T^L]_{price} \|, \| [W(u_k)_T^L]_{delay} \| \}$$

Subject to:

$$\left\{ \begin{array}{l} \frac{1}{|U|} [W(u_k)_T^L]_{power} \leq B_{power}, \\ \frac{1}{|U|} [W(u_k)_T^L]_{price} \leq B_{price}, \\ \frac{1}{|U|} [W(u_k)_T^L]_{delay} \leq B_{delay}, \\ \kappa \leq Cap(Local\_Clouds) \\ \kappa \triangleq \text{Number of mobile Users using} \\ \text{services on local cloud} \\ \forall u_k \in \{u_1, \dots, u_{|U|}\} \end{array} \right.$$

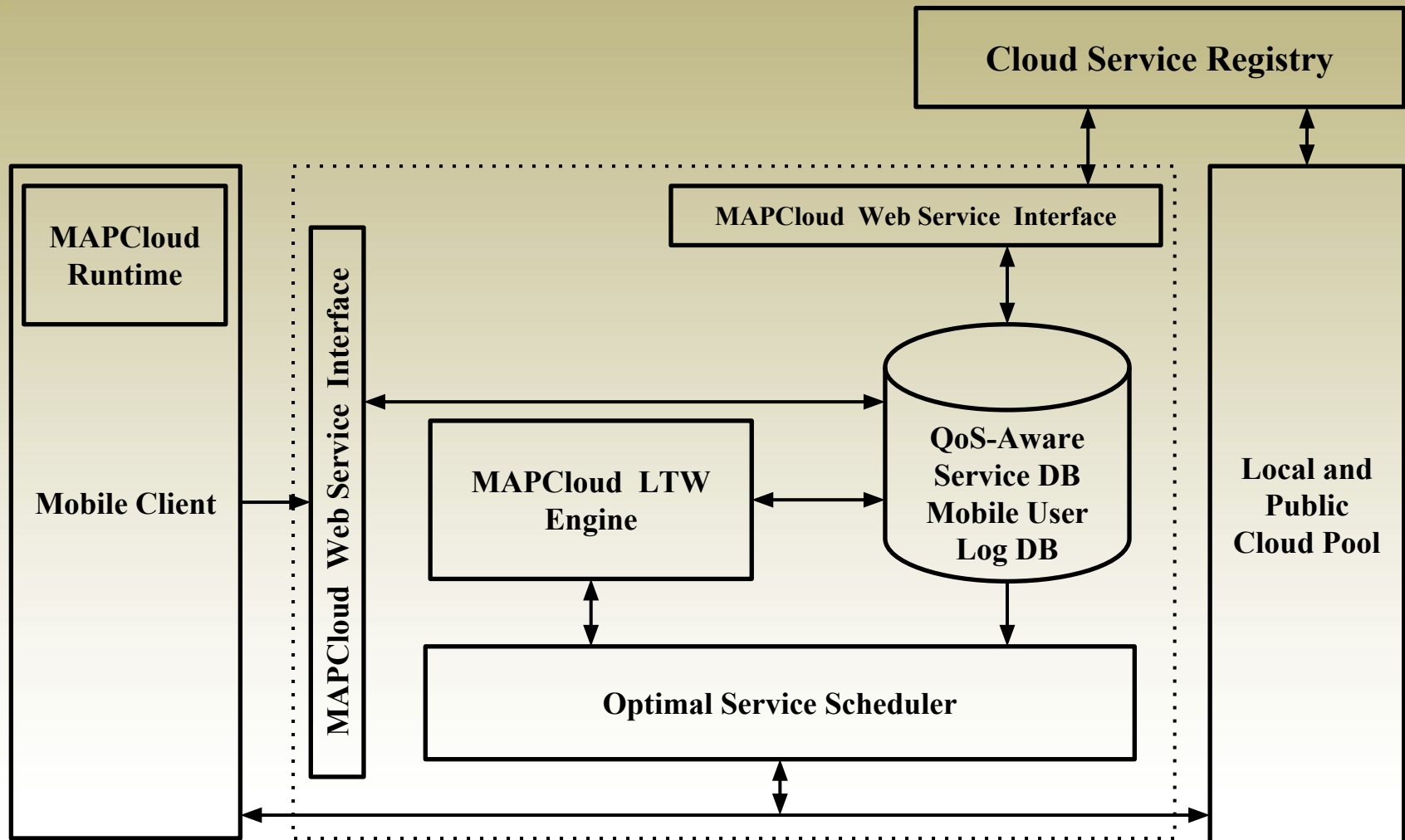
- In this optimization problem our goal is to **maximize the minimum saving of power, price and delay of the mobile applications.**





- ***MuSIC: Mobility Aware Service Allocation on Cloud.***
- based-on a ***simulated annealing*** approach.

# MAPCloud Middleware Architecture





- M. Satyanarayanan, P. Bahl, R. Cáceres, N. Davies " **The Case for VM-Based Cloudlets in Mobile Computing**", PerCom 2009.
- **M. Reza Rahimi**, Jian Ren, Chi Harold Liu, Athanasios V. Vasilakos, and Nalini Venkatasubramanian, "**Mobile Cloud Computing: A Survey, State of Art and Future Directions**", in ACM/Springer Mobile Application and Networks (MONET), Special Issue on Mobile Cloud Computing, Nov. 2013.
- **Reza Rahimi**, Nalini Venkatasubramanian, Athanasios Vasilakos, "**MuSIC: On Mobility-Aware Optimal Service Allocation in Mobile Cloud Computing**", In the IEEE 6th International Conference on Cloud Computing, (Cloud 2013), Silicon Valley, CA, USA, July 2013